UMA ABORDAGEM DE APOIO À EXECUÇÃO PARALELA DE *WORKFLOWS* CIENTÍFICOS EM NUVENS DE COMPUTADORES

Daniel Cardoso Moraes de Oliveira

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Marta Lima de Queirós Mattoso
Fernanda Araujo Baião Amorim

Rio de Janeiro

Junho de 2012

UMA ABORDAGEM DE APOIO À EXECUÇÃO PARALELA DE *WORKFLOWS* CIENTÍFICOS EM NUVENS DE COMPUTADORES

Daniel Cardoso Moraes de Oliveira

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof[a]. Marta Lima de Queirós Mattoso, D.Sc.

_____

Prof[a]. Fernanda Araujo Baião Amorim, D.Sc.

_____

Prof. João Eduardo Ferreira, D.Sc.

_____

Prof. Fabio Andre Machado Porto, D.Sc.

_____

Prof. Nelson Francisco Favilla Ebecken, D.Sc.

_____

Prof. Geraldo Bonorino Xexéo, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

JUNHO DE 2012

Oliveira, Daniel Cardoso Moraes de

Uma Abordagem de Apoio à Execução Paralela de *Workflows* Científicos em Nuvens de Computadores / Daniel Cardoso Moraes de Oliveira. – Rio de Janeiro: UFRJ/COPPE, 2012.

XVII, 182 p.: il.; 29,7 cm.

Orientadores: Marta Lima de Queirós Mattoso

Fernanda Araujo Baião Amorim

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2012.

Referências Bibliográficas: p. 162-182.

1. *Workflows* Científicos. 2. Computação em Nuvem. 3. Computação de Alto Desempenho. I. Mattoso, Marta Lima de Queirós *et. al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Ao meu filho Pedro e a minha esposa Daniele,
por serem os pilares da minha vida.

Agradecimentos

Escrever uma tese de doutorado é antes de tudo um desafio enorme. Desafio este que não é enfrentado sozinho. Muitas são as pessoas a quem devo agradecer nesse momento de conclusão dos trabalhos. Certamente um "obrigado" escrito não é suficiente para agradecer a todos que me ajudaram nesta longa e árdua jornada.

Primeiramente à Deus, por tornar tudo possível.

Ao meu filho Pedro e minha esposa Daniele por serem os amores da minha vida, sempre estando ao meu lado nos momentos bons e ruins, nas horas de dúvida, nos finais de semana sem passeio, nas férias que não podemos viajar, etc. Amo vocês com todo meu coração!

À minha mãe e ao meu pai, que foram os grandes auxiliadores durante toda a minha vida. Vocês me deram (e me dão!) todo apoio necessário para que eu pudesse atingir meus objetivos. Vocês são meu exemplo de vida.

À minha família por ser meu pilar mestre: meus irmãos Bruno e Lucas, meus avós Neli e Edésio, Maria e Martiniano (*in memoriam*), minha madrinha Vânia, meu padrinho Edgard (*in memoriam*), minha tia Deise, meu tio Fernando, minha prima Larissa e meus sogros Daniel e Maria Benedita.

À professora Marta Lima de Queirós Mattoso, considerada por mim, além de orientadora, um exemplo. Além de ter me orientado com muita responsabilidade, dedicação e atenção, suas ideias e sugestões contribuíram de forma decisiva para o meu crescimento acadêmico desde o mestrado. À professora Fernanda Araujo Baião Amorim, sempre ofertando sugestões criativas, uma amizade e orientação precisa.

Aos professores João Eduardo Ferreira, Nelson Francisco Favilla Ebecken, Fabio Andre Machado Porto e Geraldo Bonorino Xexéo por terem aceitado fazer parte desta banca. Agradeço ainda aos professores Alexandre de Assis Bento Lima e Javam de Castro Machado por ter participado do meu exame de qualificação em Dezembro de 2010.

Aos professores que de alguma maneira contribuíram com o meu trabalho: Alexandre Evsukoff, Álvaro Coutinho, Cláudia Werner, Geraldo Xexéo, Guilherme Horta

Travassos, Jano Moreira de Souza, Leonardo Murta, Mario Benevides, Myriam Costa, Patrick Valduriez, Renato Elias, Valmir Carneiro Barbosa e Vanessa Braganholo.

Ao Eduardo Ogasawara por todo o apoio e grande amizade durante todo o doutorado e pela parceria em pesquisas. À Kary Ocaña pela amizade, por toda ajuda na questão dos experimentos desta tese e pelas aulas de bioinformática. Ao Jonas Dias, João Gonçalves, Anderson Marinho, Wallace Martinho, Fernando Chirigati, Ricardo Busquet, Daniel Vega e Vítor Silva pelo trabalho em conjunto e amizade.

Aos alunos de M.Sc. que colaboraram diretamente com a abordagem desta tese: Carlos Eduardo Paulino (*i.e.* Kdú), Flavio Costa, Julliano Pintas e Vitor Gamboa. Um agradecimento especial ao aluno de IC Pedro Cruz Caminha por toda a ajuda no desenvolvimento do SciCumulus.

À Mara Prata, Patrícia Leal, Carolina Vieira, Ana Rabello, Cláudia Prata, Juliana Beltrami, Maria Mercedes, Natália Prata, Solange Santos e Sônia Galliano, por sempre me ajudarem com as questões administrativas na COPPE;

Ao Eduardo Bezerra, João Quadros, Jorge Abreu, Myrna Amorim, Fellipe Duarte e Renato Mauro, pelo apoio no CEFET/RJ;

Aos revisores anônimos de artigos, que de alguma forma contribuíram para a melhoria dos trabalhos aqui referenciados;

À CAPES, pela concessão da bolsa de doutorado entre 2010 e 2012 e ao CNPq, pela concessão da bolsa DTI em 2009, que me permitiram realizar o doutoramento;

Agradeço.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

UMA ABORDAGEM DE APOIO À EXECUÇÃO PARALELA DE *WORKFLOWS* CIENTÍFICOS EM NUVENS DE COMPUTADORES

Daniel Cardoso Moraes de Oliveira

Junho/2012

Orientadores: Marta Lima de Queirós Mattoso

Fernanda Araujo Baião Amorim

Programa: Engenharia de Sistemas e Computação

Grande parte dos experimentos em larga escala existentes modelados como *workflows* científicos são computacionalmente intensivos e necessitam de muitos recursos de computação (comumente distribuídos) que são utilizados para executar milhares de tarefas em ambientes de processamento de alto desempenho (PAD), tais como *clusters* e grades. Nos últimos anos, as nuvens de computadores começaram a oferecer um ambiente de PAD promissor com recursos elásticos que podem ser instanciados, sob demanda, sem a necessidade dos cientistas adquirirem infraestrutura própria. No entanto, a utilização de nuvens para executar *workflows* que demandam PAD apresenta desafios em aberto. Como os próprios cientistas executam os *workflows*, é difícil decidir *a priori* a quantidade de recursos e por quanto tempo os mesmos serão necessários. Nesse cenário há uma necessidade de adaptação frente à flutuação do meio. Além disso, os cientistas têm de gerenciar outras questões, como a captura de proveniência distribuída de forma a garantir a validade e reprodutibilidade do *workflow*. Esta tese apresenta uma abordagem para gerência da execução paralela de *workflows* científicos em ambientes de nuvem de forma adaptativa chamada SciCumulus. O SciCumulus verifica a capacidade computacional disponível, ajusta dinamicamente a distribuição das tarefas e dimensiona o ambiente de nuvem para alcançar um melhor desempenho. Os experimentos mostraram os benefícios da abordagem adaptativa do SciCumulus que apresentou um aumento de desempenho de até 37,9% frente a abordagens tradicionais de paralelismo em nuvens, com a vantagem de oferecer um serviço de captura de proveniência em tempo real.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

AN APPROACH TO SUPPORT PARALLEL EXECUTION OF SCIENTIFIC WORKFLOWS ON CLOUDS

Daniel Cardoso Moraes de Oliveira

June/2012

Advisors: Marta Lima de Queirós Mattoso

Fernanda Araujo Baião Amorim

Department: Systems and Computer Engineering

Most of the existing large-scale scientific experiments modeled as scientific *workflows* are computing intensive and require a huge amount of computing resources (typically distributed) to execute thousands of tasks in High Performance Computing (HPC) environments, such as clusters or grids. In recent years, *cloud* computing environments start posing as a promising HPC environment by providing elastic features that can be instantiated on demand, without the need for scientists to acquire its own infrastructure. However, the effective use of *cloud*s to execute *workflows* that demand HPC presents many open, yet important, challenges. As scientists execute scientific *workflows* that require HPC, it is difficult to decide the amount of resources and how long they will be required beforehand, since the allocation of these resources is elastic. In addition, scientists have to deal with how to capture distributed provenance information and fluctuations in the distributed environment resources. This thesis presents SciCumulus, which is an approach to adaptively manage the parallel execution of scientific *workflows* in *cloud*s. The SciCumulus verifies the available computing power, dynamically adjusts the allocation of tasks and scales the *cloud* environment to achieve a better performance, without compromising distributed provenance gathering. The experiments presented in this thesis showed the benefits of the adaptive approach of SciCumulus that evidenced a performance increase of up to 37.9% compared to traditional approaches that provide parallelism in the *cloud*s with the advantage of offering a service for provenance capture at runtime.

Índice

Índice de Figuras

Índice de Tabelas

Índice de Algoritmos

Capítulo 1 - Introdução

O processo de experimentação é uma das formas usadas para apoiar as teorias baseadas em um método científico (Jarrard 2001). De forma a considerar um corpo de conhecimento como sendo de fato "científico", sua veracidade e validade devem ser comprovadas (Mattoso *et al.* 2010c, Travassos e Barros 2003). No método científico, um experimento científico consiste na montagem de um protocolo concreto a partir do qual se organizam diversas ações observáveis, direta ou indiretamente, de forma a corroborar ou refutar uma dada hipótese científica que foca em estabelecer relações de causa/efeito entre fenômenos (Beveridge 2004, Jr 2002).

Em experimentos científicos tradicionais, como aqueles clássicos de química e física, os cientistas realizam suas pesquisas em laboratórios ou no campo. Entretanto, o ato de "fazer ciência" não se resume mais somente à pesquisa em laboratórios ou no campo. A evolução da ciência da computação nas últimas décadas permitiu a exploração de novos tipos de experimentos científicos baseados em simulação (Deelman *et al.* 2009). Neste novo cenário, os experimentos científicos se tornaram dependentes da utilização maciça de recursos computacionais e de um aparato tecnológico especializado. Um experimento desta categoria é caracterizado pelo uso de simulações do mundo real, realizadas em ambientes virtuais e que são baseadas em modelos computacionais complexos. Em grande parte dos casos, estes modelos se referem ao encadeamento de programas que são utilizados durante as simulações. Cada execução de programa pode consumir e produzir um grande volume de dados.

O encadeamento destes programas em uma simulação, de forma a gerar um fluxo coerente, não é uma tarefa trivial de ser desempenhada e em alguns casos pode inviabilizar a execução do experimento (Gil *et al.* 2007). Desta forma, a execução de experimentos baseados em simulação é apoiada por diversas técnicas e abordagens, como os *workflows* científicos (Deelman *et al.* 2009, Taylor *et al.* 2007a). Um *workflow* científico pode ser definido como o arcabouço funcional que permite a composição de programas em uma sequência de execução com o objetivo de gerar um resultado final. Um *workflow* científico também pode ser visto como uma abstração que modela o encadeamento de atividades e dados para serem executados por um motor de execução especializado. Os *workflows* são uma alternativa atraente para modelar experimentos antes representados como *scripts* ou executados manualmente. Em *workflows*

científicos, essas atividades são geralmente executadas por programas ou serviços computacionais que representam algoritmos e métodos consagrados. Cabe salientar que, neste contexto, os programas que estão associados às atividades de um *workflow* são considerados caixas pretas, *i.e.* uma vez que foram implementados por quem possui conhecimento específico do domínio, não é trivial (e nem o foco de quem especifica os *workflows*) adaptar os programas para que sejam executados pelos motores de execução. Os *workflows* científicos são especificados, executados e monitorados por sistemas complexos chamados de Sistemas de Gerência de *Workflow* Científicos (SGWfC) que possuem um motor de execução acoplado, responsável por invocar cada um dos programas do fluxo.

Devido à complexidade inerente aos modelos que fazem parte dos experimentos científicos, dificilmente estes podem ser executados em ambientes computacionais com um único processador ou cuja execução seja sequencial, uma vez que a execução do experimento neste ambiente acarretaria em um tempo de processamento demasiadamente grande que poderia inviabilizar a pesquisa como um todo (Gorton *et al.* 2008, Gray *et al.* 2005, Hey *et al.* 2009). Desta forma, é uma necessidade real a utilização de ambientes distribuídos, de grande capacidade de processamento e muitas vezes heterogêneos aliados à aplicação de técnicas de paralelismo. Como exemplos destes ambientes podemos citar os *clusters* (Dantas 2005a), as grades computacionais (Dantas 2005b, Foster e Kesselman 2004), os ambientes de computação voluntária (Anderson et al. 2002, Beberg et al. 2009, Cirne et al. 2006), as redes ponto a ponto (do inglês *peer-to-peer* ou P2P) (Valduriez e Pacitti 2005) e mais recentemente as nuvens de computadores (Oliveira *et al.* 2010a, Vaquero *et al.* 2009).

No caso dos *clusters*, grades e redes P2P, devido ao nível de maturidade alcançado pelas tecnologias, existem dezenas de propostas de abordagens para oferecer apoio à execução e a gerência de *workflows* nestes ambientes (Abramson et al. 2008, Anglano e Canonico 2008, Anglano et al. 2008, Cirne et al. 2006, Fahringer et al. 2005, Ogasawara et al. 2011, Oliveira et al. 2010c, 2011c, Smanchat et al. 2009, Sorde et al. 2007, Thain et al. 2002, Wieczorek et al. 2005). Entretanto o mesmo não pode ser dito em relação às nuvens de computadores. Apesar de estarem evoluindo em um ritmo intenso e já estarem sendo utilizadas como ambientes de processamento de alto desempenho (PAD) por diversos cientistas (Antonopoulos e Gillam 2010, Buyya *et al.* 2011, Hey *et al.* 2009, Hoffa *et al.* 2008, Kim *et al.* 2009, Matsunaga *et al.* 2008) as

nuvens ainda carecem de abordagens que ofereçam apoio no que tange a execução e a gerência de *workflows* científicos distribuídos. Portanto, nos deparamos com um novo cenário complexo de apoio à experimentação científica: a gerência de *workflows* científicos executados em paralelo em nuvens de computadores. Cabe salientar que o problema abordado nesta tese é relevante e considerado em aberto por diversos programas internacionais de apoio a chamada e-Ciência (do inglês *e-Science*) nos Estados Unidos (Atkins *et al.* 2003), no Reino Unido (UK eScience 2011) e no Brasil por meio do documento dos grandes desafios do decênio 2006-2016 da SBC (SBC 2006). Além disso, é um problema ressaltado no livro "O Quarto Paradigma" (Hey *et al.* 2009) como sendo de fundamental importância nos próximos anos.

1.1 Caracterização do Problema: Gerência de *Workflows* Científicos em Nuvens

A computação em nuvem (do inglês *cloud computing*) é um paradigma recentemente proposto, e em franca evolução, que oferece a computação como um serviço que pode ser adquirido sob demanda por diferentes tipos de usuários (inclusive cientistas), seguindo um modelo pague-pelo-uso (do inglês *pay-per-use*) (Oliveira *et al.* 2010a). Diferentemente de usuários de *clusters* e grades computacionais, os usuários da nuvem se beneficiam do conceito de elasticidade, uma vez que eles podem facilmente dimensionar a quantidade de recursos utilizados para mais e para menos de acordo com sua demanda de processamento, de forma teoricamente ilimitada (Vaquero *et al.* 2009). As nuvens são fortemente baseadas no conceito de virtualização (Barham et al. 2003) em que máquinas virtuais desempenham um papel fundamental, criando ambientes isolados e independentes implementados sobre uma mesma máquina física. Essa alocação, sob demanda, dos recursos necessários fornece uma nova dimensão para o PAD. Por exemplo, uma das opções para prover um ambiente de PAD é instanciar um conjunto de máquinas virtuais independentes que possam executar tarefas em paralelo, formando um *cluster* virtual. A grande vantagem é que esse *cluster* seria elástico, ou seja, poderia ter seu tamanho aumentado ou diminuído de acordo com a necessidade. Além disso, as máquinas virtuais componentes desse *cluster* podem ser baseadas em imagens (semelhantes a imagens ISO) pré-configuradas, que possuam todos os programas e dependências previamente instalados, diminuindo assim o trabalho com a

configuração e manutenção. Adicionalmente, esse *cluster* virtual possuiria alta disponibilidade garantida pelo provedor de nuvem.

Embora em uma primeira análise (El-Khamra *et al.* 2010, Jackson *et al.* 2010) as nuvens não tenham sido consideradas adequadas para aplicações que demandem PAD, outras avaliações, tais como a de He *et al.* (2010) fornecem evidências de que certas categorias de aplicações já são devidamente atendidas e os novos desenvolvimentos em infraestrutura de nuvem melhorarão significativamente a escalabilidade e o desempenho dos sistemas baseados em nuvem.

Justamente devido à sua capacidade elástica, alta disponibilidade e facilidade de uso as nuvens de computadores já estão sendo adotadas como uma plataforma para executar experimentos científicos de larga escala que demandam PAD. Um exemplo de demanda por PAD em experimentos científicos é a chamada exploração ou varredura de parâmetros. Durante a execução de um *workflow* científico, os cientistas podem precisar explorar o comportamento do modelo científico representado pelo *workflow* por meio do uso de diferentes valores de entrada (parâmetros ou dados distintos). Este comportamento é comumente conhecido como varredura de parâmetros (Samples *et al.* 2005, Walker e Guiang 2007). Nesse cenário, muitas diferentes execuções de *workflows* (ensaios) precisam ser avaliadas.

Como a complexidade dos *workflows* aumenta exponencialmente (por exemplo, a exploração de milhares de parâmetros e, em muitos casos usando estruturas de repetição ao longo de dezenas de atividades complexas), executar esses *workflows* exige a aplicação de técnicas de paralelismo e capacidades de PAD que podem ser oferecidas pelos ambientes de nuvem. Existem vários *workflows* científicos que foram projetados para serem executadas em paralelo (explorando varredura de parâmetros), que demandam recursos de PAD, como o Montage, um *workflow* para análise de dados astronômicos (Jacob *et al.* 2009), *workflows* de análise de cristalografia de raios-X (Oliveira *et al.* 2011a), *workflows* para detecção de genes ortólogos (Cruz *et al.* 2008b) e *workflows* para análise filogenética (Ocaña *et al.* 2011b), sendo este último o estudo de caso desta tese.

Cada uma dessas execuções de *workflows* é composta por centenas ou milhares (e até milhões em alguns casos) de tarefas paralelas que consomem centenas de arquivos e produzem milhares de outros arquivos. Para cada execução na nuvem, os cientistas

devem ser capazes de informar o prazo (o tempo máximo para aguardar a conclusão do *workflow*) e o orçamento limite (a quantidade máxima de recursos financeiros a serem despendidos, uma vez que os mesmos são pagos pelo uso). Baseado nestas informações, as tarefas de um *workflow* devem ser escalonadas, despachadas e executadas em paralelo nas múltiplas máquinas virtuais que compõem o ambiente de nuvem.

Entretanto, executar estes experimentos em nuvens ainda é um problema em aberto, porém de fundamental importância. Essas execuções trazem a tona problemas que devem ser tratados para, de fato, alcançarmos a gerência da execução de um *workflow* científico em um ambiente de nuvem de forma satisfatória. Um primeiro problema é a questão do escalonamento e despacho de tarefas nas nuvens.

O escalonamento de tarefas é um conhecido problema NP (não polinomial), mesmo em suas formas mais simples (Al-Azzoni e Down 2008, Smanchat *et al.* 2009, Yu e Buyya 2006). Desta forma, muitas heurísticas têm sido propostas para resolver o problema de escalonamento em diferentes ambientes. Nos últimos anos muitas heurísticas estáticas têm sido propostas (Assayad *et al.* 2004, Boeres *et al.* 2011, Qin e Hong 2005) para *clusters* e grades. Essas heurísticas geram planos de escalonamento ótimos, ou seja, alocam as tarefas de processamento de um *workflow* em um conjunto de máquinas disponíveis antes da execução propriamente dita do *workflow*.

Entretanto, essa abordagem não é a mais adequada para nuvens. O escalonamento de tarefas de uma execução paralela de *workflows* científicos na nuvem é uma tarefa muito complexa e desafiadora devido a quatro principais razões. Em primeiro lugar, ambientes de nuvem têm um modelo de precificação único que tem que ser considerado no momento do escalonamento para se adequar no tempo limite e orçamento informado pelos cientistas. Em *clusters* e grades, há um investimento inicial (por vezes grande) e um custo operacional pago ao longo do tempo (despesas com o consumo de energia, por exemplo) e estes fatores não impactam no escalonamento de tarefas (*i.e.* o escalonador não se preocupa com essas características ao montar o plano de escalonamento uma vez que o custo já se encontra amortizado). Já no caso da nuvem não existe um investimento inicial alto (investimento zero) e os provedores de nuvem permitem adquirir recursos, sob demanda, com base em um modelo pague-pelo-uso, por exemplo, no qual os usuários pagam por uso (uma janela de tempo fixa ou um valor por hora de CPU utilizada). Desta forma, se a distribuição das atividades for mal feita, pode gerar um custo adicional ao cientista.

Em segundo lugar, as nuvens são ambientes propícios a mudanças e podem ser suscetíveis a alterações de desempenho durante o curso de execução do *workflow*, exigindo soluções de escalonamento adaptativas. O dimensionamento elástico de recursos (*hardware* e *software*) é uma característica chave das nuvens. Para fornecer essa característica, os recursos são alocados e realocados conforme a necessidade do provedor e os usuários não se encontram (em grande parte das vezes) cientes dessas mudanças. Por exemplo, se os cientistas executarem seus *workflows* utilizando instâncias do tipo *spot* do ambiente Amazon EC2 (Amazon EC2 2010), as máquinas virtuais podem ser desalocadas sempre que o provedor precisar de mais capacidade de CPU (em época de Natal, devido a demanda na Amazon, as instâncias *spot* não são disponibilizadas, por exemplo). Outro cenário que pode ocorrer é uma máquina física apresentar problemas e as máquinas virtuais, a ela associadas, serem desligadas ou movimentadas (migradas) enquanto estão sendo utilizadas, o que faz com que haja perda significativa de desempenho. Um exemplo real deste cenário é apresentado na mensagem da Figura 1 no qual uma determinada máquina física está com problemas e a máquina virtual nela implantada será desligada sem que o usuário possa interferir.

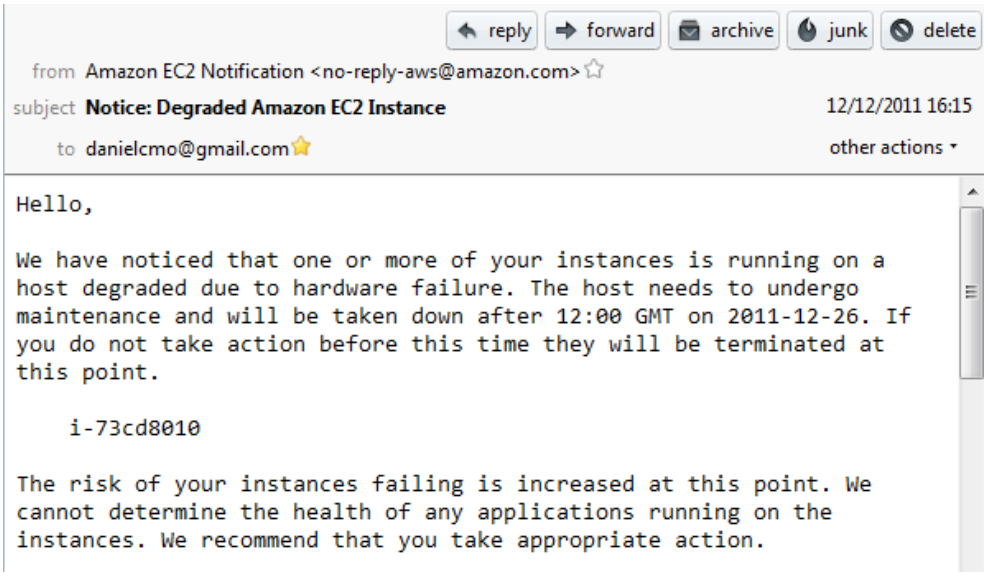


Figura 1 Cenário real de instabilidades no ambiente de nuvem

Outro cenário que ocorre com certa frequência são as reinicializações de máquinas virtuais para aplicação de atualizações de sistema operacional ou qualquer outro *software*. Essas atualizações fazem com que a máquina virtual se torne inativa por algum tempo (em geral alguns minutos), fazendo com que não possa ser utilizada na execução das tarefas do *workflow*. Um exemplo deste cenário é mostrado na Figura 2.

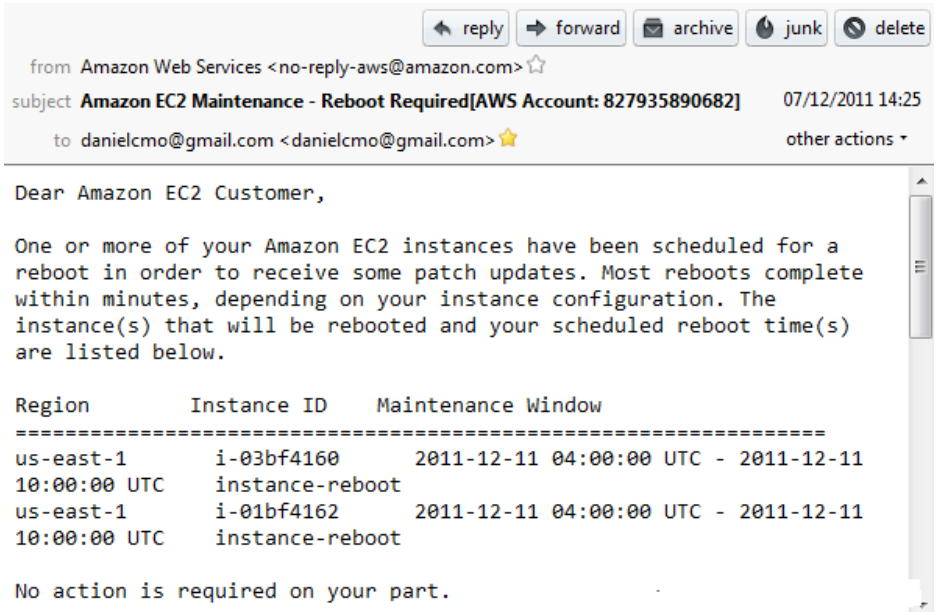


Figura 2 Cenário real de atualização de máquinas virtuais

Em terceiro lugar, nas nuvens (como em grandes *clusters* e supercomputadores) a falhas de máquinas virtuais já não são uma exceção, mas sim uma característica destes ambientes. Embora o provedor de nuvem tente garantir a confiabilidade do ambiente, o escalonamento do *workflow* tem de considerar problemas de confiabilidade ao distribuir tarefas em paralelo para executar em diversas máquinas virtuais. É importante ressaltar que os ambientes de nuvem como a Amazon EC2 (Amazon EC2 2010) possuem mecanismos de tolerância à falha para as maquinas virtuais, mas não para as aplicações que nelas executam. Ou seja, a execução do *workflow* na visão da gerência do ambiente é apenas mais uma aplicação que está sendo executada e que deve ser gerenciada pelo usuário. Um exemplo de que os ambientes não fornecem garantias da aplicação é apresentado na Figura 1 na frase "*We cannot determine the health of any applications running on the instances. We recommend that you take appropriate action*" no qual o provedor informa que não se responsabiliza pela execução de qualquer aplicação na máquina virtual.

Em quarto lugar devemos nos preocupar com a distribuição dos dados de entrada, ou seja, a transferência de dados da máquina do cientista para o ambiente de nuvem. Como as nuvens são acessadas por meio de uma conexão comum com a internet, transferir um grande volume de dados por meio dessa conexão pode ser proibitivo para a execução do experimento. No momento do escalonamento das tarefas, devemos levar esta questão em consideração.

Porém, a execução do *workflow* em paralelo não é o único aspecto importante na gerência do experimento em nuvem. Para se conseguir validar e reproduzir um *workflow*, além de fornecer subsídios fundamentais para realizar o escalonamento adaptativo levando-se em conta as características do ambiente de nuvem, necessita-se possuir dados descritores tanto do ambiente de nuvem quanto dos *workflows* que estão sendo executados (descritores de proveniência). Estes dados devem ser obtidos em um repositório de proveniência (Buneman *et al.* 2001, Davidson e Freire 2008, Freire *et al.* 2008a). Além de auxiliar a distribuição das tarefas para a execução paralela, os dados de proveniência são fundamentais para validar uma determinada execução do *workflow*.

Para que um experimento não seja refutado pela comunidade científica, o mesmo deve ser passível de reprodução sob as mesmas condições em outras instalações por outros grupos de pesquisadores utilizando o método originalmente utilizado. Desta forma, toda a informação relacionada ao experimento, tanto sua definição quanto os dados inicialmente utilizados e gerados durante a sua execução, são fundamentais para que o experimento seja considerado consistente e válido.

Entretanto, para que os cientistas se beneficiem dos descritores de proveniência produzidos ao longo dos experimentos executados em ambientes de nuvem, é necessário que eles sejam capturados, armazenados, classificados e, posteriormente, consultados (Mattoso *et al.* 2010c). Um dos maiores problemas enfrentados pelos cientistas na gerência dos experimentos em nuvem é a coleta dos descritores de proveniência do experimento cientifico. Essa coleta de descritores ainda é uma questão em aberto neste novo paradigma de computação (Oliveira *et al.* 2010a).

Essa tarefa se torna demasiadamente complexa quando lidamos muitas vezes com centenas ou milhares de máquinas virtuais executando diferentes tarefas e produzindo milhares de arquivos em paralelo, também chamada de computação de muitas tarefas (Raicu *et al.* 2008). Outro desafio está na identificação das características do ambiente. Em nuvens públicas, o ambiente de nuvem se apresenta como uma caixa-preta (informações não são publicadas) e coletar estas informações se torna um desafio. Por exemplo, como saber qual programa está instalado em uma determinada imagem de

máquina virtual? Qual a relação real de processamento entre uma máquina do tipo *micro* e uma máquina do tipo *large* no ambiente Amazon EC2?[1]

Outro problema inerente ao ambiente de nuvem é relativo à configuração do ambiente. Quando executamos um experimento em um *cluster* ou grade, o ambiente já se encontra configurado e pronto para uso. Esse cenário nem sempre é verdadeiro quando estamos lidando com nuvens. Em muitos casos o ambiente deve ser todo criado a partir do zero e dimensionar a quantidade de recursos a serem utilizados é um desafio.

Apesar de alguns SGWfC possuírem recursos para execução paralela, nenhum deles oferece um arcabouço para execução de *workflows* em nuvens que leve em conta todas essas características importantes. Ou as soluções existentes não proveem recursos de execução em nuvem, como acontece com o Kepler (Altintas *et al.* 2004a, Ludascher *et al.* 2006), o VisTrails (Callahan *et al.* 2006) e o Taverna (Hull *et al.* 2006a) ou proveem recursos para a execução paralela de *workflows* em nuvens como se os estivessem executando em um *cluster* (sem aproveitar as questões como elasticidade ou tratar variações no desempenho) como é o caso do Pegasus (Deelman *et al.* 2007) e do Swift (Zhao *et al.* 2007). Nota-se então que existe uma carência de soluções que apoiem a execução de *workflows* científicos em nuvens de computadores de forma adaptativa e com mecanismos de captura de proveniência distribuída. Desta forma, a questão de pesquisa a ser investigada nesta tese é:

"No contexto de experimentos científicos em larga escala executados em ambientes de nuvens computacionais, é possível promover a gerência distribuída do workflow científico por meio da adoção de uma abordagem que controle a execução de tarefas em paralelo na nuvem considerando as características únicas do ambiente, capture a proveniência gerada a partir desta execução em várias máquinas virtuais e utilize um modelo de proveniência que represente descritores tanto do workflow quanto do ambiente em que o mesmo está sendo executado, tornando possível o aumento de desempenho da execução e a análise de proveniência destes experimentos por parte dos cientistas?".

---

[1] Apesar de esta informação estar contida na página da Amazon EC2, ela não é 100% garantida, justamente devido às flutuações a que o ambiente está suscetível.

Nesta tese, o foco se dá no estudo e no desenvolvimento em infraestruturas reais de nuvem como a Amazon EC2, pois é um fato que as nuvens de computadores já atingiram um nível aceitável (mas não ideal) de maturidade que as confere um estado que não é mais demasiadamente transiente, desta forma não necessitando fazer uso de um ambiente de simulação como o *Cloud*Sim (Buyya *et al.* 2009).

1.2 Serviços para Execução Distribuída de *Workflows* Científicos em Nuvens Computacionais

A hipótese geral deste trabalho é que é possível prover capacidades de PAD e gerenciar a execução paralela de *workflows* científicos de maneira eficiente em um ambiente distribuído de nuvens de computadores por meio da adoção de uma infraestrutura adaptativa especializada (serviços) para o escalonamento, despacho, monitoramento e captura de proveniência distribuída das tarefas paralelas de um *workflow* científico. No cenário de experimentos científicos em larga escala, gerenciar um experimento de maneira eficiente implica fazer-se valer das características do ambiente de nuvem e do próprio *workflow* para gerar um plano de execução que melhor se adeque ao cenário. Técnicas de escalonamento aliadas a um modelo de custo específico para o ambiente de nuvem permitem a escolha de um plano de execução ao menos "sub-ótimo" (ou que pelo menos não seja inválido). Assim, a adoção de uma abordagem adaptativa para executar *workflows* científicos permite realizar a gerência da execução paralela de *workflows* científicos de modo sistemático, *i.e.*, de modo que o cientista não tenha que se preocupar com codificações paralelas, com configurações do ambiente ou com a captura de proveniência, mas somente com a especificação do experimento.

Como base para a formulação desta hipótese e das propostas que são apresentadas nesta tese foram realizados inúmeros experimentos de execução paralela de *workflows* científicos em ambientes P2P, de *clusters* e de grades computacionais (Barbosa *et al.* 2009, Coutinho *et al.* 2010, 2011, Dias *et al.* 2010a, 2010c, 2010b, Guerra *et al.* 2009, Ogasawara *et al.* 2009a, Silva *et al.* 2011a, 2011b). Estes experimentos forneceram subsídios para uma proposta mais consistente no que se refere à execução paralela de *workflows* em nuvens.

Nesta tese, é apresentado um conjunto de serviços que implementa uma abordagem adaptativa para prover escalonamento e captura de proveniência de *workflows* executados em paralelo em ambientes de nuvem. Esta abordagem é baseada

em um modelo de custo ponderado de três objetivos que considera o tempo de execução, a confiabilidade do ambiente e o custo monetário simultaneamente. Esse modelo de custo foi inspirado nas ideias de Boeres *et al.* (2011). Com base neste modelo de custo é proposto um escalonamento guloso adaptativo para distribuir as diversas tarefas de um *workflow* nas máquinas virtuais juntamente com um serviço de dimensionamento de recursos que instancia e destrói máquinas virtuais para cumprir os prazos e orçamentos informados pelos cientistas.

O modelo de custo proposto e o algoritmo de escalonamento guloso são fortemente baseados nos dados de proveniência capturados e consultados em tempo de execução. Além de fornecer informações que permitem a reprodução de experimentos, o repositório de proveniência também fornece as informações necessárias para que, a partir de execuções anteriores de *workflows*, se possa estimar o tempo de execução das tarefas correntes, analisar o volume de dados produzidos por uma tarefa ou descobrir quais as tarefas que compartilham arquivos de entrada a fim de serem agrupadas. Esse tipo de informação é fundamental para o modelo de custo e o algoritmo de escalonamento propostos e permite um ajuste fino das configurações.

Desta forma, foi projetado e desenvolvido o SciCumulus, um arcabouço para a execução paralela de *workflows* científicos em nuvens computacionais. Diferente das soluções atuais, o SciCumulus é independente de SGWfC. Ele cria automaticamente máquinas virtuais e *clusters* virtuais com base nas restrições impostas pelos cientistas. Além disso, o SciCumulus gera os dados de proveniência em tempo de execução, ou seja, registra informações sobre o *workflow* que está sendo executado enquanto o mesmo se encontra em execução. Esta tese contextualiza sua pesquisa na área de "ciência apoiada pela computação" (do inglês *e-science* ou *cyberinfrastructure*), contribuindo nesta área por apresentar:

i. Um modelo de custo ponderado multiobjetivo para escalonamento de tarefas de *workflows* científicos em nuvens, considerando o tempo total de execução, custo monetário e confiabilidade do ambiente;
ii. Uma abordagem de escalonamento gulosa e adaptativa para explorar o espaço de escalonamentos alternativos considerando as mudanças no ambiente de nuvem e o uso do modelo de custos proposto;
iii. Um modelo de proveniência que considera os dados descritivos do ambiente de nuvem e os dados referentes à estrutura e execução dos *workflows* científicos;

iv. Uma arquitetura independente de SGWfC e serviços que implementem o modelo de custos e o algoritmo de escalonamento proposto, aliado a serviços de dimensionamento de recursos, re-execução de tarefas e captura de proveniência distribuída e;
v. Uma avaliação experimental exaustiva do *workflow* de análise filogenética com o SciCumulus. Para esta avaliação experimental executamos o SciPhy (Ocaña *et al.* 2011b) em um *cluster* virtual de 128 núcleos virtuais no ambiente de nuvem Amazon EC2, mostrando os benefícios da abordagem proposta.

1.3 Metodologia de Pesquisa

A metodologia de pesquisa adotada é definida como o conjunto de procedimentos utilizados para a obtenção do conhecimento científico (Jr 1991). Esta tese se encontra na área de Banco de Dados e faz parte do domínio da Ciência da Computação. A tese adota como estratégia de pesquisa o estudo de caso e o método de pesquisa empregado é o quantitativo (Juristo e Moreno 2001), uma vez que toda a avaliação da abordagem proposta está pautada em valores de desempenho obtidos por meio da execução do *workflow* SciPhy em paralelo em um ambiente de nuvem real. As etapas realizadas nesta pesquisa podem ser resumidas em:

i. Levantamento do estado da arte na execução de *workflows* científicos em nuvens computacionais;
ii. Definição e conceituação das etapas do ciclo de vida de um experimento científico executado em nuvens de computadores, caracterizando quais são os problemas e desafios de cada fase;
iii. Elaboração do modelo de custo e dos algoritmos adaptativos para distribuição das tarefas do *workflow* em nuvens para execução em paralelo;
iv. Levantamento dos descritores do ambiente de nuvem e do *workflow* científico e consequente especificação do modelo de proveniência;
v. Projeto e desenvolvimento de um arcabouço (SciCumulus) composto de uma série de serviços para apoiar a gerência distribuída do experimento em nuvens computacionais.
vi. Execução de *workflows* de bioinformática (*workflow* SciPhy) utilizando a infraestrutura de gerência de experimentos em nuvens.

1.4 Organização da Tese

Além deste capítulo introdutório, esta tese está organizada em outros oito capítulos. O Capítulo 2 apresenta conceitos relacionados a experimentos científicos, *workflows* científicos e proveniência de dados. O Capítulo 3, referente à abordagem proposta, detalha o modelo de custos ponderado proposto enquanto que o Capítulo 4 apresenta os algoritmos de escalonamento gulosos adaptativos que utilizam o modelo de custo como base. O Capítulo 5 apresenta o SciCumulus, uma abordagem para a execução de *workflows* desenvolvida para explorar paralelismo de dados e de parâmetros em larga escala em ambientes de nuvens computacionais. O Capítulo 6 apresenta conceitualmente o *workflow* SciPhy, utilizado como estudo de caso nesta tese. O Capitulo 7 apresenta a avaliação experimental da abordagem proposta, contemplando a avaliação do modelo de custos, do algoritmo de escalonamento, das heurísticas de configuração do ambiente e da análise dos dados de proveniência. O Capítulo 8 apresenta os trabalhos relacionados. Finalmente, o Capítulo 9 conclui a tese apresentando os principais resultados alcançados e os desdobramentos para diversos trabalhos futuros na direção dos desafios da gerência de experimentos científicos em nuvens computacionais.

Capítulo 2 - Fundamentação Teórica de Experimentos Científicos em Larga Escala

Este capítulo tem como objetivo fornecer a fundamentação teórica acerca do domínio de experimentos científicos em larga escala e o ferramental associado a estes. Este capítulo está organizado da seguinte forma. A Seção 2.1 apresenta o conceito de experimento científico baseado em simulação que é o objeto de estudo desta tese, enquanto que a Seção 2.2 define o ciclo de vida deste tipo de experimento. A Seção 2.3 caracteriza o conceito de *workflows* científicos. A Seção 2.4 apresenta o conceito de proveniência de dados no contexto de *workflows* científicos e a Seção 2.5 discute sobre ambientes de processamento de alto desempenho que podem ser utilizados para processamento distribuído de *workflows* científicos. A Seção 2.6 define o conceito de computação em nuvem, conceito este central para entendimento desta tese. A Seção 2.7 propõe um ciclo de vida para o experimento executado em nuvens de computadores enquanto que a Seção 2.8 define o conceito de *cloud activity*. E finalmente, a Seção 2.9 define a questão do paralelismo por varredura de parâmetros em *workflows* científicos.

2.1 Experimento Científico Baseados em Simulação

Considerando a evolução da ciência nas últimas décadas, o ato de “fazer ciência” vem mudando com uma grande velocidade a cada ano. O volume de dados científicos hoje processados alcançou patamares históricos (Wallis *et al.* 2010). Os experimentos científicos em larga escala necessitam de recursos computacionais cada vez mais potentes para a sua execução e são, geralmente, realizados por equipes geograficamente dispersas (Gorton *et al.* 2008).

Formalmente, um experimento científico pode ser definido como "um teste executado sob condições controladas, que é realizado para demonstrar uma verdade conhecida, examinar a validade de uma hipótese, ou determinar a eficácia de algo previamente não explorado" (Soanes e Stevenson 2003). Um experimento também pode ser definido como "uma situação, criada em laboratório, que visa observar, sob condições controladas, a relação entre os fenômenos de interesse" (Jarrard 2001). Neste contexto, a palavra "controle" ou o termo “condições controladas” são usados para indicar que há esforços para eliminar, ou pelo menos reduzir ao máximo possível, os

erros ocasionais durante uma observação planejada (Juristo e Moreno 2001). Com base nessas definições, podemos concluir que um experimento científico está estritamente associado a um conjunto de ações controladas (*i.e.* um protocolo). Essas ações controladas incluem variações de testes, e seus resultados são geralmente comparados entre si para aceitar ou refutar uma hipótese científica. Os experimentos científicos são a maior preocupação da comunidade científica (Mattoso *et al.* 2010c). Existem diversos tipos de experimentos científicos, a saber: *in vivo*, *in vitro*, *in virtuo* e *in silico* (Travassos e Barros 2003).

Nesta tese estamos interessados em uma categoria de experimentos científicos em especial: os experimentos *in silico* ou baseados em simulação (Travassos e Barros 2003). No contexto desta tese, o termo "experimento científico" será consistentemente utilizado para referenciar experimentos científicos baseados em simulação. Neste tipo de experimento, tanto o ambiente quanto os participantes do experimento são simulados em ambientes computacionais. Este tipo de experimento é utilizado nos mais diversos domínios científicos como, por exemplo, análises filogenéticas (Ocaña *et al.* 2011b), genômica comparativa (Ocaña *et al.* 2011a), processamento de sequências biológicas (Lemos *et al.* 2004), estudos na área de saúde (de Almeida-Neto *et al.* 2011, Goncalez *et al.* 2011, Patavino *et al.* 2012, Sabino *et al.* 2011), prospecção de petróleo em águas profundas (Carvalho 2009, Martinho *et al.* 2009, Ogasawara *et al.* 2011, Oliveira *et al.* 2009a), mapeamento dos corpos celestes (Hey *et al.* 2009), ecologia (Hartman *et al.* 2010), agricultura (Fileto *et al.* 2003), busca de genes ortólogos dos tripanosomas causadores de doenças tropicais negligenciadas (Coutinho *et al.* 2011, Dávila *et al.* 2008), dinâmica de fluidos computacional (Guerra e Rochinha 2009a, 2009b, 2010, Guerra et al. 2009, 2012, Lins et al. 2009), estudos fisiológicos (Porto *et al.* 2011), previsão de precipitação (Evsukoff et al. 2011), monitoramento aquático (Pereira e Ebecken 2011) e pesquisa sobre energia escura (Governato *et al.* 2010). Todos esses exemplos podem ser considerados de larga escala por consumirem e produzirem um grande volume de dados e são um ponto de inflexão que merece o desenvolvimento de estudos específicos.

Experimentos em larga escala como os anteriormente citados podem ser executados milhares (ou em alguns casos, milhões) de vezes para produzir um resultado. Como cada execução destas pode demandar muitos recursos e tempo, esses experimentos usualmente requerem técnicas de paralelismo e execução em ambiente de

PAD, como *clusters* e supercomputadores, grades computacionais, ambientes de computação voluntária e mais recentemente as nuvens de computadores. Esses experimentos são especialmente complexos de serem gerenciados pelo cientista devido às questões de infraestrutura computacional como a grande quantidade de programas envolvidos na simulação e a quantidade de recursos computacionais necessários. Não é trivial controlar o volume de dados manipulados, evitar e contornar falhas de execução, etc. Essa tarefa se torna ainda mais complexa se necessitarmos capturar e armazenar descritores da execução para garantir a reprodutibilidade do mesmo (Davidson e Freire 2008). Essa captura e armazenamento é uma característica fundamental para que um experimento seja considerado "científico" de fato.

Em grande parte dos casos, esses experimentos são representados por meio do encadeamento de múltiplas combinações de programas que podem consumir grandes quantidades de dados, e são distribuídos em ambientes computacionais heterogêneos. Nesses experimentos, cada programa pode ser executado consumindo um grupo específico de parâmetros e dados, cada qual com a sua própria semântica e sintaxe. A saída de um programa é normalmente utilizada como entrada para outro programa no encadeamento (Cavalcanti *et al.* 2005).

Os *workflows* científicos são uma alternativa atraente para representar os encadeamentos de programas ao invés de usarmos uma abordagem *ad hoc* manual ou baseada em *scripts*. Os *workflows* científicos podem ser definidos como uma abstração para modelar o fluxo de atividades e de dados em um experimento. Em *workflows* científicos, essas atividades são geralmente programas ou serviços que representam algoritmos e métodos computacionais sólidos (Barker e van Hemert 2008).

Esses *workflows* são controlados e executados pelos Sistemas de Gerência de *Workflows* Científicos (SGWfC), que são mecanismos complexos que visam apoiar a configuração e execução dos *workflows*. Há muitos SGWfC disponíveis, como o Kepler (Altintas *et al.* 2004a), o Taverna (Hull *et al.* 2006a), o VisTrails (Callahan *et al.* 2006), o Pegasus (Deelman *et al.* 2007), o Askalon (Fahringer *et al.* 2005), o P-Grade (Glatard *et al.* 2007, Kertész *et al.* 2007), o DagMan (Couvares *et al.* 2007), o Triana (Taylor *et al.* 2007b), o WOODS (Medeiros *et al.* 2005) e o Swift (Zhao *et al.* 2007), cada um com suas próprias características, vantagens e desvantagens.

Nesta tese, o conceito de experimento científico engloba o conceito de *workflow*, e não podem ser tratados como um sinônimo. A execução de um *workflow* pode ser vista como um conjunto de ações controladas do experimento. Assim, a execução de um *workflow* pode ser definida como um dos ensaios (do inglês *trials*) realizados no contexto de um experimento científico para avaliar uma ação controlada. O conjunto de ensaios representado por cada execução distinta de um *workflow* define um experimento científico. Portanto, um *workflow* científico é parte de um experimento. O ciclo de vida do experimento científico envolve várias fases, incluindo a execução dos ensaios, e é detalhado na próxima seção.

2.2 Ciclo de Vida do Experimento Científico

Nesta seção apresentamos um dos modelos existentes para representar o ciclo de vida de um experimento científico (Mattoso *et al.* 2010c). A Figura 3 apresenta o ciclo de vida de um experimento cientifico em larga escala, que consiste essencialmente em múltiplas interações que são percorridas pelo cientista várias vezes no curso do experimento. Na Figura 3, as principais fases podem ser identificadas: a execução, a composição e a análise. Cada fase possui um subciclo independente, que é percorrido em momentos distintos do curso do experimento.

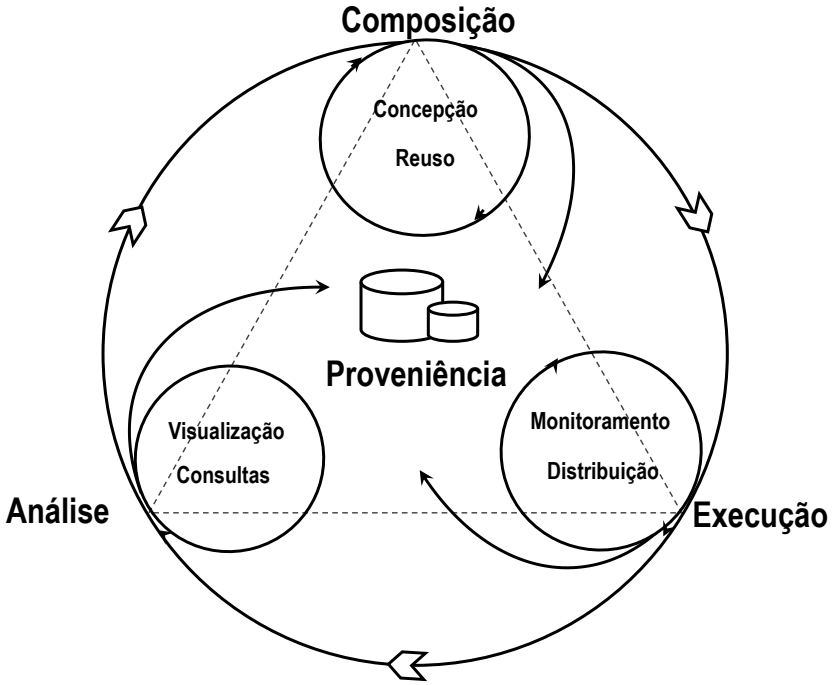


Figura 3 Ciclo de vida do experimento científico adaptado de Mattoso *et al.* (2010c)

A fase de composição é responsável pela estruturação e criação de todo o experimento, mas principalmente é responsável por instituir a sequência lógica de atividades, os tipos de dados de entrada e parâmetros que devem ser fornecidos, e os tipos de dados de saída que são gerados (*i.e.* definir a estrutura do *workflow* científico). Esta fase pode ser ainda decomposta em duas subfases: a concepção e o reuso. A concepção é responsável pela criação do experimento enquanto que o reuso é responsável por recuperar um experimento já existente e adaptá-lo para um novo propósito.

A fase de execução, foco maior desta tese, é responsável por tornar concreta e executável uma especificação de *workflow* de forma que possa ser executada por um SGWfC ou máquina de execução. Portanto, nesta fase definimos exatamente os dados de entrada e os parâmetros a serem entregues ao SGWfC a fim de executar o experimento, ou seja, criamos uma instância do *workflow* para ser executada. Esta fase pode ser decomposta em duas subfases autoexplicativas: distribuição e monitoramento. A subfase de distribuição está relacionada com a necessidade da execução de atividades do *workflow* em ambientes de PAD, principalmente devido a necessidades de desempenho. A subfase de monitoramento está relacionada à necessidade de verificar periodicamente o estado atual da execução do *workflow*, uma vez que este pode executar por um longo tempo.

Finalmente, a fase de análise é responsável por estudar os dados gerados pelas fases de composição e execução. Esta fase é altamente dependente dos dados de Proveniência (Freire *et al.* 2008b) que foram gerados nas fases anteriores (a definição de proveniência será mais bem explicada a seguir nesta tese). Esta fase pode ser decomposta em duas novas subfases: visualização e consulta. Além disso, na fase de análise, o cientista pode enfrentar duas situações diferentes ao analisar os resultados de um experimento: (i) a hipótese analisada é provavelmente correta ou (ii) baseado nos resultados, a hipótese é refutada.

No primeiro caso, os cientistas provavelmente terão que executar novamente o *workflow* para efetivamente validar a hipótese enquanto que no segundo caso, uma nova hipótese deverá ser criada. No caso de existir a necessidade de várias execuções de um *workflow* para validar uma hipótese, todas as execuções (com diferentes parâmetros e conjuntos de dados) devem ser conectadas ao mesmo experimento cientifico.

2.3 *Workflows* Científicos

A definição padrão para o conceito de *workflow*, segundo o *Workflow Management Coalition* (WfMC) (WfMC 2009), é: "A automação de um processo de negócio, completo ou apenas parte dele, através do qual documentos, informações ou tarefas são transmitidos de um participante a outro por ações, de acordo com regras procedimentais".

Embora nesta tese estejamos interessados na visão científica do uso de *workflows*, o termo "*workflow*" se originou no processo de automação de escritórios (Aalst e Hee 2002, Deelman *et al.* 2009) e essa tecnologia originou-se por volta do ano de 1970. Originalmente o foco era oferecer soluções voltadas para a geração, armazenamento e compartilhamento de documentos em uma empresa, com o objetivo de reduzir custos com impressão e a manipulação (física) de documentos em papel (Mattos *et al.* 2008).

Entretanto, nas últimas décadas, o uso desta tecnologia foi extrapolado para diferentes domínios da ciência, como a biologia e a astronomia inicialmente, que não são comerciais. Áreas que anteriormente executavam seus experimentos manualmente ou por meio de *scripts* foram naturalmente migrando para a utilização de *workflows* científicos como abstração para especificação de seus experimentos (Hey *et al.* 2009). O termo *workflow* científico é usado para descrever *workflows* em algumas destas áreas da ciência, nas quais se compartilham as características de manipulação de grandes volumes de dados, heterogeneidade na representação dos dados e demanda por alto poder de processamento (Taylor *et al.* 2007a).

Por simplificação, nesta tese um *workflow* científico é definido como um grafo acíclico dirigido (do inglês *Directed Acyclic Graph* ou DAG) (Cormen *et al.* 2009) chamado *W(A, Dep)*. Os nós, ou nodos, ($A = \{a_1, a_2, ..., a_n\}$) em *W* correspondem a todas as atividades (invocação de programas) do *workflow* a ser executado (em paralelo ou sequencialmente) e as arestas (*Dep*) estão associadas à dependência de dados entre as atividades de *A*.

Desta forma, dado $a_i \mid (1 \leq i \leq n)$, consideremos como $I = \{i_1, i_2, .., i_m\}$ o conjunto possível de dados de entrada para a atividade a_i, então $Input(a_i) \supset I$. Além disso, consideremos *O* como sendo o conjunto possível de dados de saída produzidos por a_i, então $Output(a_i) \supset O$. Uma atividade específica a_i é modelada como um a_i *(time)*

onde *time* é o tempo total de execução de a_i. A dependência entre duas atividades é modelada como $dep(a_i, a_j, ds) \leftrightarrow \exists O_k \in Input(a_j) \mid O_k \in Output(a_i)$ e *ds* é o volume de dados transferidos entre essas duas atividades (independente da unidade de medida utilizada).

2.4 Proveniência de Dados

O termo "proveniência" pode ser definido de diversas formas. No dicionário Aurélio (Buarque de Holanda 2004) o termo proveniência é definido como "*s.f.* Origem, procedência: mercadorias de proveniência estrangeira; uma pessoa de proveniência ignorada".

De todas as definições existentes, a que nos interessa no contexto desta tese é a de "origem". Registrar a origem e o histórico de uma informação em um experimento científico é fundamental para que os procedimentos executados (no caso o *workflow*) e os dados consumidos e produzidos sejam validados e passíveis de reprodução por parte de terceiros. Desta forma, Buneman *et al.* (2001) foram os primeiros autores a definir a questão de proveniência de dados em experimentos científicos, definindo o termo como a descrição da origem de um dado e o processo pelo qual este chegou a um banco de dados. Goble *et al.* (2003) resumem as diversas funcionalidades para as informações de proveniência da seguinte maneira: (i) garantia de qualidade dos dados: informações de proveniência podem ser utilizadas para estimar a qualidade e a confiabilidade dos dados baseando-se na origem dos dados e suas transformações; (ii) auditoria dos caminhos: os dados de proveniência podem traçar rotas dos dados, determinar a utilização de recursos e detectar erros na geração de dados; (iii) verificação de atribuição: mantém controle sobre as informações do dono do experimento e seus dados. Também permite a citação e atribuem responsabilidades em caso de dados errados e; (iv) informacional: permite realizar consultas baseadas nos descritores de origem para a descoberta de dados, além de prover o contexto necessário para interpretar os mesmos.

De uma maneira geral, a proveniência pode ser caracterizada em duas formas: prospectiva e retrospectiva (Cruz *et al.* 2009, Freire *et al.* 2008b). A proveniência prospectiva está interessada em capturar e armazenar os dados relativos à estrutura do processo que levou à geração de um determinado produto. Ou seja, no contexto de *workflows* científicos, a proveniência prospectiva está preocupada em capturar os dados

relativos à estrutura do *workflow* bem como as configurações de ambiente utilizadas para executá-lo.

Já a proveniência retrospectiva tem o foco em capturar os dados e seus descritores produzidos a partir da execução de um determinado processo, no nosso caso, de um determinado *workflow*. Dados de proveniência retrospectiva englobam tempos de início e fim de execução, arquivos produzidos, erros que ocorreram, informações de desempenho de atividades, entre outros.

Os dados de proveniência podem ser capturados de diferentes formas, mas a principal questão na captura se refere à granularidade dos mesmos. Buneman e Tan (2007) classificam os níveis de detalhe da proveniência capturada como de "grão fino" (do inglês *fine grain*) e "grão grosso" (do inglês *course grain*). A proveniência de "grão grosso" se refere ao registro do histórico de um conjunto de dados enquanto que a proveniência de "grão fino" se refere ao registro da linhagem de um item específico de dados. Nesta tese, abordaremos os dois tipos de proveniência, armazenando informações de arquivos produzidos (conjunto de dados) bem como sobre os dados (itens) extraídos destes arquivos.

Para armazenar a proveniência retrospectiva devem ser utilizados, preferencialmente, modelos de dados que se baseiem na mais recente recomendação do *Open Provenance Model* (OPM) (Moreau *et al.* 2008a), ou de sua evolução: o modelo PROV (Moreau e Missier 2011, Moreau *et al.* 2011), o qual também propõe uma representação genérica de proveniência. A diferença principal entre o OPM e o PROV é a representação. Enquanto que o OPM realiza sua representação por meio de grafos, o PROV optou por representar a proveniência por meio do modelo ER. Além disso, o PROV já possui ontologias associadas a sua representação, o que facilita a interoperabilidade. Tanto o OPM quanto o PROV expressam as relações causais entre Processos, Agentes, Artefatos e Papéis existentes em *workflows*.

Uma das principais vantagens de se utilizar recomendações como estas é garantir a interoperabilidade de descritores de proveniência oriundos de ambientes heterogêneos, independentemente da tecnologia e dos SGWfC utilizados. Por esse motivo, principalmente o OPM já vem sendo utilizado por diversos SGWfC (Davidson e Freire 2008, Freire *et al.* 2008b) como formato para exportação de proveniência e foi foco de

diversos fóruns de discussão (Anderson *et al.* 2007, Moreau *et al.* 2008b, ProvChallenge 2009, 2010).

2.5 Ambientes para Processamento de Alto Desempenho

Conforme dito anteriormente, vários *workflows* científicos necessitam de ambientes de PAD para executarem em tempo hábil. Entretanto, existem diversos tipos de ambiente de PAD que podem ser utilizados para esta execução, cada qual com vantagens e desvantagens associadas. Nesta seção discutimos os principais ambientes de PAD existentes.

Os *clusters* de computadores, também chamados de aglomerados de computadores, podem ser definidos como um conjunto de máquinas que, por meio de mensagens de comunicação, conseguem trabalhar como se fossem uma única máquina com alto poder de processamento e armazenamento. Uma característica importante dos *clusters* é a homogeneidade de sua estrutura e a utilização de redes de alto desempenho com baixa latência (*i.e. infiniband*). As execuções de um *workflow* em ambientes de *cluster* requerem a utilização de um escalonador (Bayucan *et al.* 2000, Staples 2006, Thain *et al.* 2002), para que se consiga tirar alguma vantagem da estabilidade e das características de homogeneidade dos *clusters*.

As grades de computadores podem ser definidas como uma estrutura de malha que combina recursos distribuídos de *software* e *hardware* que são (em sua maioria) heterogêneos e fracamente acoplados. A utilização de um ambiente de grade normalmente depende de uma camada de *software* ou de um intermediário para gerenciar as execuções distribuídas. Um exemplo de intermediário é o Globus *Toolkit* (Foster e Kesselman 1996). A execução de *workflows* científicos em necessita também de um escalonador: porém, diferentemente dos *clusters*, esse escalonador necessita se preocupar com latência na transferência de dados, autenticações e questões de segurança (Thain *et al.* 2002, Yu e Buyya 2006, Yu *et al.* 2005).

Os ambientes de computação voluntária se baseiam na utilização de máquinas geograficamente dispersas, porém não dedicadas a uma determinada tarefa. A ideia principal é que exista um servidor central que controle a execução de pequenas tarefas em máquinas de terceiros, onde o tempo inativo das máquinas é disponibilizado para execuções distribuídas. Em termos de execuções de *workflows* nestes ambientes, podemos gerar facilmente milhões de tarefas uma vez que a quantidade de recursos

disponível é consideravelmente maior do que em *clusters* e grades. O servidor central envia então, aos voluntários, pequenas tarefas de forma que os *workflows* possam ser processados de forma mais rápida do que seriam em supercomputadores (em teoria).

Mais recentemente o conceito de nuvem de computadores surgiu como um ambiente de PAD promissor, fato este que motivou o desenvolvimento desta tese. Estes ambientes são caracterizados pela diversidade dos recursos e pelo acesso através de uma camada de virtualização (máquinas virtuais). A grande vantagem das nuvens é que o usuário (no contexto desta tese, o cientista) tem um ambiente de alta capacidade de processamento, onde ele tem o controle absoluto das ações, elasticidade dos recursos e, além disso, só necessita pagar pelo que é efetivamente usado, diferentemente dos *clusters* e grades onde é necessário um investimento inicial grande, seja monetário ou de configuração e manutenção. A Seção 2.6 descreve com mais detalhes os conceitos de nuvens de computadores.

Cada ambiente citado nesta seção pode ser implementado de diversas maneiras e seguindo diferentes arquiteturas para organização dos computadores e acesso de dados. No contexto desta tese, a arquitetura das máquinas não produz impactos severos, porém a organização do acesso aos dados sim. As arquiteturas para acesso aos dados podem ser divididas em arquiteturas de discos compartilhados (do inglês *shared disk*) e arquiteturas de discos não compartilhados (do inglês *shared-nothing*) (Bouganim *et al.* 1996, Dantas 2005a, 2005b). As arquiteturas de discos não compartilhados dependem exclusivamente dos discos locais em cada máquina. Enquanto que nas arquiteturas de discos compartilhados todas as máquinas tem acesso aos discos, deixando a distribuição da execução mais flexível. Na Figura 4 apresentamos um exemplo de arquitetura de disco não compartilhado enquanto que na Figura 5 tem-se um exemplo de uma arquitetura acessando discos compartilhados. Como veremos no decorrer da tese, optamos por utilizar uma arquitetura de discos compartilhados na abordagem proposta.

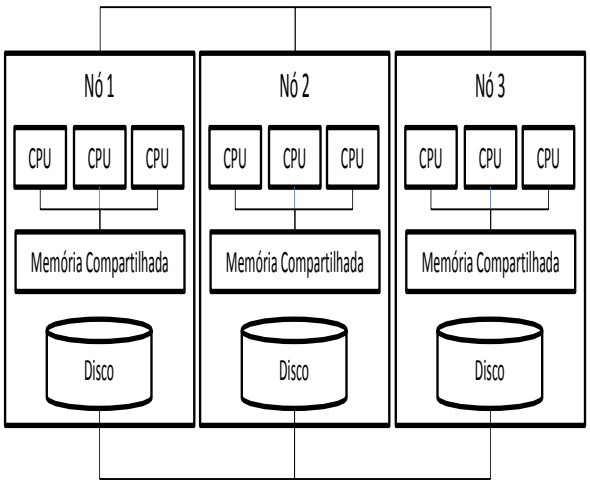


Figura 4 Arquitetura de disco não compartilhado

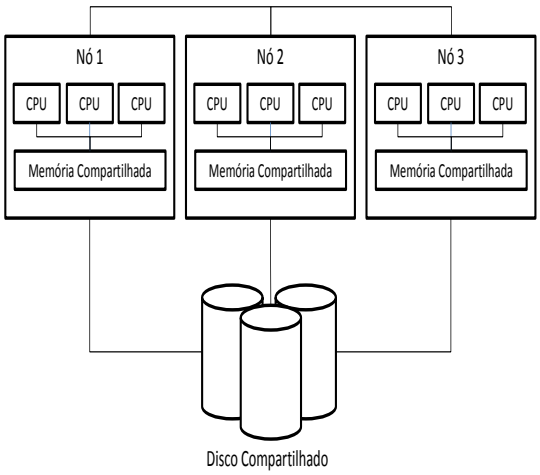


Figura 5 Arquitetura de disco compartilhado

2.6 Nuvens de Computadores

A computação em nuvem (do inglês *Cloud Computing*) (Kim *et al.* 2009, Marinos e Briscoe 2009, Napper e Bientinesi 2009, Vaquero *et al.* 2009, Wang *et al.* 2008) surgiu como um novo paradigma de computação distribuída, onde serviços baseados na *Web* têm como objetivo permitir a diferentes tipos de usuários obterem uma grande variedade de recursos de *software* e *hardware*. Desta forma, a computação, como costumávamos conhecer há alguns anos, mudou completamente. Os programas e os dados passaram dos *desktops* para a nuvem. Os usuários são capazes de acessar programas, documentos e dados de qualquer computador que esteja conectado na internet. Conceitos de elasticidade (Chen *et al.* 2008), disponibilidade (Yu e Vahdat 2006) e segurança (Leusse *et al.* 2008) se tornaram questões chave a serem tratadas.

A computação em nuvem apresenta um grande potencial para apoiar experimentos científicos que necessitem de ambientes de PAD. Na verdade, as nuvens já introduziram uma série de benefícios (Gorder 2008, Sullivan 2009) para este tipo de experimento. A natureza das necessidades da computação científica se encaixa bem com a flexibilidade e a elasticidade, sob demanda, oferecida pelas nuvens. De acordo com Simmhan *et al.* (2010) o uso efetivo de nuvens pode reduzir o custo real com equipamentos e sua consequente manutenção, além da questão das atualizações constantes de *software* e *hardware*.

Usuários de centros de supercomputação podem utilizar as nuvens para reduzir o tempo de espera em filas de escalonadores (Armbrust *et al.* 2010). As nuvens permitem

utilizar mais máquinas virtuais durante um horário de pico nos centros de dados, ou durante eventos especiais, como experimentos que requerem uma grande quantidade de recursos reservados durante uma janela de tempo maior. Além disso, grandes conjuntos de dados utilizados e gerados pelo experimento podem ser mais facilmente compartilhados para a análise, dentro ou fora da nuvem, democratizando o acesso aos resultados científicos, embora a transferência de dados em nuvens seja um problema em aberto.

Desta forma, as nuvens podem proporcionar um ambiente adequado para a execução paralela de *workflows* científicos que demandem PAD. Entretanto, muitos problemas continuam sem solução como, por exemplo, os gargalos de transferência de dados, a flutuação de desempenho das máquinas virtuais, a falta de interoperabilidade dos ambientes, etc. Desta forma, novas propostas são necessárias para fornecer uma solução transparente para os cientistas executarem seus *workflows* em paralelo em nuvens de forma sistemática. A seguir, de forma a explicar melhor o cenário atual da computação em nuvem, apresentamos uma taxonomia desenvolvida (Oliveira *et al.* 2010a) que detalha as principais características das nuvens, além dos padrões utilizados neste ambiente.

2.6.1 Taxonomia para Computação em Nuvem em e-Science

Mesmo existindo uma enorme quantidade de artigos (científicos e comerciais) apresentando soluções de nuvem, os conceitos envolvidos com a computação em nuvem não são totalmente detalhados ou explicados. Considerando o crescente interesse neste assunto e a dificuldade em encontrar definições organizadas associadas a este paradigma, apresentamos nesta seção uma taxonomia para o campo de computação em nuvem a partir de uma perspectiva de e-Science (Oliveira *et al.* 2010a).

As taxonomias (Fuller *et al.* 2007) são uma estrutura de classificação especial onde os conceitos são organizados de forma hierárquica. A taxonomia de nuvem proposta fornece uma compreensão do domínio e visa ajudar os cientistas a entenderem este ambiente. A taxonomia considera uma visão de *e-Science* da computação em nuvem, apresentando algumas das suas principais facetas. Utilizando a taxonomia proposta (Figura 6) como um vocabulário comum pode-se facilitar o trabalho dos cientistas para encontrar dentre os ambientes existentes, aquele que mais lhe seja adequado.

A taxonomia proposta classifica as características do domínio de computação em nuvem baseado em diferentes aspectos: características arquiteturais, de modelo de negócio, de infraestrutura tecnológica, de privacidade, de normas, de precificação, de orientação e de acesso.

2.6.1.1 Modelo de Negócio

No aspecto Modelo de Negócio, de acordo com o modelo adotado, as abordagens de nuvem são geralmente classificadas em três categorias principais (NIST 2010): Software como Serviço (do inglês *Software as a Service* ou SaaS), Plataforma como Serviço (do inglês *Platform as a Service* ou PaaS) e Infraestrutura como Serviço (do inglês *Infrastructure as a Service* ou IaaS), criando um modelo chamado SPI (do inglês *Service-Platform Infrastructure*) (Youseff *et al.* 2008, Zhu e Wang 2008).

Em SaaS o *software* é disponibilizado através da *Web* para o uso comercial ou livre como um serviço sob demanda. Em IaaS o provedor oferece uma infraestrutura computacional (como um *cluster*, por exemplo) para o usuário final através da *Web*. Em IaaS, o usuário final normalmente é responsável por configurar o ambiente para usar. Definimos PaaS como a entrega de uma plataforma de programação como um serviço. O processo de entrega de uma plataforma como serviços facilita a implantação de programas na nuvem.

2.6.1.2 Privacidade

De acordo com o aspecto de privacidade, podemos classificar em ambientes de nuvem em três categorias: privados, públicos e mistos (também chamados de híbridos, porém optamos por denominar mistos para não confundir com a arquitetura híbrida de nuvem (Zhang *et al.* 2009)). Nuvens públicas podem ser consideradas o mais tradicional de todos os tipos de nuvem no que diz respeito à privacidade. Neste tipo de nuvem os vários recursos são dinamicamente disponibilizados por meio da Internet, por meio de aplicações *Web* ou serviços *Web* (Alonso *et al.* 2010), para qualquer usuário. Nuvens privadas são ambientes que emulam a computação em nuvem em redes privadas, dentro de uma corporação ou uma instituição científica. Já um ambiente de nuvem mista é aquele que é composto por múltiplas nuvens, sejam elas públicas ou nuvens privadas. O conceito de nuvem mista ainda é um pouco nebuloso, pois não é definida a fronteira

entre os dois ambientes. Este aspecto das nuvens é importante para os experimentos científicos, dada a importância dos níveis de privacidade na pesquisa.

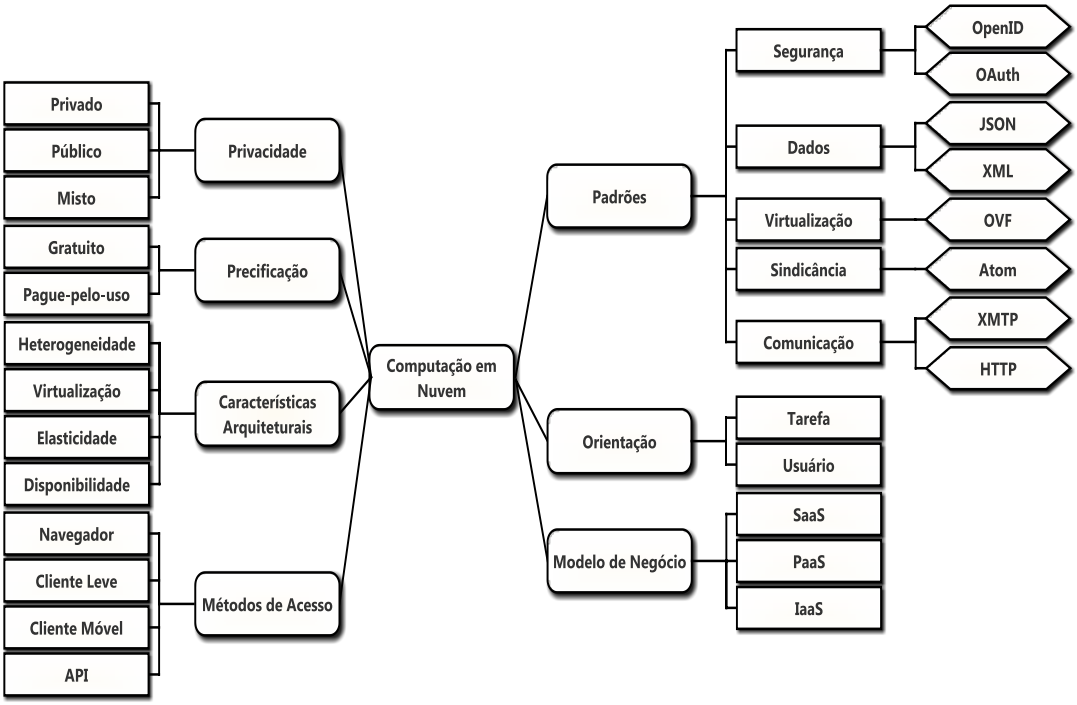


Figura 6 Taxonomia para o domínio da computação em nuvem adaptada de Oliveira *et al.* (2010a)

2.6.1.3 Precificação

Uma vez que as nuvens inserem um novo desafio para os cientistas que é lidar com os custos de um experimento (de modo que não se torne proibitivo), devemos classificar as abordagens de nuvem de acordo com o aspecto de precificação. A precificação em nuvens pode ser classificada em duas categorias: gratuita e pague-pelo-uso. A primeira categoria é a mais simples: a gratuita. A categoria gratuita é a aplicada quando o cientista está usando o seu ambiente próprio de nuvem, onde os recursos estão disponíveis gratuitamente para usuários autorizados. A categoria pague-pelo-uso (do inglês *pay-per-use*) é aquela onde o cientista paga um valor específico relacionado à sua utilização de recursos, seja ele grande ou pequeno. A categoria pague-pelo-uso pode ser aplicada em ambientes de nuvem comerciais e científicos. Os cientistas pagam pelo uso da nuvem da mesma forma que os usuários comerciais o fazem, como por exemplo, ocorre no ambiente de nuvem Nimbus (Wang *et al*. 2008).

2.6.1.4 Características Arquiteturais

Este aspecto classifica os ambientes de nuvem de acordo com a sua aderência (ou não) a um conjunto de características arquiteturais. Desta forma, de acordo com o aspecto de características arquiteturais, podemos elencar uma série de categorias de características desejáveis em um ambiente. Uma característica fundamental de uma nuvem é a heterogeneidade. Uma nuvem deve apoiar a agregação de *hardware* heterogêneo assim como vários tipos de recursos de *software*, assim como acontece com *workflows* científicos. A virtualização (Asosheh e Danesh 2008) também é uma característica fundamental para as nuvens. Por meio da virtualização, muitos usuários podem se beneficiar da mesma infraestrutura física usando instâncias independentes. A virtualização também aumenta a segurança (Jensen *et al.* 2009), uma vez que permite o isolamento de ambientes. Nas nuvens, cada usuário tem acesso exclusivo ao seu ambiente virtualizado. Outra característica arquitetural fundamental é a elasticidade. Aliada a virtualização, esta característica é a base para as propostas apresentadas nesta tese. O conceito de elasticidade é definido como o aumento (ou diminuição), sob demanda (e teoricamente sem limites), do número de máquinas virtuais em um ambiente de nuvem. Cada uma dessas características de arquitetura é padronizada por normas específicas.

2.6.1.5 Métodos de Acesso

Podemos classificar os ambientes de nuvem de acordo com o aspecto de métodos de acesso. Na maioria dos casos, podemos encontrar quatro tipos de categorias: Navegadores, Clientes Leves, Clientes Móveis e APIs (Pressman e Roger 2009). Os navegadores são a forma mais comum de acesso para serviços de nuvem. Muitas aplicações são acessíveis somente em navegadores *Web*. É intuitivo, já que quase todos os computadores tem pelo menos um *browser* instalado. Clientes leves e móveis são os tipos de acesso às nuvens de um *desktop* "burro" ou de *tablets* e telefones celulares. Tornou-se popular o acesso aos serviços através de telefones em vez de *desktops*. E, finalmente, a API é uma maneira fundamental para o acesso nuvens. A API é um artefato de acesso através de linguagens de programação como Java (Sun 2010) ou Python (Python 2010). Usando uma API, aplicações mais complexas podem usar a infraestrutura de nuvem em uma forma nativa. Uma vez que os experimentos científicos modelados como *workflows* científicos são especificados nos SGWfC, uma necessidade

importante é conectar os SWfMS às nuvens usando uma API, porque uma API pode ser facilmente invocada por componentes programáveis. Esta, aliás, foi a abordagem utilizada nesta tese e melhor explicada no Capítulo 5.

2.6.1.6 Padrões

Este aspecto classifica os ambientes de nuvem de acordo com os padrões ao qual ele adere. Dessa forma, é dividido em categorias de padrões encontrados na literatura para nuvens. Todos os padrões aqui apresentados são normatizados pelo W3C (do inglês *World Wide Web Consortium*). O protocolo de mensagens e presença (do inglês *Extensible* XMPP) (XMPP 2010) é uma tecnologia aberta de comunicação em tempo real, que é utilizada por uma ampla gama de aplicações. O Protocolo de Transferência de Hipertexto (do inglês *Hyper Text Transfer Protocol* ou HTTP) (HTTP 2010) é o padrão mais conhecido de comunicação e é intuitivo de usá-lo na nuvem, uma vez que é usado em aplicações *Web* básicas. O OAuth (OAuth 2010) é um protocolo de segurança para publicar e interagir com os dados protegidos. Além disso, é um protocolo aberto para permitir a autorização de utilização de uma API de uma forma simples. Por outro lado, o OpenID (OpenID 2010) é um padrão aberto e descentralizado para autenticação de usuários e controle de acesso, permitindo aos usuários registrar-se em muitos serviços com a mesma identidade digital, assim como já foi adotado em grades computacionais. Além disso, podemos encontrar o Atom (Atom 2010) que é um protocolo de licenciamento de conteúdo com base em HTTP para criação e atualização de recursos da *Web*.

A virtualização é um aspecto fundamental da computação em nuvem e necessita também de padrões. O OVF (OVF 2010) está sendo considerado um dos padrões *de facto* para a virtualização. O OVF permite a distribuição flexível e segura de *software* e dados, facilitando a mobilidade de máquinas virtuais. Como acontece em muitos sistemas *Web*, os dados são geralmente representados e transferidas utilizando XML (McLaughlin 2001) e JSON (JSON 2010).

2.6.1.7 Orientação

Um aspecto importante da computação em nuvem para experimentos científicos é a orientação, que pode ser centrada em tarefa ou centrada no usuário. A orientação de um ambiente de nuvem muda conforme o tipo de serviço disponibilizado. Por exemplo,

quando um *software* é disponibilizado na nuvem como um serviço, podemos considerá-lo como centrado em tarefa, porque é orientado para a tarefa que será executada. Em outras palavras, o cientista precisa transferir o controle para os proprietários do *software* em vez de ter o controle do mesmo. No entanto, quando a infraestrutura é fornecida como um serviço, o cientista tem o controle do processo. Os programas, aplicativos e dados são escolhidos pelo mesmo. Assim, a abordagem pode ser considerada como centrada no usuário. As abordagens propostas nesta tese são centradas no usuário, ou no nosso caso, o cientista, que deve ter controle do curso do experimento.

2.7 Ciclo de Vida do *Workflow* Científico em Nuvem

Um *workflow* científico que é executado em um ambiente de nuvens de computadores segue as mesmas etapas do ciclo de vida de um experimento, porém com algumas fases adicionais. Podemos definir o ciclo de vida de um *workflow* científico executado em nuvens como sendo composto de sete etapas principais como mostrado na Figura 7.

O ciclo de vida do *workflow* executado em nuvem descreve as fases pelas quais o *workflow* passa desde a sua especificação até a publicação dos resultados. Além das fases de composição, execução e análise já presentes no ciclo de vida do experimento, o ciclo de vida do *workflow* executado em nuvens engloba uma fase de configuração que não existe no ciclo de vida original. Como os *workflows* dependem do provimento de infraestrutura por parte do provedor de nuvem, este ambiente ainda não está completamente configurado para a execução do *workflow*. Adicionalmente, ainda existem questões como a transferência dos dados e o posterior *download* dos mesmos para fins de análise. Devemos lembrar que todas essas ações estão associadas com os dados de proveniência, ou seja, ou produzem dados ou os consomem de alguma maneira (ou ambos).

Na fase de composição do *workflow* os cientistas elaboram as especificações, informando quais programas deverão ser utilizados e qual a dependência de dados entre eles e em qual área da nuvem os mesmos se encontram. Esta especificação impacta diretamente na configuração do ambiente como veremos a seguir. Na fase de configuração do ambiente é realizada a transferência de dados da máquina local para a nuvem, a criação das imagens com os programas que fazem parte do *workflow* e a criação do *cluster* virtual, onde o *workflow* será executado em paralelo. Esta fase possui um subciclo, pois a configuração do *cluster* virtual é uma tarefa que não termina

enquanto o *workflow* não finaliza sua execução. Isso porque o tamanho do *cluster* pode aumentar ou diminuir dependendo da demanda de capacidade de processamento do *workflow*.

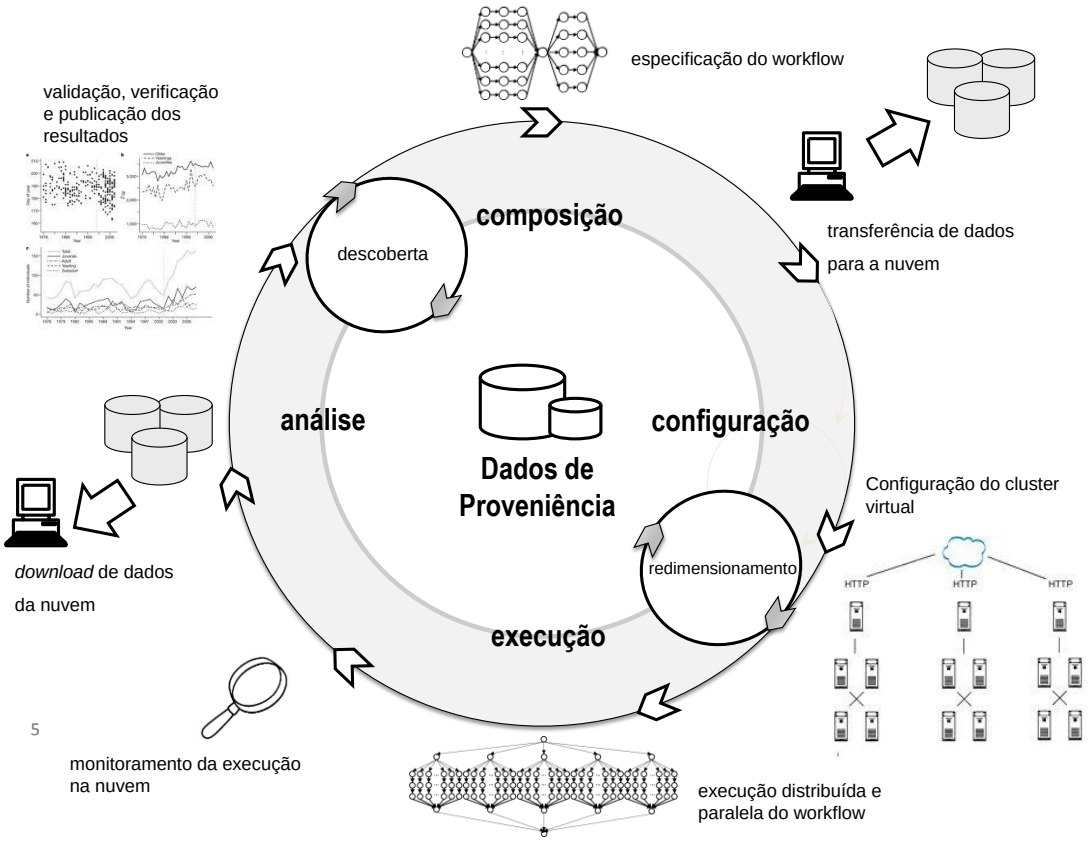


Figura 7 Ciclo de vida de um *workflow* científico executado em nuvens de computadores

Na fase de execução, as tarefas do *workflow* são de fato despachadas e executadas nas diversas máquinas virtuais que foram instanciadas na fase de configuração. Nesta fase são realizados os escalonamentos de forma a aproveitar as vantagens dos ambientes de nuvem. Após o despacho do *workflow* para execução ocorre a fase de monitoramento do *workflow* para analisar a ocorrência de erros, quais atividades já terminaram etc.

Na fase de análise os dados são efetivamente baixados para a máquina do cientista para enfim serem analisados. Esta fase inclui uma subfase de descoberta, onde os cientistas desenharão conclusões baseados nos dados e verificarão se a hipótese de pesquisa original foi corroborada ou refutada. No caso de ter sido refutada, o ciclo se repete novamente, mudando-se alguns parâmetros de entrada. Retomaremos essa definição de ciclo de vida ao longo da tese, apontando em que etapa do ciclo as abordagens propostas se encaixam.

2.8 O Conceito de *Cloud Activity*

Dado que o foco desta tese é na gerência da execução paralela de um *workflow* científico em nuvens, uma definição importante é a do menor grão que será paralelizado. Dado que queremos executar uma atividade em paralelo na nuvem, conforme já mencionado, uma abordagem de paralelismo que podemos adotar e que não impacta em reescrever o código do programa invocado é fazer com que cada valor de parâmetro combinado com um dado a ser consumido seja executado em paralelo em uma máquina virtual diferente. Como o ambiente de nuvem é distribuído e cada máquina virtual é independente da outra, devemos encapsular em uma única unidade de trabalho o programa a ser executada, os dados, os valores de parâmetros e a estratégia de paralelismo que será utilizada.

A esta unidade de trabalho damos o nome de *Cloud Activity*[2]. Nesta tese, uma *cloud activity* é a menor unidade de trabalho que pode ser processada em paralelo. O conceito de *cloud activity* se assemelha em partes ao conceito de ativação de atividade definido por Ogasawara *et al.* (2011). A *cloud activity* (Oliveira *et al.* 2010b, 2010c, 2011c, 2011a) contém o código executável para ser invocado, a estratégia de paralelismo definida pelos cientistas (*i.e.*, quais os parâmetros variar), os valores dos parâmetros (o conjunto de valores de parâmetros) e dados a serem consumidos. As *cloud activities* não são sinônimo de atividades do *workflow*, uma vez que as *cloud activities* podem encapsular uma ou mais atividades e suas dependências.

Por exemplo, consideremos três atividades a_1, a_2 e a_3 definidas em um *workflow* científico e que estão associados às aplicações app_1, app_2 e app_3, respectivamente. Essas três atividades podem ser encapsulados em uma única *cloud activity* ca_1. Isto significa que esta *cloud activity* ca_1 invoca e executa no ambiente de nuvem cada um dos programas associados às atividades a_1, a_2 e a_3. Em outras palavras, uma *cloud activity* pode estar associada a uma única atividade do *workflow* (como apresentado na Figura 8) ou a um conjunto de atividades (*i.e.* sub-*workflow*, como apresentado na Figura 9).

[2] Optamos por não traduzir o termo *Cloud Activity* para gerar uma identificação com os artigos já publicados pelo autor desta tese.

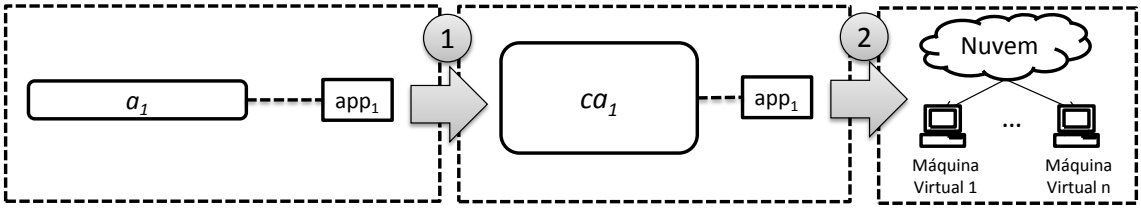


Figura 8 Mapeamento de uma atividade do *workflow* para uma *cloud activity*

Podemos definir formalmente uma *cloud activity* como se segue. Dado um *workflow W*, um conjunto $Ca = \{ca_1, ca_2, ..., ca_k\}$ de *cloud activities* é criado para a execução paralela. Cada *cloud activity* ca_i está associada a uma atividade a_i específica (ou a um conjunto de atividades) que é representada como $Act(ca_i) = a_i$. O conjunto de *cloud activities* associadas a uma determinada atividade a_i do *workflow W* é denotada como $Ca(a_i)$.

Cada ca_i consome a sua própria entrada de dados $InputCa(ca_i)$ e produz a saída de dados $OutputCa(ca_i)$. Assim, podemos estabelecer a dependência entre duas *cloud activities* como $DepCa(ca_i, ca_j, ds) \leftrightarrow \exists\ r \in Input(ca_j) \mid r \in Output(ca_i) \wedge dep(Act(ca_i),Act(ca_j),ds)$. A *cloud activity* específica *ca* é modelada como *ca(time)* onde a variável *time* é o tempo de execução total de *ca*.

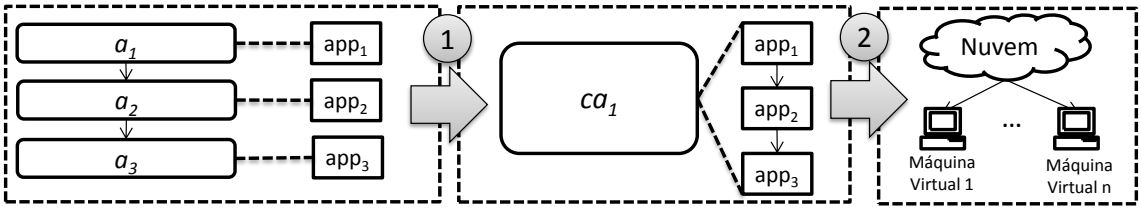


Figura 9 Mapeamento de um sub-*workflow* em uma *cloud activity*

As *cloud activities* são regidas por uma Máquina de Estados Finita específica (do inglês *Finite State Machine* ou FSM), apresentada na Figura 10. A FSM proposta possui cinco principais estados: Criado, Pronto, Ativado, Bloqueado e Terminado. Quando uma *cloud activity* é criada, ela recebe o estado de "Criado". Automaticamente, se não houver erro, recebe um estado "Pronto". Caso contrário, se, por alguma razão, uma *cloud activity* não pode imediatamente iniciar a execução, ele recebe um estado "Bloqueado". Um exemplo é quando uma *cloud activity* tem uma dependência de dados com outra *cloud activity* que não tenha sido executada ainda. A *cloud activity* pode continuar para o estado "Ativado" se ela pode ser executada normalmente. Uma vez que

termina a execução de uma *cloud activity*, ela recebe o estado "Terminado". Quando a *cloud activity* tem um problema como um programa corrompido ou dados corrompidos ou inexistentes, e não pode ser executada, então vai diretamente do estado "Ativado" para o estado "Terminado".

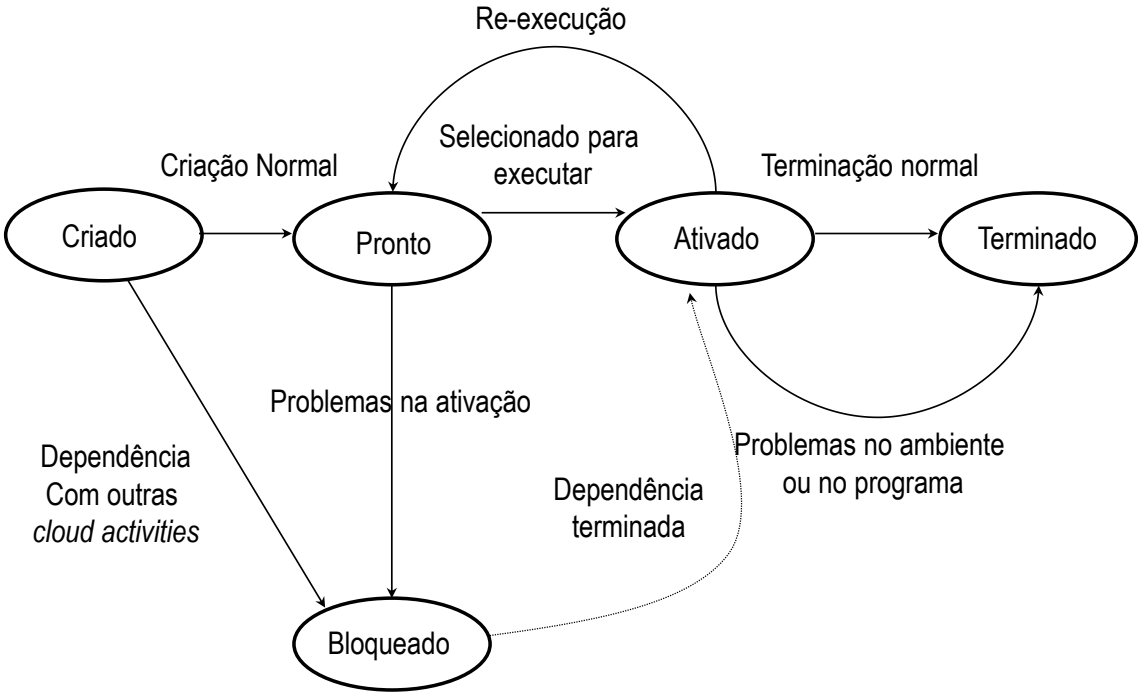


Figura 10 Máquina de estados finita que rege as *cloud activities* (Oliveira *et al.* 2011c)

2.9 Paralelismo por Varredura de Parâmetros

O paralelismo por varredura de parâmetros, que é apresentado na avaliação experimental desta tese, é definido como execuções simultâneas de várias *cloud activities*, onde cada ca_i consome um subconjunto específico chamado Pv_i que é a combinação do conjunto de parâmetros de domínio *Pt* de uma atividade a_i a ser processada. Este tipo de parâmetro pode ser alcançado por meio da geração de diversas *cloud activities* de uma atividade específica e cada um *cai* processa um subconjunto específico do possível conjunto de valores de parâmetros Pv_i.

Para tornar mais claro, suponhamos que duas *cloud activities* ca_i e ca_j têm parâmetros pt_1 e pt_2. Suponha também que o domínio de valores possíveis para os parâmetros pt_1 e pt_2 são $Dpt_1 = \{x, x'\}$ e $Dpt_2 = \{y, y'\}$. Vamos considerar que ca_i consome valores *x* e *y* para os parâmetros pt_1 e pt_2 respectivamente, e ca_j consome valores *x'* e *y'*. Desta forma, podemos definir as combinações $Pv_1 = \{x,y\}$ e $Pv_2 = \{x', y'\}$. Consequentemente, o paralelismo por varredura parâmetros pode ser obtido através

do processamento Pv_1 e Pv_2 em paralelo na nuvem. Os diversos Pv_i são formados tomando os valores dos parâmetros do domínio em uma determinada ordem. Desta forma, Pv_1 é formado por $Dpt_1[1]$ e $Dpt_2[1]$, Pv_2 é formado por $Dpt_1[2]$ e $Dpt_2[2]$, e assim sucessivamente. O conjunto de dados de saída O_k gerado pela k-ésima execução precisa ser agregado, uma vez que ele se refere a execuções diferentes do mesmo *workflow*. Neste tipo de paralelismo, *I* é o mesmo conjunto para cada ca_i.

Existe ainda o caso em que *I* pode ser fragmentado e executado com diversas combinações de valores de parâmetros. O paralelismo de dados em *workflows* científicos caracteriza-se pela execução simultânea de várias *cloud activities* de uma atividade a_i d o *workflow W*, onde cada ca_i processa um subconjunto disjunto do conjunto completo de dados de entrada, que pode ser representado por um conjunto de fragmentos $F = \{fi_1, fi_2, ..., fi_k\}$. O conjunto de dados de saída fragmentados fo_k gerado a cada k-ésima execução tem que ser fundidos (seguindo um critério específico de fusão), a fim de produzir o resultado final (*Oi*). Na prática, este tipo de paralelismo só difere da varredura de parâmetros por necessitar fundir os dados ao fim da execução.

Capítulo 3 - Um Modelo de Custo Multiobjetivo para Execução de *Workflows* Científicos em Nuvens Computacionais

De acordo com o que foi apresentado na introdução desta tese, as abordagens existentes para execução paralela de *workflows* científicos que podem ser adotadas em ambientes de nuvem não são aderentes a todas as características inerentes a estes ambientes. Ou seja, o escalonamento das *cloud activities* (a escolha de qual *cloud activity* será executada em qual máquina virtual) não considera características importantes quando estamos executando *workflows* em nuvens. Assim, abordagens adaptativas se fazem necessárias. De modo a tratar este problema a abordagem proposta nessa tese é composta por um modelo de custo que distribui as *cloud activities* de acordo com a capacidade das máquinas virtuais e um algoritmo de escalonamento guloso que escalona as mesmas adaptativamente. Tanto o modelo de custo quanto o algoritmo de escalonamento se enquadram na fase de execução do ciclo de vida do *workflow* executado em nuvem.

Este capítulo apresenta o modelo de custo para execução paralela de *workflows* em nuvens, inspirado em décadas do já consolidado conhecimento sobre modelos de custo em bancos de dados (Elmasri e Navathe 2006, Özsu e Valduriez 2011) e sobre modelos de custo em ambientes distribuídos. Na Seção 3.1 é apresentada a definição de modelo de custo no contexto desta tese. Na Seção 3.2 é introduzido o formalismo do modelo no que se refere ao ambiente de nuvem. Na Seção 3.3 é apresentada a modelagem do tempo de execução dos *workflows* executados em paralelo. Na seção 3.4 é apresentada a modelagem da confiabilidade do sistema. Na Seção 3.5 é apresentada a modelagem do custo de transferência. Na Seção 3.6 é apresentada a modelagem do custo monetário enquanto que na Seção 3.7 apresentamos o modelo de custo completo utilizado nesta tese. Finalmente, na Seção 3.8 discutimos sobre a normalização dos valores das variáveis do modelo.

3.1 Definição de Modelo de Custo

Um modelo de custo consiste em uma série de fórmulas que tem como objetivo estimar o custo envolvido na execução de uma tarefa em um ambiente computacional específico

(Elmasri e Navathe 2006, Özsu e Valduriez 2011). Um modelo de custo é comumente implementado em um sistema que executa diversas simulações de custo ou utiliza heurísticas de forma a escolher uma solução ótima (ou em muitos casos a mais adequada, quando a ótima não pode ser obtida em tempo viável).

No contexto desta tese, a tarefa cujo custo será estimado é uma *cloud activity*, o ambiente computacional de execução da tarefa é o ambiente de nuvem, e o sistema em que o modelo de custo está implementado é o SciCumulus (que é detalhado no capítulo 5) que utiliza o modelo de custo proposto para realizar o escalonamento e o despacho de diversas *cloud activities* em paralelo em um conjunto de máquinas virtuais que juntas formam um *cluster* virtual na nuvem. O modelo de custo aqui apresentado foi inspirado nas ideias propostas por Boeres *et al.* (2011) em seu estudo sobre escalonamento em *clusters* de computadores. O modelo de custo proposto leva em consideração três diferentes dimensões simultaneamente (multiobjetivo), atribuindo pesos a cada um dos critérios em sua estimativa:

i. O tempo de execução do *workflow*;
ii. A confiabilidade das máquinas virtuais que compõem o ambiente e;
iii. O custo monetário total.

A seguir neste capítulo detalhamos cada uma dessas dimensões, apresentando seus elementos e como os utilizamos para estimar o custo e realizar o escalonamento.

3.2 Formalização do Ambiente de Nuvem

De modo a formalizar o ambiente de nuvem, consideremos $VM = \{VM_0, VM_1,..,VM_{m-1}\}$ como sendo o conjunto de *m* máquinas virtuais que compõem um *cluster* virtual e se encontram disponíveis para uso geral na execução do *workflow*. Consideremos também que cada máquina virtual possui um valor associado a ela chamado de índice de retardo de computação (do inglês *computational slowdown index – csi*, sigla que usaremos a partir de agora). Esta métrica é calculada de forma que seja inversamente proporcional ao poder computacional da máquina virtual em questão. Existem duas maneiras de obtermos o valor de *csi*:

i. Realizar uma estimativa extra modelo: um valor é atribuído para cada máquina virtual sem que se tenha realizado nenhum experimento nas mesmas (valor putativo) e;

ii. Realizar uma estimativa baseada em dados de proveniência de *workflow* já executados: um valor médio de *csi* é calculado baseado nos tempos execuções passadas de experimentos que estão registrados na base de dados de proveniência (o esquema desta base de proveniência é detalhado no capítulo 5 desta tese). A ideia é que, baseado nos tempos de execução anteriores das *cloud activities*, identifiquemos a máquina virtual com melhor desempenho e a esta atribuímos o valor 1 para *csi* (máximo valor possível). As outras máquinas virtuais que fazem parte do *cluster* virtual tem seu valor *csi* relativo ao valor da máquina com melhor desempenho (*i.e.* 0,81).

Uma vez definido o *csi* de cada máquina virtual envolvida na execução, podemos modelar uma máquina virtual como a tupla *vm*(*network, csi, memory, maxGranFactor, partGranFactor, avgTime, prevExecTime*), onde o valor de *network* refere-se à largura de banda que a máquina virtual tem acesso para transferência de dados, o valor de *csi* é o índice de retardo computacional da máquina virtual e o valor de *memory* é a capacidade de memória RAM de cada máquina virtual. Os valores *maxGranFactor, partGranFactor, avgTime,* e *prevExecTime* são informações relativas a tempos médios de execuções passadas na máquina virtual e que são utilizados e explicados no Capítulo 4 desta tese, mas que se referem a parâmetros de agrupamento de *cloud activities* para diminuição da sobrecarga de comunicação e transferência. O índice de retardo computacional de uma máquina virtual específica pode ser obtido também através da função $csi(VM_j)$.

3.3 Tempo de Execução

Consideremos $P(ca_i,VM_j)$ como o tempo de execução previsto de uma *cloud activity* ca_i na VM_j no *cluster* virtual, assim sendo $P(ca_i,VM_j) = ca_i.time \times csi(VM_j)$, onde $ca_i.time$ é o tempo médio desnormalizado (média do tempo de execuções passadas dividido pelo valor de *csi* das máquinas em que as *cloud activities* foram executadas) de execução da *cloud activity*.

Consideremos também $\varphi(W,VM)$ como o escalonamento de todas as *cloud activities* de um conjunto *CA* do *workflow W* em *VM*. Formalmente, dado um *workflow W* que inclui uma série de atividades $A = \{a_1, a_2 \ldots, a_n\}$ e um conjunto de *cloud activities* $CA=\{ca_1, \ldots, ca_k\}$ criadas para execução paralela, seja $\varphi(W,VM) = \{sched_1, sched_2, \ldots, sched_k\}$ o conjunto dos escalonamentos para CA. Consideremos também que

o escalonamento de uma *cloud activity* é definido como *sched*(ca_i, VM_j, *start*, *end*), onde *start* e *end* são os tempos de início e fim da execução de uma *cloud activity* ca_i na VM_j.

Uma vez que o ambiente de nuvem é um ambiente que sofre constantes mudanças (máquinas virtuais são criadas e destruídas durante a execução do *workflow* ou são mudadas de máquina física sem que o usuário esteja a par desta mudança), não podemos estabelecer um plano de escalonamento *a priori*. Desta forma, os diversos *sched* devem ser gerados durante o curso de execução do *workflow*. Complementando, definimos $ord(sched_i)$ como a posição do $sched_i$ na sequência de todos os escalonamentos realizados, e por consequência $sched_i < sched_j \leftrightarrow ord(sched_i) < ord(sched_j)$.

Uma vez que temos como objetivo criar *clusters* virtuais para executar os *workflows* em paralelo, consideremos como $|VM|$ o número de máquinas virtuais que compõem o *cluster* virtual e $|CA|$ como o número de *cloud activities* que devem ser processadas. Desta forma, podemos calcular o limite superior da taxa de processamento (*i.e. throughput*) das *cloud activities* neste *cluster* virtual como:

$$\beta = \frac{|\mathbf{VM}|\times|\mathbf{CA}|}{\sum_{\mathbf{i=0}}^{\mathbf{k}} \mathbf{P}(\boldsymbol{ca_i}, \boldsymbol{VM_j})} \quad (1)$$

Uma vez que a Equação 1 representa a taxa de processamento das *cloud activities* no *cluster* virtual, podemos definir σ como o número de *cloud activities* que se encontram em uma fila para serem processadas, *i.e.* as que aguardam escalonamento. Desta forma, podemos definir o *makespan* (tempo total de execução do *workflow*) de todas as *k cloud activities* em um *workflow* científico como:

$$MS = \frac{\sigma}{\beta} = \frac{\sum_{i=0}^{k} P(cai_i, VM_j)\sigma}{|VM|\times|CA|} \quad (2)$$

Antes de iniciar o escalonamento do *workflow* na nuvem utilizando este modelo de custo, necessitamos estimar o valor de $P(ca_i, VM_j)$ antes mesmo de estimar o *makespan* MS. Podemos utilizar o valor médio de $P(ca_i, VM_j)$ de todas as execuções passadas apenas consultando o repositório de proveniência disponível ou utilizando uma estimativa extra modelo. Esta estimativa extra modelo deve ser usada somente quando não existirem dados de proveniência disponíveis. Quando possuímos acesso aos dados de execuções passadas de *workflows* podemos calcular o tempo médio de execução das *cloud activities* e usá-lo como entrada do modelo de custo. Reparem que toda

informação utilizada neste modelo de custo se encontra representada e disponível no esquema de proveniência proposto (Capítulo 5).

3.4 Confiabilidade

Conforme discutido anteriormente na introdução desta tese, máquinas virtuais podem apresentar flutuações de desempenho durante o ciclo de vida de um *workflow* executado nas nuvens: elas podem apresentar falhas, mudar de servidor enquanto estão executando, ou serem criadas ou destruídas durante a execução do *workflow*. No modelo de custo proposto assumimos que as máquinas virtuais podem falhar seguindo uma distribuição Poisson (Larsen e Marx 2011) $F(VM_j)$, $\forall\ VM_j \in VM$ com valor constante. Esta premissa vem sendo adotada em diversos trabalhos de escalonamento de tarefas em grandes *clusters* e grades computacionais (Assayad *et al.* 2004, Boeres *et al.* 2011, Sonmez *et al.* 2010).

O valor $F(VM_j)$ expressa a probabilidade de certa quantidade de falhas ocorrerem na VM_j durante um intervalo de tempo fixo (durante a execução do *workflow* científico) independente do tempo decorrido desde o último evento de falha (a última falha da máquina virtual). Esta informação pode ser obtida também através do repositório de proveniência, caso as máquinas virtuais sejam as mesmas utilizadas em ensaios anteriores. No caso das máquinas que acabaram de ser instanciadas, estes valores não podem ser obtidos e devemos assumir que a confiabilidade é igual a 1 (melhor valor de confiabilidade possível). Esta é uma estimativa otimista, mas optamos por utiliza-la, pois inicialmente o provedor de nuvem garante a "qualidade" das máquinas virtuais instanciadas. Desta forma, é utilizada a métrica de custo de confiabilidade. O custo de confiabilidade é o inverso da confiabilidade da execução do *workflow*. Se minimizarmos o custo de confiabilidade, aumentamos a confiabilidade da execução do *workflow* como um todo. Assim, o custo de confiabilidade $R(ca_i,VM_j)$ pode ser definido como:

$$R(ca_i,VM_j) = F(VM_j) \times P(ca_i,VM_j) \quad (3)$$

O valor de $R(ca_i,VM_j)$ deve ser minimizado para que a confiabilidade se aproxime de 1. Desta forma, a confiabilidade total do sistema será dada pela soma de todas as confiabilidades calculadas para todas as *cloud activities* envolvidas na

execução. Consideremos $\gamma(VM_j)$ como o conjunto de *cloud activities* já executadas em VM_j, a confiabilidade total associada à VM_j pode ser calculada como:

$$R_p(VM_j) = \sum_{\forall ca_i \in \gamma(VM_j)} R(ca_i, VM_j) \quad (4)$$

3.5 Transferência de Dados

Outro custo importante que devemos contabilizar é o de transferência de dados através da rede. Como o acesso à nuvem se dá por meio de conexão comum com a Internet, esse custo muitas vezes pode ser proibitivo para a execução do experimento. Este custo está diretamente associado ao volume de dados de entrada e saída que deve ser transferido para e da nuvem, respectivamente. Como este modelo de custo foi elaborado para ser acoplado ao SciCumulus, assumimos que o *workflow W* lê e escreve informação em uma área compartilhada (Oliveira *et al.* 2011a), pois foi a escolha de implementação realizada nesta tese. Desta forma todos os escalonamentos possíveis em $\varphi(W,VM)$ leem e escrevem o mesmo volume de dados na mesma área compartilhada. Assim sendo, o custo de leitura e escrita é sempre o mesmo independente de $\varphi(W,VM)$. Independente deste fato, a modelagem realizada poderia ser aplicada mesmo no caso em que executássemos o *workflow* em um ambiente de arquitetura de discos não-compartilhados.

Para modelar o custo de transferência de dados vamos considerar ca_i como uma *cloud activity* associada à atividade a_i ($Act(ca_i) = a_i$) de W com tempo de execução definido como $ca_i.time$ e com $sched(ca_i,VM_k)$ definido. Da mesma forma consideremos a *cloud activity* ca_j associada à atividade a_j ($Act(ca_j) = a_j$) com $DepCa(ca_i, ca_j, D_{cai \rightarrow caj})$ e $sched(ca_j,VM_x)$, onde $D_{cai \rightarrow caj}$ é a quantidade de dados transferidos entre ca_i e ca_j. Considerando que $VM_k \neq VM_x$ temos que inserir uma nova dependência de dados e uma *cloud activity* artificial em W para representar a transferência de dados entre ca_i e a área compartilhada e da área compartilhada até ca_j. Desta forma, temos que substituir $depCa(ca_i, ca_j, D_{cai \rightarrow caj})$ por $depCa(ca_i, dm_w, D_{cai \rightarrow caj})$ e $depCa(dm_w, ca_j, D_{cai \rightarrow caj})$, onde dm_w é uma *cloud* acitivty artificial pra poder representar a transferência, além disso temos que definir $sched(dm_w, VM_k)$ e $sched(dm_w, VM_x)$. O tempo de execução da *cloud activity* dm_w será dado por:

$$\mathrm{dm_w.time} = \frac{\mathrm{Dca}_i \rightarrow \mathrm{ca}_j}{\min(VM_k.network, VM_x.\mathrm{network})} \quad (5)$$

O tempo total de transferência de dados (*TT*) para todas as *cloud activities* de *W* (assumindo que existam *(k+1)* transferências para *k cloud activities*) é dado por:

$$TT = \sum_{w=0}^{k} dm_w.time \quad (6)$$

3.6 Custo Monetário

Conforme discutido na introdução desta tese, o custo monetário é um dos diferenciais das execuções de *workflows* científicos em ambientes de nuvem. Como muitas vezes executamos *workflows* em ambientes públicos onde existe a cobrança pelo uso, esse custo deve ser avaliado de forma que não se torne proibitivo para a execução do experimento. Comecemos definindo o custo monetário de execução como $M(\varphi)$ que depende diretamente do esquema de precificação adotado pelo provedor de nuvem (como o Amazon EC2 ou GoGrid (Li et al. 2010)). A maioria dos provedores de nuvem existentes calcula o pagamento com base em uma janela de tempo, ou seja, o usuário paga um valor específico por janela de tempo, e o que varia é a granularidade da janela de tempo.

Por exemplo, o Amazon EC2 utiliza a janela de tempo de uma hora, enquanto que ao utilizar o GoGrid o usuário adota planos pré-pagos em que deposita um valor *a priori* e o provedor desconta os valores deste depósito por minuto de processamento (minuto de CPU) e não por minuto em que a máquina virtual está instanciada como no Amazon EC2. Esta parte da modelagem do modelo de custo que propomos nesta tese foi inspirada nas ideias de Kllapi *et al.* (2011). Assim sendo, o tempo total de execução deve ser calculado como uma soma de várias janelas de tempo de tamanho *Qt* o que nos leva ao conjunto $\delta = \{\delta_1, \delta_2, \ldots, \delta_x\}$ de janelas de tempo. Para cada máquina virtual envolvida na execução temos que pagar um valor por janela (seja ela com grão de hora ou minuto) que denominamos *Vq*. Vamos considerar $\omega(\delta_i, VM_j)$ como uma função para retornar se a VM_j está executando uma *cloud activity* na janela de tempo δ_i, dada por:

$$\omega(\delta i, VMj) = 1, se\ VMj\ está\ executando\ em\ \delta i\ e\ 0, caso\ contrário$$

Temos também que considerar o tempo em que as máquinas virtuais permanecem inativas enquanto aguardam a transferência de dados. Desta forma, o nosso custo monetário de execução é separado em duas partes principais. A primeira está relacionada com o custo de execução das *cloud activities* propriamente dito, e a segunda está relacionada com o tempo em que uma máquina virtual fica aguardando até que os

dados sejam transferidos. Assim, o custo monetário de execução é definido pela seguinte equação:

$$M(\varphi) = (Vq \times \sum_{j=1}^{|VM|} \sum_{j=1}^{MS/Qt} \omega(\delta_i, VM_j)) + (Vq \times \sum_{i=0}^{k} dm_i.time) \quad (7)$$

Além do custo de execução, os provedores de nuvem também cobram por unidade de informação transferida pela rede. Assim, o custo monetário de transferência $Tr(\varphi)$ também depende do esquema de preço do provedor de nuvem (como o Amazon EC2 ou GoGrid). Consideremos como Vt o valor que o cientista tem que pagar por cada unidade (seja ela qual for) de dados transferidos. Desta forma, o montante total a ser pago para a transferência de dados é:

$$Tr(\varphi) = Vt \times \sum_{j=0}^{k-1} \sum_{i=0}^{k-1} depCa.ds(ca_i, ca_j) \quad (8)$$

3.7 Modelo de Custo

O modelo de custo proposto é, então, baseado no tempo de execução, no custo monetário e em critérios de confiabilidade. Para escalonar as *cloud activities* com base nestes três objetivos, utilizamos um modelo de custo ponderado já amplamente adotado em protocolos de redes de computadores (Sardiña *et al.* 2009) em que o cientista tem de informar o peso de cada critério (que varia de cientista para cientista ou de *workflow* para *workflow*). A vantagem de oferecer um modelo de custo ponderado é que o cientista pode realizar uma sintonia fina de critérios de forma simples, sem que tenha que eleger um critério principal e outros "subordinados". Esta abordagem difere de diversos trabalhos anteriores onde o cientista tinha que eleger obrigatoriamente um critério que seria 100% mais importante que os outros e, caso fosse possível, os outros critérios seriam levados em consideração no escalonamento (Kloh *et al.* 2010, Mc Evoy e Schulze 2011). Estes trabalhos são devidamente comparados com a abordagem proposta no Capítulo 8 desta tese.

Desta forma, para cada VM_j em um conjunto *VM'* de máquinas virtuais que estão ociosas é requisitada uma *cloud activity* para ser processada. É, então, realizada uma pesquisa para descobrir a melhor *cloud activity* ca_i na lista de *cloud activities* disponíveis *Ca'* (as prontas para serem executadas, *i.e.* que não tem dependência com *cloud activities* que ainda não foram processadas) para executar na VM_j seguindo o modelo de custo de 3 objetivos. Dada uma $VM_j \in VM$ como a próxima máquina virtual

a executar uma *cloud activity*, temos de encontrar $ca_i \in Ca'$ que minimize a função de custo a seguir:

$$f(ca_i, VM_j) = \alpha_1 \times (P(ca_i,VM_j) + \sum_{j=0}^{k} \frac{depCa.ds(cai,caj)}{VM_j.network}) + \alpha_2 \times R(ca_i,VM_j) + \alpha_3 \times (Vh \times [P(ca_i,VM_j) + \sum_{j=0}^{k} \frac{depCa.ds(cai,caj)}{VM_j.network}] + Vt \times \sum_{j=0}^{k} depCa.ds(ca_i, ca_j)) \quad (9)$$

Note que a ca_i escolhida é aquela que satisfaz $F(ca_i, VM_j) = min \ \forall VM_j \in VM \{f(ca_i, VM_j)\}$. O peso associado a cada critério é uma variável na forma $0 \leq \alpha_i \leq 1$ ($i = 1,2,3$), que representa o nível de relevância de cada critério definido pelo cientista. Permitindo aos cientistas informar α_i, fornecemos uma maneira de realizar uma sintonia fina na execução do *workflow*. O espaço de busca para este modelo de custo é baseado nestes três objetivos, uma vez que o modelo de custo se propõe a encontrar o melhor conjunto de *cloud activities* disponíveis para executar em uma específica máquina virtual seguindo os critérios de confiabilidade, tempo de execução e critérios de custo monetário. Entretanto, não podem ser aplicadas técnicas de otimização tradicionais, pois o espaço de busca deste problema de escalonamento é demasiadamente grande (m^k, onde m define o número de máquinas virtuais envolvidas na execução e k a quantidade de *cloud activities*), e, portanto a busca por uma solução ótima pode ser inviável à medida que a quantidade de *cloud activities* aumenta.

Este modelo de custo deve ser acoplado a um algoritmo de escalonamento e desenvolvido no contexto de um sistema computacional. No Capítulo 4, apresentamos o algoritmo de escalonamento adaptativo guloso utilizado nesta tese e no Capítulo 5 o mediador para execução paralela de *workflows* na nuvem, onde tanto o modelo de custo e o algoritmo de escalonamento guloso são implementados.

3.8 Normalização dos Valores das Variáveis do Modelo

Para garantir a mesma magnitude dos três objetivos do modelo de custo proposto, os valores das variáveis devem ser normalizados, ou seja, devemos transformar todas as variáveis do modelo de modo que os limites máximo e mínimo do valor das variáveis sejam os mesmos e de preferência em intervalos pequenos como [0,1], independente do critério adotado. Como o método de normalização a ser adotado não impactaria de forma alguma no resultado do modelo de custo, estudamos alguns métodos existentes.

Optamos por adotar a técnica de normalização por desvio padrão (Larsen e Marx 2011), onde consideramos a posição média dos valores e os graus de dispersão em relação à média dos valores existentes. Assim, o novo valor da variável será dado por:

$$f(x) = \frac{(x-\mu)}{\sigma} \quad (10)$$

Onde μ é a média dos valores das variáveis do modelo e σ o desvio padrão calculado. Adotando esta normalização garantimos que os objetivos do modelo tenham a mesma amplitude.

Capítulo 4 - Escalonamento Adaptativo de *Workflows* Científicos em Nuvens Computacionais

Conforme mencionado no Capítulo 3 desta tese, o espaço de busca é demasiadamente grande. Na verdade, problemas desta ordem geralmente não são úteis sob o ponto de vista otimização, ou seja, é praticamente inviável descobrir uma solução ótima em um tempo aceitável conforme o tamanho de *m* e *k* crescem. Isso porque problemas de escalonamento de tarefas são conhecidamente problemas NP–Completo, mesmo em suas formais mais simples (Al-Azzoni e Down 2008, Smanchat *et al.* 2009, Yu e Buyya 2006). Em geral, para resolver problemas como este, temos três possibilidades:

i. Uso da força bruta;
ii. Aplicação de meta-heurísticas que não garantem a escolha da solução ótima ou;
iii. Utilização de algoritmos gulosos que focam em uma otimização local, com o objetivo de se aproximar ao máximo da otimização global, sem nenhuma garantia de que será alcançada a solução ótima.

No contexto desta tese, optamos por desenvolver um algoritmo guloso de escalonamento baseado no modelo de custo apresentado no Capítulo 3. De acordo com o que vimos no Capítulo 3, as soluções de escalonamento representam o *trade-off* entre os três critérios (tempo de execução, confiabilidade e custo monetário). Para explorar estes critérios, nesta tese vamos nos concentrar em estudar quatro cenários diferentes:

C1. Encontrar o escalonamento que leve ao menor tempo de execução;
C2. Encontrar o escalonamento que leve ao menor custo monetário;
C3. Encontrar o escalonamento mais "seguro" no qual as atividades apresentem a menor quantidade de falhas possível e;
C4. Encontrar um plano de execução (escalonamento) balanceado, no qual cada um dos critérios – tempo de execução total, confiabilidade e custo monetário - tem igual peso.

Na verdade, para implementar cada um desses cenários, basta que definamos diferentes valores de α_i (*i.e.* $0 \leq \alpha_i \leq 1$) no modelo de custo, a fim de se concentrar em cada critério específico. No entanto, independentemente do cenário em particular que está sendo resolvido executaremos o mesmo algoritmo de escalonamento, apenas variando o valor de α_i no modelo de custo.

A abordagem adaptativa proposta é baseada tanto no modelo de custo quanto em um conjunto de algoritmos inter-relacionados. Os algoritmos principais são os que implementam o escalonamento guloso (Cormen *et al.* 2009) e de dimensionamento de recursos, sendo que este último faz com que o ambiente se adapte à demanda de processamento gerenciando a quantidade de máquinas virtuais envolvidas na execução, aumentando ou diminuindo a sua quantidade sob demanda para atender ao prazo ea o orçamento limite informados pelos cientistas. Complementariamente propomos um algoritmo de agrupamento de *cloud activities* de forma a diminuir a sobrecarga de comunicação e transferência de dados, um algoritmo que executa a *cloud activity* e atualiza os metadados utilizados no escalonamento e um algoritmo que captura informações de desempenho das máquinas virtuais durante o curso de execução do *workflow*. Esse último algoritmo é fundamental uma vez que estamos trabalhando em um ambiente baseado em virtualização, onde não conhecemos de fato o poder computacional que nos está sendo disponibilizado. Assim, a abordagem adaptativa proposta neste capítulo é composta por cinco algoritmos:

i. Um algoritmo que captura informações de desempenho das máquinas virtuais envolvidas na execução
ii. Um algoritmo de escalonamento guloso;
iii. Um algoritmo de agrupamento de *cloud activities*;
iv. Um algoritmo para execução das *cloud activities* e;
v. Um algoritmo de dimensionamento de recursos.

Cada um destes algoritmos é detalhado a seguir neste capítulo. Na Seção 4.1 apresentamos o algoritmo de captura de informações de desempenho das máquinas virtuais na nuvem. Na Seção 4.2 apresentamos com detalhes o algoritmo guloso de escalonamento utilizado como base nesta tese. Na Seção 4.3 é discutido o algoritmo de agrupamento de *cloud activities*. Na Seção 4.4 apresentamos o algoritmo de execução e registro de metadados e finalmente na Seção 4.5 discutimos o algoritmo de dimensionamento de recursos.

4.1 Algoritmo de Captura de Informações de Desempenho

Conforme discutido anteriormente, como o ambiente de nuvem é baseado em virtualização de recursos, não temos garantias de qual é o poder computacional real que está sendo disponibilizado em uma máquina virtual, principalmente nos ambientes de

nuvens públicas. Por exemplo, no ambiente Amazon EC2, uma série de medidas é tomada para fornecer a cada máquina virtual uma quantidade consistente e previsível de capacidade de CPU, de acordo com o que é informado em sua página, porém sem garantias reais.

Desta forma, como não há garantias de desempenho, a quantidade de CPU neste ambiente é expressa em termos de Unidades Computacionais (UC) do EC2. Assim, uma UC do EC2 fornece a capacidade de CPU equivalente de um processador Xeon variando de 1,0 a 3,0 GHz. Ao longo do tempo, o Amazon EC2 adiciona ou substitui recursos que fazem parte da definição de uma UC do EC2, se acharem que estes recursos irão fornecer garantias mais nítidas sobre a capacidade computacional. Ou seja, o desempenho da máquina virtual não necessariamente é o que é informado pela Amazon, e devemos medir o desempenho real para um escalonamento eficiente.

Além disso, máquinas virtuais podem ser movidas enquanto executam, fazendo com que ocorra degradação no desempenho. O Algoritmo 1 tem como objetivo medir o desempenho das máquinas virtuais através de medidas obtidas com a execução de um programa de teste de desempenho (do inglês *toy program*) em cada uma das máquinas virtuais e atualizar o valor de *csi* de cada máquina virtual, valor este fundamental para o modelo de custo ponderado que foi proposto. O Algoritmo 1 deve ser preferencialmente implementado em um agente autônomo que execute independentemente dos *workflows* científicos. Tal agente é parte da arquitetura proposta e apresentada no Capítulo 5 desta tese.

Algoritmo 1 Captura de informações

Entrada:
VM: o conjunto de máquinas virtuais a serem utilizadas (dinâmico)
Saída:
-

```
1: VM' ← {}
2: while (true)
3:    VM ← listExistingVM
4:    VM' ← VM
5:    for each vm in VM'
6:       if not available(vm) then
7:          remove(vm,VM')
8:       else
9:          time ← executeToy(vm)
10:       updatecsi(vm,time)
12:      end if
12:   end for
```

13: VM ← VM'
14: updateProvenanceRepository(VM)
15: **end while**

No Algoritmo 1, as novas máquinas virtuais são adicionadas à lista de disponibilidade (linha 3) e as máquinas virtuais que não estão mais disponíveis são removidas da mesma (linha 7). Este algoritmo pode ser classificado como um algoritmo de espera ocupada (Oliveira et al. 2010e), uma vez que permanece em um laço contínuo, independente do escalonamento das *cloud activities* e da execução do *workflow*. Caso a máquina virtual esteja disponível, o algoritmo executa o programa de teste de desempenho (linha 9) e invoca o procedimento que atualiza o valor *csi* da máquina virtual (linha 10).

4.2 Algoritmo de Escalonamento Guloso

Uma vez que já conseguimos coletar os valores de *csi* de cada máquina virtual, já estamos aptos a iniciar o escalonamento baseado no modelo de custo. Desta forma, o Algoritmo 2 é o responsável por escolher a *cloud activity* mais adequada para a execução em uma determinada máquina virtual ociosa com base no modelo de custo proposto. Na prática, este algoritmo escolhe o melhor agrupamento de *cloud activities* (que são encapsuladas em uma nova *cloud activity* tal como apresentado a seguir no Algoritmo 3) para executar em uma máquina virtual específica.

O Algoritmo 2 inicia carregando a lista de *cloud activities* disponíveis e sem dependência de dados com nenhuma outra *cloud activity*, ou seja, as *cloud activities* que estão prontas para serem executadas (linha 3), e a lista de máquinas virtuais disponíveis (linha 4). Depois disso, o algoritmo inicializa as máquinas virtuais com informações fundamentais para gerenciar os agrupamentos de *cloud activities* (linhas 5-10). Estas informações são relativas ao tamanho do agrupamento que será realizado em um passo posterior.

Algoritmo 2 Escalonamento guloso

Entrada:
W: o *workflow* científico
VM: o conjunto de máquinas virtuais a serem usadas (dinâmico)
Deadline: o prazo para que o *workflow* termine
Budget: o orçamento limite informado pelos cientistas
Saída:
$\varphi(W,VM)$: o escalonamento realizado de *W* em *VM*

```
1: loadBalance(VM, Deadline, Budget)
2: φ(W,VM)←∅
3: available ←{ca_i ∈ CA| ∀ca_j ∈ CA ¬DepT(ca_i,ca_j, ds)}
4: ready ←{vm_j ∈ VM | ∀vm_j ∈ VM av(vm_j)}
5:  for each vm in VM do
6:    vm.partGranFactor ← 1
7:    vm. execTime ←  ∞
8:    vm.prevExecTime ← ∞
9:    vm.maxGranFactor ← ∞
10: end for each
11: while available ≠  ∅ do
12:  for each vm in ready do /* Ordernar a lista de VM */
13:       maxGranFactor   ← getMaxGranFactor (vm)
14:       partGranFactor ← getPartGranFactor (vm)
15:       avgTime ← getAvgTime(vm)
16:       if maxGranFactor ≠ ∞ then
17:         clusters ← group(available,maxGranFactor,avgTime)
18:       else
19:         clusters ← group(available,partGranFactor, avgTime)
20:       end if
21:       for each x in clusters do
22:         cost ← f(x, vm)
23:         possible ← possible + { sched(x_i, VM_j, cost)}
24:       end for each
25:     chosen ← min(possible)
26:     φ(W,VM) ← φ(W,VM) + chosen
27:     perform(chosen,Threshold)
28:     ready ← ready − {vm} + {vm_j ∈ VM | av(vm_j)}
29:     available ←{ca_i ∈ CA | ∀ca_j ∈ CA ¬DepT(ca_i,ca_j, ds)}
30:  end for each
31: end while
32: return φ(W,VM)
```

Quando todas as máquinas virtuais são inicializadas, o algoritmo começa a analisar se existem *cloud activities* disponíveis para serem executadas (linha 11). Se houver pelo menos uma *cloud activity* para executar o algoritmo procura uma máquina virtual ociosa para executar as *cloud activity* disponíveis (linha 12). É importante ressaltar que o Algoritmo 2 ordena as *cloud activities* de acordo com a capacidade da máquina virtual e do cenário em questão. Baseado nos valores de α_i ($0 \leq \alpha_i \leq 1$) o algoritmo consegue inferir qual o cenário que estamos lidando (C1, C2, C3 e C4). Se o cenário for o C1, então o algoritmo ordena as máquinas virtuais da menos potente (*csi* menor) para a máquina virtual mais poderosa (o maior *csi*). Por outro lado, se o cenário considerado é o C2, em seguida, o algoritmo ordena a lista de máquinas virtuais disponíveis a partir da mais poderosa (o maior *csi*) para a menos poderosa (*csi* menor).

Se houver alguma máquina virtual ociosa, o algoritmo analisa se há um fator de granularidade ou um fator de granularidade parcial associado a esta máquina virtual.

Estes fatores indicam qual o tamanho máximo do agrupamento que aquela máquina virtual suporta sem degradação de desempenho severa. Estes fatores são usados para agrupar as *cloud activities* disponíveis para execução e as encapsular em uma nova *cloud activity* (linhas 13-20). Então, com base nessa nova *cloud activity* (variável *clusters*) o algoritmo calcula o agrupamento mais adequado para executar na máquina virtual usando o modelo de custo proposto (linhas 21-27), gerando os possíveis *sched* e escolhendo o que oferecer o menor custo. Depois disso, as listas de *cloud activities* e de máquinas virtuais disponíveis são atualizadas (linhas 28-29). No final do algoritmo o plano de escalonamento final é informado ao cientista (linha 32). É importante ressaltar que este plano não é gerado *a priori* e sim durante a execução do *workflow*.

A complexidade do algoritmo no pior caso é $O(n^2k)$, no qual n representa o número de *cloud activities* do *workflow* e k o número de máquinas virtuais envolvidas na execução. Cabe salientar que apesar de existirem três estruturas de varredura aninhadas (*while*, *for each* e *for each*), o segundo *for each* é complementar ao primeiro *while*, ou seja, o número de *cloud activities* processadas pelo primeiro *while* é dependente do número de *cloud activities* agrupadas no *loop* do *for each*. Assim, na prática, o pior caso ocorre quando cada agrupamento de *cloud activities* contém uma e somente uma *cloud activity*. Além disso, como estamos lidando com *workflows* que contém uma grande quantidade de *cloud activities*, intuitivamente podemos perceber que $n >> k$, fazendo com que o fator k não represente um impacto grande na complexidade do algoritmo.

4.3 Algoritmo de Agrupamento de *Cloud Activities*

O objetivo principal do Algoritmo 3 é produzir as novas *cloud activities* através do agrupamento e encapsulamento de duas ou mais *cloud activities* em uma nova. O foco deste processo de agrupamento é reduzir a sobrecarga de comunicação e dados de transferência. Toda vez que uma máquina virtual envia uma mensagem solicitando uma nova *cloud activity* para processar temos que lidar com um custo de comunicação. Em casos em que as *cloud activities* são rápidas de serem executadas e a quantidade de *cloud activities* a serem processadas for muito grande, podemos sofrer o impacto desta comunicação. Se pudermos agrupar *cloud activities* que consomem (ou consomem parcialmente) os mesmos arquivos de entrada poderíamos reduzir essa sobrecarga.

Algoritmo 3 Agrupamento de cloud activities

```
Entrada:
  Ca: lista de cloud activities disponíveis
  GranFactor: fator de granularidade
  AvgTime : tempo de execução médio
Saída:
  Clusters: lista de agrupamentos gerados

1: if GranFactor = 0 then
2:    Clusters ← 0
3:    Return
4: end if
5: if GranFactor > |Ca| then
6:   Clusters ← Ca
7:   Return
8: end if
9: subSetCount ← 0
10: subSet ← {}
11:  while Ca ≠ ∅ do
12:     while (subSetCount < GranFactor) or (Ca ≠ ∅) do
13:        act ← getAvailable(Ca)
14:       subSet ← subSet + act
15:       predExecTime ← getPredictedTime(SubSet)
16:       predAvgTime ← predExecTime / GranFactor
17:       if predAvgTime ≤ AvgTime then
18:          subSetCount ← subSetCount + 1
19:          Ca ← Ca – subSet
20:       else
21:          subSet ← subSet - act
22:       end if
23:     end while
24: Clusters ← Cluster + subSet
25: subSet ← {}
26: subSetCount ← 0
27 end while
28: return Clusters
```

Ao fazer isso, evitamos as transferências de dados que são caras em ambientes de nuvem (tanto em termos de tempo de execução quanto em termos monetários). Além disso, mesmo que as *cloud activities* não consumam os mesmos dados de entrada, evitamos a troca de mensagem e o sincronismo entre os nós de comunicação que impõem essa sobrecarga. Por exemplo, suponhamos que a *cloud activity* ca_1 consuma os arquivos f_1 e f_2 e a *cloud activity* ca_2 consuma f_1, f_2 e f_3. Se escalonarmos ca_1 na VM_1 e ca_2 na VM_2 temos de transferir dois arquivos para VM_1 e 3 arquivos para VM_2 enquanto ao ponto que se encapsularmos ca_1 e ca_2 em uma nova *cloud activity* ca_3 teremos que transferir apenas 3 arquivos para a máquina virtual escolhida (f_1, f_2 e f_3). No entanto, um dos desafios fundamentais é como determinar o tamanho dos agrupamentos e quais as *cloud activities* que devem ser agrupadas.

A ideia é então podermos ajustar dinamicamente o tamanho do agrupamento com base em dados de proveniência já existentes (para estimar o tempo de execução e verificar quais arquivos são de fato consumidos e produzidos por cada *cloud activity*). Ao analisar dados de proveniência relacionados com os experimentos executados anteriormente, somos capazes de estimar o tempo de execução da nova *cloud activity* (duas ou mais *cloud activities* encapsuladas em uma nova). Além disso, pretende-se manter o tempo de execução médio das *cloud activities* associadas à máquina virtual em questão.

O Algoritmo 3 começa verificando se o fator de granularidade é nulo e neste caso, o agrupamento é impossível de ser definido (linha 2). Se o número de *cloud activities* disponíveis a serem agrupadas é inferior ao fator de granularidade, apenas um agrupamento é formado (linha 6). Por outro lado, se existem mais *cloud activities* do que o valor do fator de granularidade, o algoritmo gera os vários agrupamentos possíveis, escolhendo *cloud activities* disponíveis (que preferencialmente compartilham arquivos de entrada para evitar a transferência de dados) a serem agrupadas para manter uma aproximação do tempo médio de execução (linhas 11-27).

4.4 Algoritmo de Execução de *Cloud Activities* e Registro de Metadados

Conforme visto anteriormente, para agrupar as *cloud activities* são necessárias algumas informações de granularidade. Para tal, estas informações devem ser capturadas de alguma forma. O Algoritmo 4 é responsável pelo registro do fator de granularidade para cada máquina virtual no sistema. Esta informação é um pré-requisito para o Algoritmo 3 apresentado anteriormente. O Algoritmo 4 inicia executando a *cloud activity* (ou grupo de *cloud activities*) (linha 1). Depois disso, ele calcula o tempo médio de execução (linha 3) e o tempo médio de execução da execução anterior (linha 4) relativo a maquina virtual. Se o novo tempo médio é inferior ao tempo médio anterior (linha 5) o algoritmo aumenta o fator de granularidade (linha 9). Por outro lado, se o tempo médio novo é superior ao tempo médio anterior, o algoritmo define o fator de granularidade máximo com o valor atual (linha 6).

Algoritmo 4 Execução de cloud activities

Entrada:
Chosen: um dado escalonamento
Threshold: o threshold para definir o fator de granularidade máximo
Saída:
-
1: execTime ← execute(chosen)
2: vm ← getVM(chosen)
3: avgTime ← $\frac{\text{execTime}}{\text{vm.partGranFactor}}$
4: prevAvgTime ← $\frac{\text{vm.prevExecTime}}{\text{vm.partGranFactor} - 1}$
5: **if** (avgTime ≥ prevAvgTime × Threshold) **then**
6: vm.maxGranFactor ← partGranFactor
7: vm.avgTime ← prevAvgTime
8: **else**
9: vm.partGranfactor ← vm.partGranFactor + 1
10 vm.avgTime ← avgTime
11:**end if**
12:vm.prevExecTime ← vm.execTime

É importante ressaltar que o fator de granularidade máximo só é configurado caso a média de tempo da execução atual tenha superado a média de tempo anterior ajustada baseado em um valor de *threshold* informado pelo cientista. Por exemplo, podemos aceitar uma perda de 5% até 10% quando aumentamos o fator de granularidade, então o *threshold* deveria ser configurado com valor entre 1,05 e 1,10 (5% e 10% respectivamente).

4.5 Algoritmo de Dimensionamento de Recursos

A abordagem proposta nesta tese é adaptativa em relação à distribuição das *cloud activities* de acordo com o modelo de custo, ao agrupamento de *cloud activities* e em relação à quantidade de recursos utilizada. O Algoritmo 5 concentra-se em permitir que o sistema dimensione a quantidade de recursos disponíveis (máquinas virtuais) para uso de forma que se possa ajustar ao prazo e ao orçamento limite informados pelos cientistas. Desta forma, este algoritmo permite que os cientistas rapidamente dimensionem o tamanho do *cluster* virtual que foi criado (quantidade de máquinas virtuais), tanto para cima quanto para baixo, de acordo com suas necessidades de computação (prazo e orçamento).

Algoritmo 5 Dimensionamento de recursos

Entrada:
Deadline: o prazo informado pelo cientista
Budget: o orçamento limite informado pelo cientista
VM: a lista de máquinas virtuais disponíveis
CA: a lista de *cloud activities* disponíveis
Threshold: o *threshold* para definir a variação do *throughput*

Saída:
-

1: throughput ← ∞
2: prevThroughput ← ∞
3: **while** (CA ≠ ∅) **do**
4: VM' ← VM
5: throughput ← $\frac{|VM|\times|CA|}{\sum_{i=0}^{k} P(ca_i,VM_j)}$
6: **if** throughput ≤ prevThroughput × Threshold **then**
7: MS ←simulateMS(VM,CA)
8: MC ← simulateMC(VM,CA)
9: **if** (MS ≥ Deadline) **and** (MC ≤ Budget) **then**
10: **while** MS' ≤ Deadline **and** MC' ≤ Budget **do**
11: VM' ← VM' + 1
12: MS'← simulateMS(VM',CA)
12: MC' ← simulateMC(VM',CA)
13: **end while**
14: VM ← VM'
15: **end if**
16: **if** (MS ≤ Deadline) **and** (MC ≥ Budget) **then**
17: **while** MS' ≤ Deadline **and** MC' ≤ Budget **do**
18: VM' ← VM' - 1
19: MS'← simulateMS(VM',CA)
20: MC' ← simulateMC(VM',CA)
21: **end while**
22: VM ← VM'
23: **end if**
24: **if** (MS ≥ Deadline) **and** (MC ≥ Budget) **then**
25: VM ← {}
26: CA ← {}
27: **end if**
28: **end while**

O Algoritmo 5 inicia verificando se o *throughput* (quantidade de *cloud activities* terminadas por unidade de tempo) tem variado (linha 6) de acordo com o que foi definido na Equação 1 no Capítulo 4. Caso contrário, nada muda no conjunto de máquinas virtuais disponíveis. Se o rendimento reduziu, ou seja, o valor do *throughput* diminuiu, o algoritmo simula o novo *makespan* (MS) conforme definido na Equação 2 e o novo custo monetário (MC) (linhas 7-8) baseado no modelo de custo. Se o *makespan* novo é maior do que o prazo informado pelo cientista e o custo monetário é menor do que o orçamento informado, o número de máquinas virtuais é ampliado, a fim de

cumprir o prazo limite (linhas 9-15). Por outro lado, se o prazo for atendido, mas o custo monetário é maior que o orçamento, então o algoritmo diminui o número de máquinas virtuais envolvidas na execução (linhas 16-23). Esta redução no número de máquinas virtuais pode causar a interrupção de algumas *cloud activities*. Desta forma, algumas *cloud activities* devem ser escalonadas novamente.

Capítulo 5 - SciCumulus: um Mediador para Execução Paralela de *Workflows* Científicos em Nuvens Computacionais

Nos capítulos anteriores foram apresentados conceitos relativos ao escalonamento adaptativo para execução paralela de *cloud activities* de *workflows* científicos em ambientes de nuvens. Este capítulo apresenta o SciCumulus, que é um mediador para gerenciar a execução paralela de *workflows* científicos neste tipo de ambiente. A motivação do desenvolvimento do SciCumulus foi baseada no bom desempenho obtido pelo mediador Hydra (Coutinho *et al.* 2010, 2011, Guerra *et al.* 2009, Ogasawara *et al.* 2009a) e pelo motor de execução Chiron (Ogasawara *et al.* 2011) na distribuição de atividades de varredura de parâmetros de *workflows* científicos em *clusters* de computadores. Estes desenvolvimentos anteriores forneceram subsídios para o desenvolvimento do SciCumulus.

O SciCumulus foi desenvolvido como uma solução computacional para atender as fases do ciclo de vida de um *workflow* executado em nuvem, desde a definição do *workflow* e a carga dos dados até a criação do *cluster* virtual e as consultas aos dados de proveniência. Assim, neste capítulo detalharemos a arquitetura do SciCumulus, seu modelo de proveniência e uma série de serviços que são acoplados ao arcabouço proposto.

Na seção 5.1 é apresentada a arquitetura conceitual do SciCumulus. Na seção 5.2 é apresentado o SciCumulus-ECM, um serviço que visa configurar o ambiente para o SciCumulus de forma otimizada. Na seção 5.3 é apresentado o SciMultaneous, um serviço do SciCumulus que foca em reexecutar *cloud activities* que falharam. Na seção 5.4 é apresentado o SciLightning, um serviço que tem como objetivo o monitoramento do *workflow* que está sendo executado em paralelo na nuvem. Na seção 5.5 apresentamos o BioSciCumulus, o serviço de execução de experimentos de biologia evolutiva que foi desenvolvido com base no SciCumulus. Na Seção 5.6 discutimos sobre a representação dos *workflows* no SciCumulus. Na Seção 5.7 apresentamos o modelo de proveniência utilizado no SciCumulus. Na seção 5.8 é apresentado o modelo de execução do SciCumulus. E finalmente na Seção 5.9 são apresentados detalhes de implementação.

5.1 Arquitetura

O SciCumulus é um mediador projetado para distribuir e controlar a execução paralela de *cloud activities* de *workflows* científicos (ou até mesmo *workflows* científicos completos) despachadas a partir de um SGWfC em um ambiente de nuvem, como o Amazon EC2 (Amazon EC2 2010). O SciCumulus orquestra a execução das *cloud activities* do *workflow* científico em um conjunto distribuído de máquinas virtuais (*cluster* virtual), oferecendo dimensionamento dos recursos durante o curso de execução do *workflow* e mecanismos de tolerância a falhas através de seus serviços.

Para isolar os cientistas da complexidade de distribuir as atividades dos *workflows* científicos em ambientes de nuvem, o SciCumulus fornece mecanismos para oferecer paralelismo por varredura de parâmetros. Controlar esse tipo de paralelismo em ambientes de nuvem é difícil quando a gerência é feita de forma *ad hoc* devido à grande quantidade de *cloud activities* a serem gerenciadas. Desta forma, o SciCumulus fornece a infraestrutura necessária para dar o apoio computacional para o paralelismo de *workflows* com coleta de proveniência distribuída. O SciCumulus (Oliveira et al. 2010b, 2010c, 2011c, 2011a) foi projetado para seguir uma arquitetura de quatro camadas:

i. Camada Cliente: esta camada é responsável por realizar a interface entre o ambiente de nuvem e o SGWfC. Seus componentes são instalados nas máquinas dos cientistas e despacham as atividades do *workflow* para serem executadas em paralelo no ambiente de nuvem usando um SGWfC local, como o VisTrails (Callahan *et al.* 2006);
ii. Camada de Distribuição: esta camada cria e gerencia a distribuição de *cloud activities* em uma ou mais máquinas virtuais instanciadas em um ambiente de nuvem. Seus componentes podem ser instalados em qualquer ambiente, mas preferencialmente na nuvem para diminuir o impacto de comunicação com os componentes da camada de execução;
iii. Camada de Execução: esta camada executa efetivamente uma determinada *cloud activity*. É responsável pela execução de programas encapsulados em *cloud activities* e por coletar dados de proveniência. Seus componentes são instalados nas várias máquinas virtuais envolvidas na execução paralela das *cloud activities*;

iv. Camada de Dados: essa camada é responsável por armazenar dados de entrada e dados de proveniência consumidos e gerados pela execução paralela das atividades dos *workflows* científicos. Além disso, essa camada tem informações sobre as características do ambiente coletadas por um agente autônomo. Seus componentes estão instalados em máquinas virtuais específicas para armazenamento de dados.

A arquitetura do SciCumulus pode ser teoricamente implantada em qualquer ambiente de nuvem (como o Amazon EC2 ou o GoGrid) e conectada a qualquer SGWfC existente, diminuindo o esforço de desenvolvimento por parte dos cientistas. A seguir detalhamos a arquitetura do SciCumulus e explicaremos cada um de seus componentes. A Figura 11 apresenta a arquitetura conceitual do SciCumulus em suas quatro camadas.

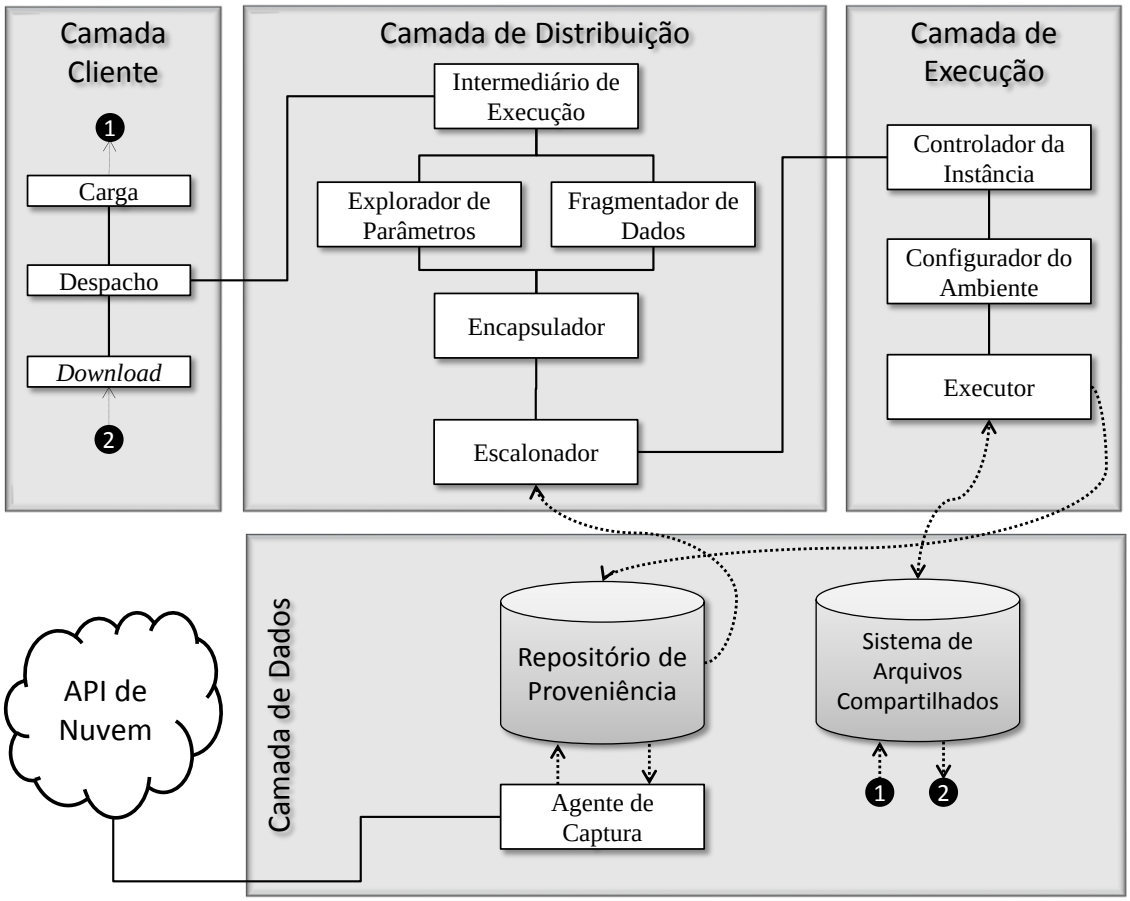


Figura 11 Arquitetura conceitual do SciCumulus

A camada cliente é responsável por iniciar a execução paralela de atividades de *workflows* na nuvem. Os componentes da camada cliente podem ser instalados em

qualquer SGWfC existente, tais como o VisTrails, o Kepler e o Taverna. Os componentes desta camada devem ser executados na fase de configuração do ciclo de vida do *workflow* executado em nuvem, uma vez que a transferência dos dados e o despacho da atividade para a nuvem são configurações prévias à execução paralela do *workflow*. Os componentes instalados no SGWfC iniciam a execução paralela do *workflow* na nuvem. Há três componentes principais nesta camada: o componente de carga, o componente de despacho, e o componente de *download*.

O componente de carga move os dados de entrada para a nuvem, enquanto que o componente de *download* reúne os resultados finais nas máquinas virtuais e os traz para a máquina do cientista. Em outras palavras, esses componentes são responsáveis pela preparação de dados de entrada e sua carga e pela transferência dos dados de saída. Os componentes de carga e de *download* devem ser preferencialmente invocados quando o experimento científico recebe/produz dados de entrada/saída de tamanho de pequeno a médio, uma vez que a transferência de dados para/de o ambiente de nuvem ocorre sobre uma conexão comum com a internet. Por exemplo, se os cientistas pretendem executar um *workflow* passando um arquivo de 100 MB de entrada, é possível transferir essa quantidade de dados sem grandes impactos no desempenho do *workflow*. Entretanto, esta transferência pode não ser viável quando os cientistas desejam lidar com dados de entrada grandes (100 GB, por exemplo). Nos casos de processamento de grande quantidade de dados, estes mesmos dados preferencialmente devem estar na nuvem para que não ocorra um impacto de transferência de dados. Nestes casos, portanto, os cientistas devem transferir tais dados de suas estações de trabalho para nuvem *a priori*.

O componente de despacho cria um arquivo de configuração (que pode ser representado como um arquivo XML), a partir de especificações realizadas pelo cientista, que armazena a configuração para a execução distribuída e inicia o processo de execução das *cloud activities* na nuvem se comunicando diretamente com os componentes da camada de distribuição. Este arquivo de configuração contém o nome da imagem a ser utilizado, quantas máquinas virtuais utilizar (caso o cientista opte por executar no modo estático), quais os parâmetros serão explorados, quais os valores a explorar, etc.

A camada de distribuição controla a execução de atividades paralelas em nuvens através da criação e gerenciamento das *cloud activities* que contêm o programa a ser executado, a estratégia paralela a ser seguida (indicando quais parâmetros explorar, por

exemplo), os valores dos parâmetros e dados de entrada a serem consumidos. A camada é composta por cinco componentes: o intermediário de execução, o explorador de parâmetros, o fragmentador de dados, o encapsulador e o escalonador. Estes componentes são invocados na fase de execução do ciclo de vida do *workflow* executado em nuvens. O intermediário de execução faz a interface entre a camada cliente e a camada de distribuição. Ele inicia a execução paralela do *workflow* considerando os metadados fornecidos pelo componente de despacho na camada cliente. Dependendo da estratégia paralela escolhida pelos cientistas, este componente pode invocar o explorador de parâmetros e/ou o fragmentador de dados. O intermediário de execução também é responsável por enviar mensagens de sincronização para o componente de despacho na camada cliente. O explorador de parâmetros manipula as combinações de parâmetros recebidos pela camada cliente para uma atividade específica do *workflow* que está sendo paralelizado (ou o sub-*workflow*, caso a *cloud activity* seja composta de mais de uma atividade). Este componente produz as combinações dos valores dos parâmetros a serem consumidos por diversas *cloud activities*.

O fragmentador de dados atua quando os dados de entrada devem ser fragmentados (de acordo com a definição do cientista), e gera subconjuntos dos dados que podem ser distribuídos ao longo das máquinas virtuais durante a execução juntamente com os valores de parâmetros. É importante ressaltar que o componente fragmentador de dados invoca um programa (definido pelos cientistas) que fragmenta os dados de entrada. O encapsulador gera todas as *cloud activities* a serem transferidas para as máquinas virtuais envolvidas na execução paralela.

O escalonador define quais máquinas virtuais receberão uma *cloud activity* específica para executar. Note que as máquinas virtuais podem ser fornecidas por qualquer provedor de nuvem. O escalonador tem de levar em conta as máquinas virtuais disponíveis para uso, as permissões para acesso, e o poder computacional de cada uma delas (valor de *csi*). O escalonador do SciCumulus é baseado no modelo de custo e no algoritmo de escalonamento apresentados nos Capítulos 3 e 4, respectivamente. Ele funciona em dois modos diferentes: estático e adaptativo. No modo estático, o escalonador considera apenas as máquinas virtuais disponíveis no início da execução, o que pode gerar problemas de desempenho e, por isso, este modo de trabalho não é indicado. Este problema, no entanto, pode ser dirimido se pudermos estimar *a priori*, a

quantidade de máquinas virtuais a serem utilizadas, conforme discutiremos na Seção 5.2.

Por outro lado, o modo adaptativo visa analisar os recursos disponíveis para a nuvem durante o curso da execução do *workflow*, a fim de usar mais (ou menos) máquinas virtuais e adaptar o tamanho do grupo de atividades de acordo com a capacidade de processamento das máquinas virtuais disponíveis e de acordo com as restrições informadas pelo cientista. Baseado no escalonamento escolhido, o escalonador transfere as *cloud activities* para as máquinas virtuais disponíveis e controla a sua execução. Além disso, pode também combinar os resultados parciais (se necessário) para organizar os resultados finais a serem transferidos para a camada cliente (que está instanciada no SGWfC).

A camada de execução é responsável por invocar os códigos executáveis nas diversas máquinas virtuais disponíveis para uso. Ela é composta por três componentes principais: o controlador de instância, o configurador do ambiente e o executor. Similarmente aos componentes da camada de distribuição, esses componentes fazem parte da fase de execução do ciclo de vida de um *workflow* executado em nuvem. O controlador de instância faz a interface entre a camada de distribuição e a camada de execução. Ele é também responsável por controlar o fluxo de execução na máquina virtual quando a *cloud activity* é associada a dois ou mais aplicativos para serem executados (um sub-*workflow*, por exemplo). O configurador do ambiente configura todo o ambiente, definindo as variáveis de ambiente, e criando diretórios necessários para gerenciar a execução. Ele desempacota a *cloud activity*, cria réplicas do programa para um local específico, e cria espaços de trabalho (do inglês *workspaces* - áreas diferentes no sistema de arquivos compartilhado) para armazenar informações sobre uma execução de uma *cloud activity* específica, como por exemplo, seus arquivos de configuração e os dados a serem consumidos. O executor invoca a aplicação específica e coleta os dados de proveniência, armazenando-os em um repositório também localizado na nuvem (um banco de dados relacional).

Finalmente, a camada de dados contém todos os repositórios de dados utilizados pelo SciCumulus. Ela possui dois componentes principais: o sistema de arquivos compartilhados, e o repositório de proveniência. Os componentes da camada de dados são invocados nas fases de execução (como visto anteriormente) e análise do ciclo de vida do *workflow* executado em nuvem. O repositório de proveniência contém dados

importantes de proveniência (Freire *et al.* 2008b) coletados durante o curso do experimento. Além disso, ele contém informações sobre o ambiente de nuvem, como os tipos de máquinas virtuais disponíveis, as máquinas virtuais instanciadas no momento e as características de localidade destas máquinas. Esta informação é captada por um agente autônomo que monitora o ambiente a procura de mudanças no mesmo (novas máquinas virtuais disponíveis ou máquinas virtuais destruídas). O sistema de arquivos compartilhado contém todos os dados de entrada consumidos por diversas *cloud activities* durante a execução paralela.

Os componentes supracitados fornecem o arcabouço conceitual mínimo para a execução paralela de *workflows* científicos em nuvens computacionais. Entretanto, existem questões importantes de implementação, otimizações que podem ser desenvolvidas e necessidades importantes dos cientistas que não são atendidas com esta arquitetura básica. Desta forma, uma série de serviços adicionais foi desenvolvida em conjunto com alunos de M.Sc. e será explicada a seguir nesta tese juntamente com detalhes de implementação do SciCumulus e o modelo de proveniência utilizado.

5.2 SciCumulus-ECM

Conforme mencionado anteriormente, o SciCumulus pode trabalhar com os modos de escalonamento estático e adaptativo. Em ambos os modos, o conjunto inicial de máquinas virtuais a serem utilizadas pode impactar (positivamente ou negativamente) na execução. No caso do escalonamento estático este problema fica mais evidente, pois as máquinas disponíveis no início da execução serão as mesmas (ou uma quantidade menor, caso alguma pare de funcionar) até o final da execução. Se este conjunto de máquinas virtuais for mal dimensionado, a execução do *workflow* pode sofrer impactos negativos. No caso do escalonamento adaptativo, apesar do mesmo conseguir aumentar e diminuir a quantidade de máquinas virtuais envolvidas na execução durante o curso do experimento, esta adaptabilidade introduz uma sobrecarga de processamento. Se a quantidade de máquinas for ideal (ou perto do ideal) no início da execução, poucas mudanças na quantidade de recursos precisarão ser realizadas.

Entretanto, dimensionar o conjunto inicial de máquinas virtuais a serem utilizadas não é uma tarefa trivial. Por exemplo, o Amazon EC2 oferece mais de 10 diferentes tipos de máquinas virtuais que podem ser instanciadas para utilização. Dessa forma, é necessário determinar quais desses recursos computacionais serão instanciados

e utilizados, seguindo um critério específico como o tempo total de execução do *workflow* ou custo monetário. Com este objetivo, foi definido um serviço na arquitetura do SciCumulus que, em síntese, deve inicialmente capturar informações fundamentais do ambiente de nuvem, catalogar recursos com funcionalidades equivalentes que podem ser selecionados, e aplicar um modelo de custo a fim de determinar a configuração ideal do ambiente. É importante ressaltar que este problema é inerente às nuvens, pois sua configuração é naturalmente dinâmica.

Esta seção apresenta o SciCumulus-ECM (*Environmental Cost Model*) (Viana *et al.* 2011b, 2011a), o serviço de dimensionamento do SciCumulus baseado em dois limites pré-estabelecidos pelos cientistas (tempo de execução e custo monetário). Com isso, o SciCumulus-ECM permite melhor uso dos recursos e cumprimento dos requisitos impostos pelos mesmos. O SciCumulus-ECM vem sendo desenvolvido no contexto da dissertação de mestrado do aluno Vitor Gamboa Viana, sob orientação da Prof.ª. Marta Lima de Queirós Mattoso.

A partir dos requisitos descritos pelos cientistas acerca da execução do *workflow* (tempo total de execução e custo monetário de execução), o SciCumulus-ECM determina os recursos da nuvem que serão instanciados para execução de um determinado *workflow*, em um determinado tempo. Em outras palavras, este serviço é responsável por aperfeiçoar a seleção e instanciação dinâmica de máquinas virtuais antes de escalonar as *cloud activities* de fato. Como a escolha dos tipos de máquinas virtuais a serem utilizadas no escalonamento é um problema análogo ao problema do caixeiro viajante (Applegate *et al.* 2007), sua complexidade é também categorizada como NP (Cormen *et al.* 2009). Isto o remete para um campo de complexidade exponencial, em que o esforço computacional necessário para a sua resolução cresce exponencialmente com o tamanho do problema. Assim, dado que é difícil (se não impossível) determinar a solução ótima deste problema em tempo hábil, optamos por utilizar um método baseado em heurísticas que, do ponto de vista matemático, não asseguram a obtenção de uma solução ótima, porém apresentam grande chance de gerar uma solução aceitável em tempo viável (Bäck 1996).

Desta forma, a implementação do SciCumulus-ECM aplica heurísticas definidas em Algoritmos Genéticos (Goldberg 1989, Ochi 1998) para a escolha do conjunto de máquinas virtuais a serem utilizadas. Esses algoritmos genéticos utilizados na sintonia inicial dos parâmetros da execução adaptativa se mostram eficientes, porém sua

utilização no escalonamento adaptativo em si insere severos *overheads* de execução. Segundo Ochi (1998), a ideia principal por trás dos algoritmos genéticos é tentar simular algumas etapas do processo de evolução das espécies. A estrutura básica de um algoritmo genético (denotado a partir de agora como simplesmente AG) é formada pelas seguintes etapas:

i. Representação de uma solução na estrutura de cromossomo;
ii. Construção de uma população inicial de cromossomos;
iii. Um mecanismo de avaliar os cromossomos segundo suas aptidões;
iv. Um mecanismo de reprodução de cromossomos - Operador Genético e;
v. Definição de parâmetros de uma AG.

Inicialmente, é define-se um objeto de configuração, ou seja, uma descrição de como os cromossomos devem ser configurados. Nesta tese, cada cromossomo possui cinco genes, um para cada tipo de máquina virtual que consideramos na execução do SciCumulus (*large, micro, extra-large, extra-large de alta capacidade* e *extra-large quádrupla*). Definiu-se que os valores (*i.e.* alelos) dos genes sejam do tipo inteiro, uma vez que representam a quantidade de máquinas virtuais desse tipo que serão instanciadas. Também é necessário especificar um limite inferior e superior, que definem os valores para cada tipo de máquina virtual (comumente o valor mínimo é 1 e o máximo é 20, uma vez que foi utilizado o Amazon EC2 e esta é uma limitação do provedor). A população inicial de cromossomos é gerada aleatoriamente e foi configurado para que ocorram no máximo 200 gerações, uma vez que não se pode consumir muito tempo na configuração do ambiente ao invés de utilizar esse tempo na execução do *workflow*. A cada geração, é realizada uma simulação da execução do *workflow* baseada em um simulador (Viana *et al.* 2011a) que utiliza o modelo de custo e os algoritmos de escalonamento propostos nesta tese. O resultado da simulação é um valor de *fitness* (tempo de execução e valor monetário) que é utilizado para avaliar se a geração atual (configuração do ambiente) é melhor (ou não) do que a anterior. O valor de *fitness* é utilizado para verificar se o teste atual está se aproximando (ou não) da melhor solução. No Capítulo 7 desta tese são apresentados resultados de avaliação do SciCumulus-ECM.

5.3 SciMultaneous

Quando pensamos em computação em nuvem, em geral, pensamos em conceitos como tolerância às falhas. Entretanto, no que tange à execução de *workflows* em nuvem, ainda existem muitos pontos a serem aperfeiçoados (Oliveira *et al.* 2010c) principalmente com relação ao controle de falhas e re-execução de atividades. Existem dois níveis de tolerância que podemos considerar: nível de tarefa e nível de *workflow*. A tolerância em nível de tarefa é utilizada em sistemas paralelos e distribuídos, e está relacionada às falhas das tarefas do *workflow* que foram executadas em paralelo, no contexto desta tese a execução das diversas *cloud activities*. A tolerância em nível de *workflow* requer técnicas aplicadas ao fluxo de execução, usadas para tratar condições de erro do *workflow* como um todo (Ferreira *et al.* 2010), o que está fora do escopo desta tese.

Uma grande vantagem obtida com o tratamento de erros e re-execuções no SciCumulus é que a nuvem oferece novas possibilidades, pois devido à sua elasticidade e disponibilidade, podemos traçar heurísticas variadas, de acordo com a importância da atividade e de suas *cloud activities* associadas que precisam ser monitoradas. Assim, foi proposto e desenvolvido o SciMultaneous (Costa *et al.* 2011), um serviço que implementa um conjunto de heurísticas para re-execução de atividades de *workflows*, inspiradas nas estratégias de controle presentes em SGBD. O SciMultaneous é um trabalho em conjunto com o aluno de mestrado Flavio da Silva Costa, sob orientação da Prof.ª Marta Lima de Queirós Mattoso.

Execuções de *workflows* reais (Ocaña *et al.* 2011b, 2011a, Oliveira *et al.* 2011a) evidenciam a necessidade de controle da execução das atividades do *workflow* em nuvens computacionais, pois é indesejável ter que reiniciar uma execução que já dura dias por conta da falha em *cloud activities* de uma atividade. Em especial, em ambientes de nuvem com larga escala de processadores, não encontramos mecanismos que monitorem a execução das tarefas (e consequentemente das atividades a elas associadas) e que sejam capazes de tomar decisões para contornar problemas na execução. O SciMultaneous opera no nível de tarefa, avaliando o repositório de proveniência (que possui dados em tempo real) em busca de *cloud activities* que tenham sido executadas com erro. Tais *cloud activities* são então re-executadas pelo SciMultaneous. O SciMultaneous adota duas heurísticas básicas:

1. Redundância de *cloud activities*: mais de uma cópia da *cloud activity* é despachada de forma a existir um *backup* em caso de falha;
2. Monitoramento contínuo: por meio de consultas ao banco de proveniência, o SciMultaneous identifica tarefas e atividades que falharam e toma medidas corretivas.

A heurística de redundância de *cloud activities* tem como objetivo executar com redundância as *cloud activities* consideradas críticas, estando assim mais preparada para uma recuperação no caso de falha, como acontece no Hadoop (Hadoop 2012). Os dados de proveniência presentes no repositório de proveniência do SciCumulus são utilizados para o ranqueamento das *cloud activities*. Este ranqueamento tem como objetivo indicar a ordem de criticidade de cada *cloud activity*. Esta primeira heurística então, busca nos princípios já consolidados em SGBD sua inspiração (Elmasri e Navathe 2006). As atividades classificadas como críticas terão sua execução replicada em mais de uma máquina virtual, seguindo a mesma ideia de um *backup* de banco de dados. No caso da falha de processamento na máquina virtual original, a execução do *workflow* seria desviada para a máquina virtual redundante, sem perda significativa de desempenho.

A heurística do monitoramento contínuo das *cloud activities* determina que, no caso de falha, *cloud activities* sejam direcionadas para outra máquina virtual do mesmo provedor de nuvem ou até mesmo para outra nuvem gerida por um provedor diferente. Esse redirecionamento não é trivial, pois requer buscar uma máquina virtual que já possua os programas que a *cloud activity* irá invocar previamente instalados. A heurística funciona da seguinte forma: primeiro tenta-se executar a atividade em questão na mesma nuvem e na mesma máquina virtual, pois a falha ocorrida pode representar algum problema pontual no ambiente, que em uma segunda tentativa não ocorrerá. Em seguida, se o erro persistir, é iniciada a execução em outra máquina virtual da nuvem. Por fim, se ainda assim não obtiver sucesso, é iniciada a execução em outro provedor de nuvem, com maior poder de processamento por máquina virtual (ou mais confiável). Esta heurística busca aproveitar a localidade dos dados, da primeira nuvem, pois se a *cloud activity* em questão for dependente de dados, estes deverão ser migrados entre as nuvens, o que demandará um custo. Esta heurística se assemelha à re-execução de *log* de transação em banco de dados (Elmasri e Navathe 2006). Em ambas heurísticas tomamos como premissa que o cientista dispõe de duas ou mais nuvens computacionais para que o SciCumulus possa configurar o ambiente de execução paralela de atividades

do *workflow*. Essa premissa não é problemática, pois existem dezenas de provedores disponíveis, tanto no Brasil quanto no exterior.

5.4 SciLightning

Uma necessidade fundamental que os cientistas têm em relação aos *workflows* científicos é a questão do monitoramento, para que a qualquer momento eles possam ter conhecimento do estado da execução. Entretanto, quando lidamos com *workflows* científicos executados em paralelo e em ambientes distribuídos como nuvens, a tarefa não é trivial de ser desempenhada. Não é nada simples monitorar a execução de *workflows* científicos nestes ambientes (Cruz *et al.* 2008a) e permitir o acompanhamento e a análise parcial dos resultados em pontos predeterminados (Mattoso *et al.* 2010c). Para oferecer tal serviço, foi desenvolvido o SciLightning. O SciLightning é um trabalho em conjunto com o aluno de mestrado Julliano Trindade Pintas, sob orientação da Prof.ª Marta Lima de Queirós Mattoso.

O objetivo do SciLightning é prover um serviço de monitoramento de execução de *workflows* científicos em paralelo, que notifique o cientista sobre eventos que lhe sejam importantes por meio de dispositivos móveis e redes sociais (*e.g. Facebook* e *Twitter*) e que seja independente da solução de execução utilizada. Desta forma, o serviço SciLightning facilita o acompanhamento e a análise parcial dos resultados. Apesar de o SciLightning ser independente de solução para execução em paralelo, focamos, nesta tese, no desenvolvimento da interface com o SciCumulus e futuramente o integraremos ao motor de execução Chiron (Ogasawara *et al.* 2011).

O SciLightning é dividido em três componentes principais: o SciLightning Monitor, o SciLightning Social e o SciLightning Móvel. O SciLightning Monitor é uma aplicação desenvolvida em Java para *Web* que tem como principal objetivo realizar o monitoramento da execução de *workflows* através da realização de consultas periódicas ao repositório de proveniência do SciCumulus. Isso só é possível porque, diferentemente dos SGWfC existentes, o SciCumulus gera os dados de proveniência de forma *on-line*. A relação de quais *workflows* serão monitorados, assim como a definição de parâmetros de monitoramento para cada *workflow*, são definidos através da camada cliente do SciCumulus. Além desta funcionalidade principal, o SciLightning Monitor também permite o registro dos dispositivos móveis e envia notificações para serem tratadas pelos componentes SciLightning Social e SciLightning Móvel.

O SciLightning Social é uma conta de aplicação do *Twitter* (O'Reilly e Milstein 2011, Russell 2011) configurada para enviar notificações sobre a execução de *workflows* aos cientistas através do serviço de mensagens diretas. No *Twitter*, mensagens diretas são mensagens curtas e não públicas enviadas entre dois usuários (contas do *Twitter*). A biblioteca Twitter4J (Twitter4j 2011) foi utilizada com o objetivo de facilitar a comunicação do SciLightning com o *Twitter*, já que esta biblioteca encapsula todas as requisições HTTP e tratamentos de resposta necessários para a comunicação de uma aplicação Java qualquer com o mesmo. Para autenticar a aplicação junto aos servidores do *Twitter*, foi utilizado o mecanismo de autenticação OAuth (OAuth 2010), que é um padrão aberto no qual não é necessário o compartilhamento das credenciais (tipicamente, um nome de usuário e senha), mas sim *tokens* fornecidos pela aplicação autenticadora (IETF, do inglês *Internet Engineering Task Force*). Para receber notificações, é necessário que o cientista esteja seguindo o usuário de *Twitter* responsável por enviar as notificações, que nesta tese é representado pelo nome @SciLightning.

O SciLightning Móvel é uma aplicação Android (Butler 2011, Mednieks *et al.* 2011) que recebe e armazena as notificações derivadas do monitoramento do SciLightning Monitor. Para envio das notificações foi utilizado o serviço *Cloud To Device Message* (C2DM), que consiste de um mecanismo simples e leve para envio de mensagens para aplicações em dispositivos móveis (desenvolvido pela Google). A principal vantagem de utilizar este serviço é que não é necessário que a aplicação do dispositivo móvel realize requisições periódicas (*polling*) ao servidor para saber se existem novas mensagens. Não é necessário nem que a aplicação do dispositivo móvel esteja executando em plano de fundo, para que as mensagens sejam recebidas. Neste caso, o sistema operacional inicializa a aplicação quando uma mensagem chega. Deste modo, promove uma economia considerável de energia e banda de internet. Para registrar a aplicação SciLightning Móvel, o cientista necessita apenas de seu nome de usuário e senha. Porém, internamente, o registro ocorre em dois passos:

i. O SciLightning Móvel solicita um *token* de identificação aos servidores do serviço C2DM.
ii. Se o *token* for recebido com sucesso, as informações de nome de usuário, senha e *token* de identificação são submetidas para o SciLightning Monitor que só finalizará o registro após a validação das credenciais do cientista.

O SciLightning Monitor possui o *token* de identificação de cada dispositivo móvel registrado. Cada *token* de identificação está relacionado a apenas um cientista, que por sua vez, poderá possuir vários dispositivos registrados. Quando uma notificação possuir um determinado cientista como destinatário, esta notificação será enviada para todos seus dispositivos móveis registrados para este cientista. Ao receber uma notificação, o SciLightning Móvel cria uma mensagem na barra de estado do dispositivo (Figura 1). Caso o cientista queira analisar detalhes da notificação, ele deve selecionar a mensagem na barra de estado, o que faz com que a tela da aplicação SciLightning Móvel liste todas as notificações recebidas até o momento. No Capítulo 7 desta tese apresentamos uma análise sobre a utilização do SciLightning na execução do estudo de caso.

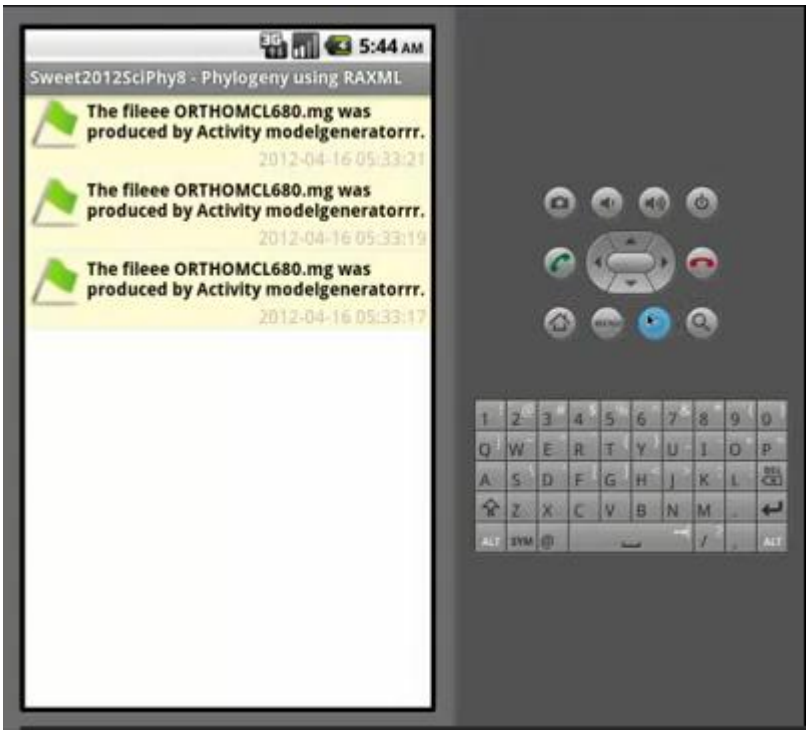


Figura 12 Barra de estado do SciLightning móvel

5.5 BioSciCumulus: O SciCumulus como Serviço

Embora o SciCumulus represente um passo à frente no que se refere a tecnologias para apoio à execução paralela de *workflows* científicos, o cientista ainda necessita possuir conhecimentos de ciência da computação para instalá-lo, utilizá-lo e configurá-lo corretamente. Essa dificuldade se deve à necessidade de configurar algumas dependências de *software* e configuração do ambiente de nuvem. Esta tarefa pode ser muito complexa para alguém não especialista em computação.

Uma solução para este problema é disponibilizar o SciCumulus como um serviço seguindo o modelo de negócio SaaS (apresentado no Capítulo 2 desta tese), altamente adotado pela comunidade de computação em nuvem. Seguindo esse modelo, os componentes do SciCumulus e seus dados associados se encontram previamente implantados e hospedados na nuvem e os mesmos são acessados pelos cientistas utilizando um navegador *Web* comum através da internet.

Como projeto piloto da disponibilização do SciCumulus como serviço, foi desenvolvido o BioSciCumulus, um serviço *Web* gratuito, baseado na *Web* para gerenciar a execução paralela de um conjunto de *workflows* científicos biológicos (estes *workflows* são apresentados no Capítulo 6 desta tese). O BioSciCumulus foi construído baseado no SciCumulus e seus serviços associados e implantado no Amazon EC2. Utilizando o BioSciCumulus, o biólogo/bioinformata pode criar novos experimentos e analisar dados de proveniência em tempo de execução, enquanto colabora com outros cientistas em tempo real. O BioSciCumulus disponibiliza os seguintes *workflows* pré-configurados de bioinformática para serem executados em paralelo em uma nuvem:

i. O SciHmm (Ocaña *et al.* 2011a), que executa uma análise de genômica comparativa através da realização de um teste de validação cruzada para ajudar na decisão de qual programa de alinhamento utilizar em futuras análises filogenéticas / filogenômicas;
ii. O SciPhy (Ocaña *et al.* 2011b), que realiza a análise filogenética e;
iii. O SciPhylomics (Oliveira et al. 2012), que constrói super-alinhamentos (concatenando os alinhamentos individuais) para realizar a análise filogenômica.

O BioSciCumulus pode ser livremente acessado a partir de área de trabalho dos cientistas e oferece a capacidade de executar uma série de *workflows* de análise evolutiva em paralelo em recursos distribuídos (máquinas virtuais). O BioSciCumulus pode ser invocado usando um navegador da *Web* genérico (Internet Explorer, Mozilla Firefox, Google Chrome, Opera, Safari) que se encontram disponíveis para a maioria das plataformas (Windows, Unix / Linux, MacOS), reforçando assim o recurso de independência de plataforma. A versão para *download* do BioSciCumulus está disponível em http://bioscicumulus.sourceforge.net/. Futuras versões do BioSciCumulus incluirão aplicações de genômica evolutiva e estrutural, a fim de facilitar a adoção de *workflows* mais complexos.

5.6 Representação de *Workflows*

Toda a representação das atividades e dos dados no SciCumulus é baseada em um modelo algébrico para *workflows* proposto por Ogasawara *et al.* (2011) para realizar a execução paralela de suas *cloud activities*. Segundo este modelo algébrico, os dados de um *workflow* a ser executado no SciCumulus são representados como relações (similares as do modelo relacional) e todas as atividades do *workflow* são regidas por operações algébricas. Existem diversos tipos de operadores algébricos propostos (*Map*, *SplitMap*, *Reduce*, *Filter*, *SRQuery* e *MRQuery*), porém a explicação destas operações foge ao escopo desta tese, uma vez que a intenção não é explorar a álgebra e nem possíveis otimizações. Limitamo-nos apenas a explicar o operador *Map* (que é a utilizada nos experimentos desta tese) que tem como objetivo a execução direta de um programa que consome uma tupla de entrada e produz uma tupla de saída, *i.e.* consome dados de entrada e produz dados de saída. Cada tupla de entrada é composta por campos que representam na prática os parâmetros que estão sendo explorados. Os valores em cada tupla representam os valores dos parâmetros utilizados em cada execução.

O *workflow* é então especificado por meio de um arquivo XML conforme o esquema XPDL (WfMC 2009). A especificação segue este padrão para que possamos integrar futuramente o SciCumulus com outras abordagens existentes, como a GExpLine (Ogasawara *et al.* 2009b, Oliveira *et al.* 2010d) ou o ProvManager (Marinho *et al.* 2011).

A Figura 13 apresenta o XML *Schema* (Fallsite e Walmsley 2004) de um documento XML de definição de um *workflow* no SciCumulus. Cada documento XML de definição de um *workflow* no SciCumulus possui um elemento raiz *SciCumulus* e o elemento *database* que especifica a base de dados utilizada pelo SciCumulus para registro de dados de proveniência e para acesso ao esquema global. Como elemento filho de *SciCumulus*, temos o elemento *SciCumulusWorkflow* que tem como objetivo descrever o *workflow* que estamos querendo representar. Cada *workflow* no SciCumulus é descrito pelos atributos: *tag* que define um rótulo para o *workflow*, *description* que representa a descrição feita pelo cientista sobre o *workflow* representado, servindo como anotação para o *workflow*, *exectag* que serve para representar uma rodada do *workflow*, *i.e.* um ensaio do *workflow* (do inglês *trial*), e finalmente, o atributo *expdir* que define o

diretório nas diversas máquinas virtuais onde se encontram os dados do experimento que está sendo executado.

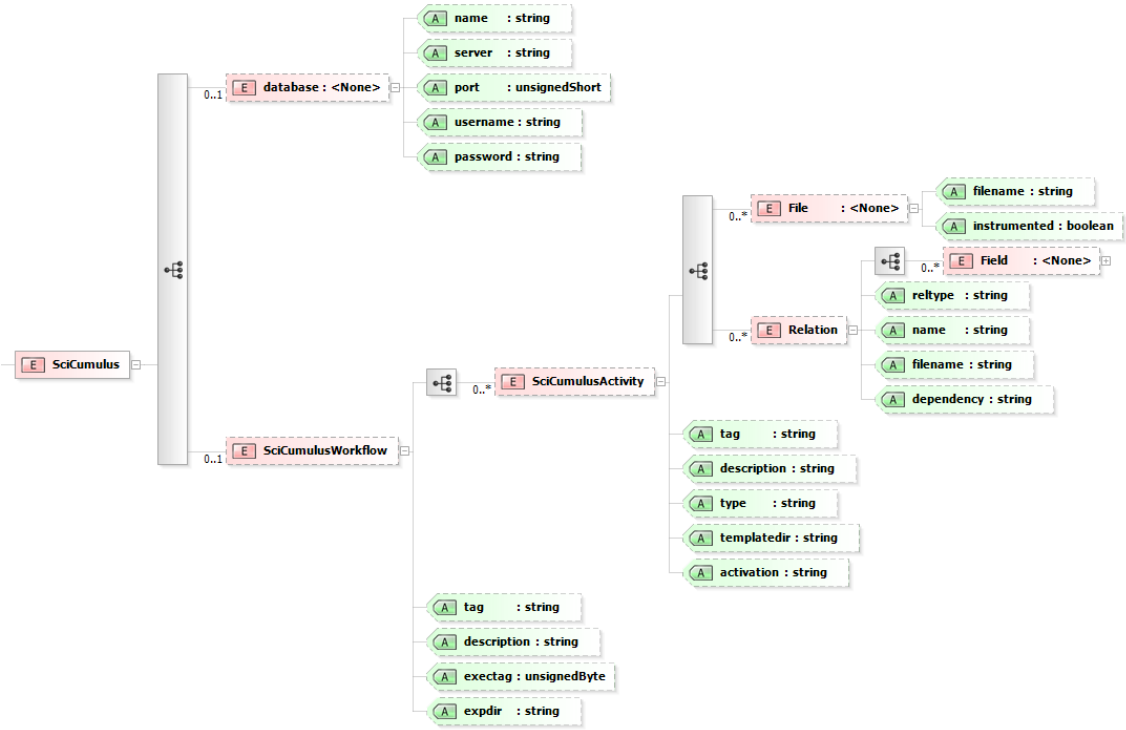


Figura 13 Esquema XML do arquivo de definição do *workflow* no SciCumulus

O *workflow* serve também como elemento agrupador no XML, uma vez que ele é composto por diversas atividades (*SciCumulusActivity)*. Similarmente ao *workflow*, cada atividade é identificada por rótulos (atributo *tag*) e possui descrições (atributo *description*). Cada atividade possui um atributo *type* que define o tipo de operação algébrica que está sendo executada. No contexto desta tese e dos experimentos que serão apresentados, apenas atividades do tipo *Map* foram utilizadas. O atributo *activation* define qual será a chamada de linha de comando do programa que representa a atividade. O valor desse atributo é definido por uma linha de comando com *tags* de instrumentação (conforme explicado na Subseção 5.9.4) ou por um arquivo que contém a chamada para o programa. Essa segunda opção é útil quando um determinado programa necessita de mais de um comando para ser executado. Este arquivo deve estar no caminho definido no atributo *templatedir*. O elemento *Relation* denota a especificação da relação de entrada da atividade (qual arquivo explicita os dados e parâmetros a serem consumidos). Cada relação é composta de campos (atributo *Field*) que definem os dados que são trafegados entre atividades (em geral os valores de parâmetros que são explorados). O elemento *File* representa os arquivos que devem ser

manipulados durante a execução do *workflow*, inclusive os arquivos que devem ser instrumentados, como por exemplo, um *script* que tenha sido definido no atributo *activation*. A Figura 14 apresenta um exemplo de arquivo XML de especificação de um *workflow* no SciCumulus.

```
<?xml version="1.0" standalone="no"?>
<SciCumulus>
  <database name="scicumulus_adaptive" server="mp4-4.dyndns.info"
     port="5432" username="scicumulus" password="**********" />
  <SciCumulus*Workflow* tag="filogenia" description="This is a test using anflex."
    exectag="Experimento Kary/200 organismos - Adaptive" expdir="/root/exp">
    <SciCumulus*Activity* tag="mafft" description="mafft" type="MAP"
          templatedir="/root/exp/template_mafft" activation="experiment.cmd">
      <Relation reltype="Input" name="A" filename="parameter.txt" />
      <Field name="NAME" type="string" />
      <Field name="FASTA_FILE" type="string" />
      <Relation reltype="Output" name="C" filename="output_mafft.txt" />
      <Field name="NAME" type="string" />
      <Field name="NUM_SEQ" type="string" />
      <Field name="FASTA_FILE" type="string" />
      <File filename="experiment.cmd" instrumented="true" />
    </SciCumulus*Activity*>
  </SciCumulus*Workflow*>
</SciCumulus>
```

Figura 14 Especificação do *workflow* no SciCumulus que executa uma atividade

O *workflow* da Figura 14 contém apenas uma atividade regida pelo operador algébrico *Map*. A atividade é identificada pelo rótulo *mafft*. A atividade *mafft* é executada através do script *experiment.cmd*, que consome os atributos (*Field*) NAME e FASTA_FILE de sua relação (*Relation*) de entrada. A atividade *mafft* produz os atributos NAME, NUM_SEQ e FASTA_FILE na sua relação de saída. O arquivo de entrada (*File*) *parameter.txt* contém os valores dos campos explicitados no XML. Cada linha deste arquivo gerará uma nova *cloud activity* para ser executada. Um exemplo deste arquivo contendo a relação de entrada da atividade *mafft* é apresentado na Figura 15. Cada linha do arquivo define um conjunto de valores de parâmetros que serão associados a uma *cloud activity* específica.

```
NAME;FASTA_FILE
G1;ORTHOMCL2033
G2;ORTHOMCL1895
G3;ORTHOMCL2034
G4;ORTHOMCL1896
G5;ORTHOMCL2035
G6;ORTHOMCL1897
G7;ORTHOMCL2036
```

```
G8;ORTHOMCL1898
G9;ORTHOMCL2037
G10;ORTHOMCL1899
```

Figura 15 Arquivo contendo a relação de entrada para a atividade *mafft*

5.7 Modelo de Proveniência do SciCumulus

O SciCumulus foi projetado e implementado para operar em nuvens de computadores, assim como seu modelo de proveniência. Ele representa todas as informações sobre o experimento que está sendo executado e sobre o ambiente de nuvem propriamente dito. Este modelo de proveniência é baseado no *Open Provenance Model* (OPM) (Moreau *et al.* 2008a, 2008b). O OPM é uma recomendação aberta e focada no ponto de vista interoperabilidade entre os dados de proveniência dos diversos SGWfC, mas também com respeito à comunidade de seus colaboradores, revisores e usuários. A ideia principal do OPM é representar as relações causais entre processos, agentes, artefatos e papéis envolvidos em uma execução de *workflow* científico, seja ele paralelo ou não. Diferentemente do que possa parecer, o OPM não é diretamente instanciável em um banco de dados, mas é uma representação padrão de proveniência de dados para a maioria dos SGWfC (Altintas *et al.* 2004b, Callahan *et al.* 2006, Deelman *et al.* 2007, Fahringer *et al.* 2005, Hull *et al.* 2006b, Taylor *et al.* 2007c, Zhao *et al.* 2007).

A Figura 16 apresenta o modelo conceitual projetado para o repositório de proveniência do SciCumulus enquanto que a Figura 17 apresenta o esquema lógico criado a partir do modelo conceitual. O modelo conceitual não apresenta os atributos das entidades para que o mesmo não fique poluído e atrapalhe o entendimento. Todas as informações relacionadas a execuções anteriores de *workflows* científicos que foram utilizadas no modelo de custo e no algoritmo de escalonamento proposto nessa tese são recuperadas a partir do repositório de proveniência do SciCumulus. O modelo conceitual é apresentado por meio de um diagrama ER e o modelo lógico é representado por meio de um diagrama de classes da *Unified Modeling Language* (UML) (Bezerra 2007, Fowler 2003) e foi modelado seguindo as recomendações do OPM e com base nos requisitos levantados com os cientistas. Parte das informações presentes (aquelas relativas a tempos de execução e estado da execução) no modelo de proveniência são capturadas automaticamente pelos componentes do SciCumulus durante a execução do *workflow*, enquanto o agente autônomo capta as informações sobre o ambiente de nuvem.

O modelo de proveniência do SciCumulus (tanto o conceitual quanto o lógico) é composto por três partes principais: (i) Elementos que representam os *workflows* executados na nuvem e os artefatos consumidos e produzidos pela execução do *workflow*; (ii) Elementos que representam informações sobre o ambiente de nuvem e; (iii) Elementos que representam informações de monitoramento do SciCumulus (vide Subseção 5.4 para mais detalhes).

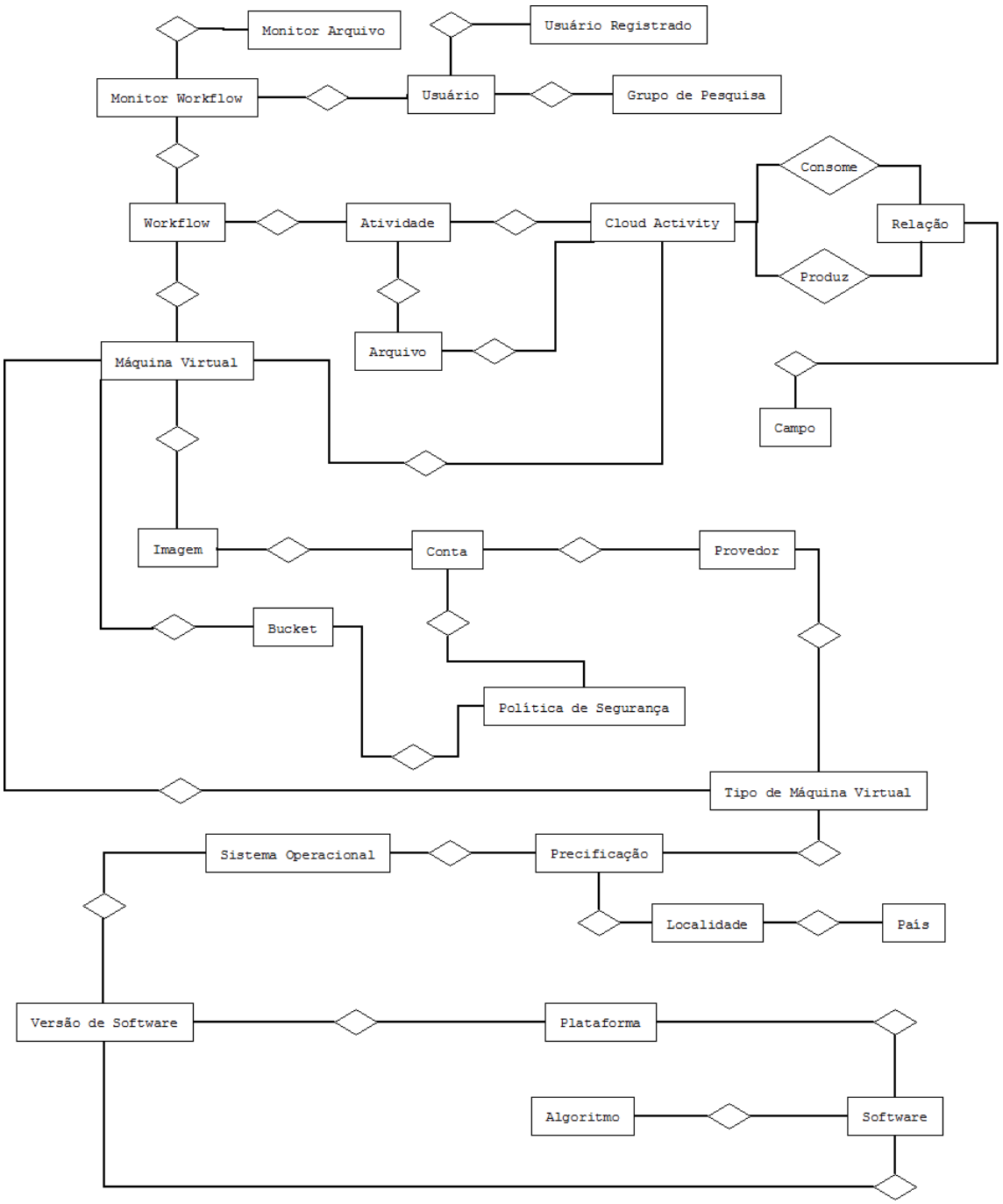


Figura 16 Modelo conceitual do repositório de proveniência do SciCumulus

Inicialmente detalharemos as classes que representam informações relativas ao *workflow* executado, ou seja, sua estrutura. A classe *Workflow* possui os atributos *tag*, que servem para classificar um determinado experimento (filogenia, prospecção de petróleo em aguas profundas, etc.) e *tagExecução*, que identifica unicamente uma tentativa de execução do experimento (do inglês, *trial*). Quando um cientista solicita a execução de uma tentativa já existente, o SciCumulus seleciona apenas as atividades não executadas para "continuar" a execução anterior. O atributo *expDir* aponta para o diretório em que se encontram os dados do experimento, no caso o disco compartilhado na nuvem. *Início* e *fim* indicam os tempos de início e fim da execução do *workflow*. O atributo *descrição* possibilita ao cientista descrever o experimento que esta sendo executado para fins de anotação.

A classe *Atividade* possui um atributo *tag* que identifica a atividade de um determinado *workflow*. O atributo *estado* representa os estados que a atividade pode assumir e seguem os valores apresentados na FSM descrita no Capítulo 2 desta tese. O atributo *comando* é usado para instrumentação do comando que será utilizado para invocar o programa associado a esta atividade. O atributo *extrator* é usado para representar o programa que processa os dados do resultado da execução das *cloud activities* relacionadas com a atividade para transformá-los em uma nova saída. O extrator pode ser utilizado para recuperar dados que estejam contidos em arquivos binários ou para validar algum arquivo que foi produzido na execução das *cloud activities*. Os atributos *início* e *fim* contêm, respectivamente, a informação do tempo de início da execução da primeira *cloud activity* desta atividade e a informação do tempo de término da execução da última *cloud activity* desta atividade.

A classe *CloudActivity* possui os atributos *número* e *comando* que representam, respectivamente, o identificador da *cloud activity*, bem como o número de identificação da linha de execução. O atributo *workspace* indica o local em que a *cloud activity* consumiu e produziu os seus dados. O atributo *estado* e *log* são importantes para armazenamento de proveniência referente aos dados de execução da *cloud activity*. Os atributos *início* e *fim* contêm, respectivamente, a informação do início e término da execução desta *cloud activity*.

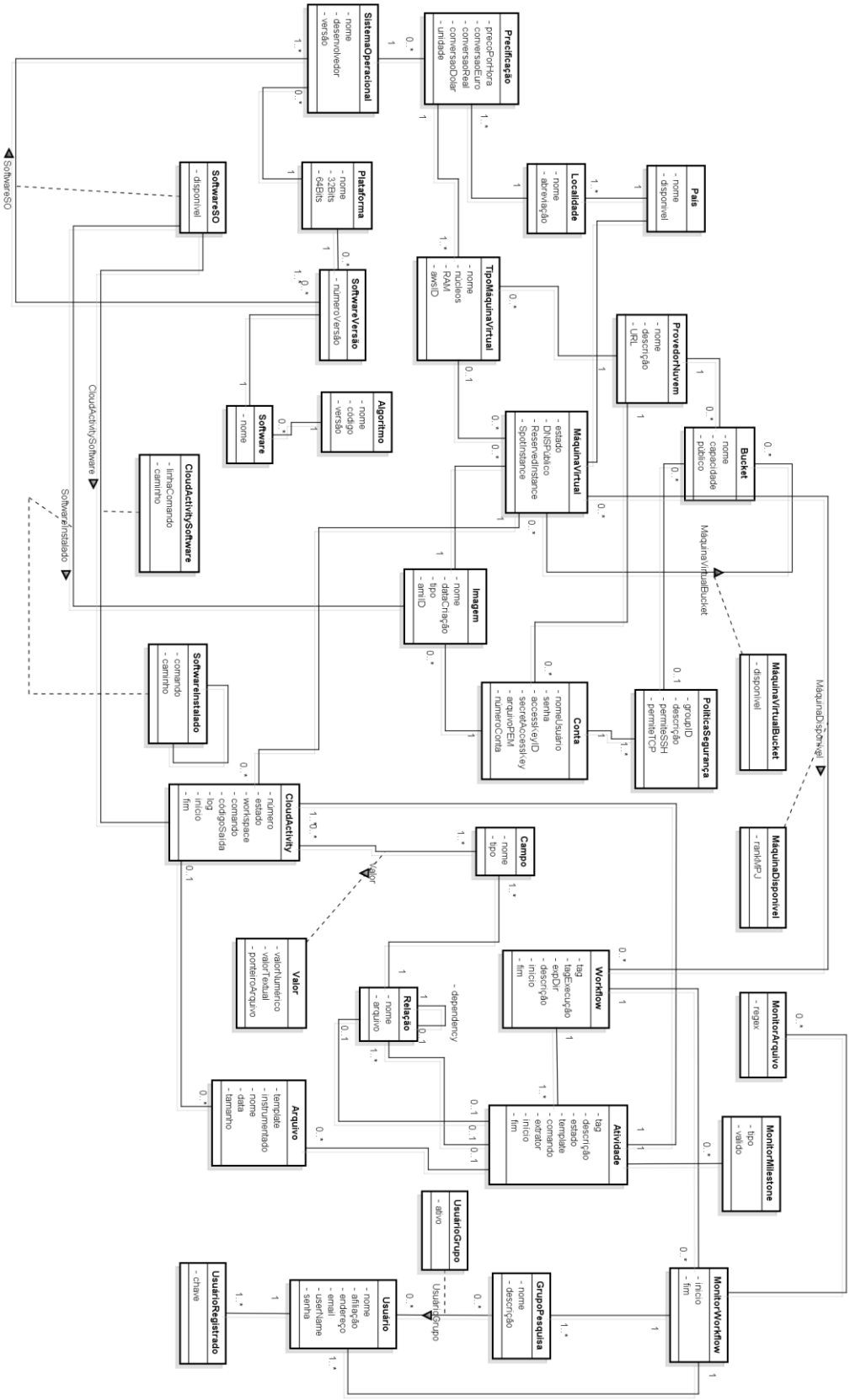


Figura 17 Esquema lógico de proveniência do SciCumulus

Uma vez que o SciCumulus está baseado em um modelo algébrico proposto por Ogasawara *et al.* (2011), devemos armazenar os elementos deste modelo, ou seja, relações, campos e valores de tuplas. A classe *Relação* contém atributos *nome* e *arquivo*. A classe *Campo* tem o *nome* e *tipo* dos campos que fazem parte de uma relação, similar a uma tabela em um banco de dados relacional. A classe *Valor* indica os valores de uma tupla consumida ou produzida durante a execução de uma *cloud activity*. Os atributos *valorNumerico*, *valorTextual* e *ponteiroArquivo* são preenchidos de acordo com o tipo de atributo. A classe *Arquivo* é usada para coletar as informações de todos os arquivos consumidos/produzidos durante a execução de uma *cloud activity*, ou seja, *nome*, *data* e *tamanho*.

Serão detalhadas agora as classes que representam os dados coletados do ambiente de nuvem através do agente autônomo. As classes *País* e *Localidade* representam os dados referentes aos países e locais nesses países onde as máquinas virtuais e os *buckets* são criados e instanciados. Dependendo do local onde as máquinas virtuais são criadas, existe uma *Precificação* associada, e esses dados são representados pela classe de mesmo nome. Nesta classe, o preço por hora, a unidade monetária de cobrança, a conversão para Euro (**€**), conversão para Dólar (US$) e para Real (R$) são representadas nos atributos *preçoPorHora*, *unidade*, *conversaoEuro*, *conversaoDolar* e *conversaoReal* respectivamente.

A precificação também depende do *Sistema Operacional* utilizado. A classe de mesmo nome representa o *nome*, a *versão*, e o *desenvolvedor* respectivamente. Cada sistema operacional é baseado em uma arquitetura, que pode ser de 32 ou 64 bits. O mesmo pode-se dizer dos *softwares* que são utilizados pelas atividades dos *workflows*. A classe *Software* representa os programas utilizados nos experimentos onde casa *software* possui um *nome* e está baseado em um *Algoritmo*. Os *software*s também possuem versões que estão associadas a uma determinada arquitetura seja ela de 32 ou 64 bits. Esta informação é representada pela classe *SoftwareVersão*.

A classe *ProvedorNuvem* representa os provedores de nuvem que podem ser utilizados e os atributos *nome*, *descrição* e *URL* de acesso representam os dados de cada provedor. Um provedor disponibiliza vários tipos de máquinas virtuais que estão representadas na classe *TipoMáquinaVirtual*. Cada tipo de máquina possui um *nome*, um *ID* (no caso da Amazon EC2), a quantidade de *núcleos* (do inglês, *cores*) e a quantidade de memória *RAM*. Baseados nesses tipos existentes, várias máquinas virtuais

podem ser instanciadas sem nenhum *software* instalado (somente o sistema operacional) ou podem ser baseadas em uma imagem pré-configurada. A classe *MáquinaVirtual* representa os dados das máquinas virtuais e contém os atributos *estado*, *DNSPúblico* (endereço de acesso à máquina virtual), *ReservedInstance* e *SpotInstance* sendo que os dois últimos atributos indicam se uma máquina virtual é uma instância reservada ou *spot* (segundo os critérios da Amazon). A classe *Imagem* representa os dados das imagens que foram configuradas e que estão disponíveis para uso. Cada imagem possui um *ID*, um *nome* e um *tipo*. Os dados que são consumidos pelas *cloud activities* estão armazenados em um disco compartilhado na nuvem denominado *Bucket*. A classe de mesmo nome possui os atributos nome e capacidade para representar os *buckets* utilizados.

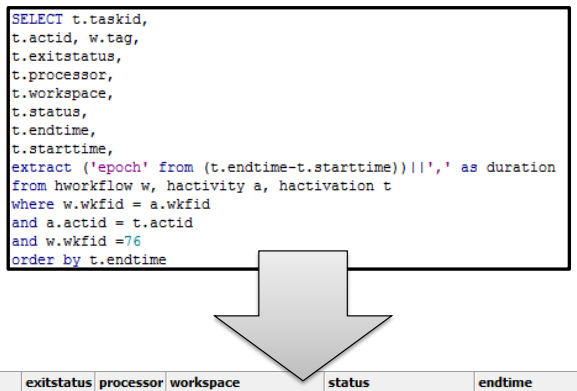
A classe *MonitorWorkflow* armazena os monitoramentos realizados (ver detalhes na Seção 5.4), ou seja, quais *workflows* serão monitorados e quais usuários ou grupos serão notificados. Para cada monitoramento podem ser definidos parâmetros de notificação, que tornam possível o envio de notificações, como por exemplo, quando as atividades do *workflow* gerarem um arquivo com nome que atenda uma das expressões regulares pré-definidas (*i.e. regex*) ou quando o *workflow* atingir certos pontos de execução pré-definidos (*i.e. milestones*). Esses dois parâmetros de monitoramento são representados pelas classes *Arquivo* e *Milestone*, respectivamente. Tanto a relação de *workflows* que devem ser monitorados, quanto os parâmetros de monitoramento são preenchidos pelo SciCumulus no início da execução do *workflow*.

Informações sobre usuário, como *login* e *senha* e conta do *twitter* utilizados para o registro de monitoramento são armazenadas na classe *Usuário*. Estes usuários podem ser organizados em grupos através da classe *UsuárioGrupo*, o que permite as notificações sejam destinadas a todos os integrantes do grupo. A classe *ÚltimoMonitoramento* é uma classe auxiliar que armazena a data do último monitoramento realizado, que tem como objetivo otimizar as consultas de monitoramento. Por último, a classe *ErroMonitoramento* é uma classe que armazena os códigos de erros e suas respectivas mensagens, que vão ser utilizadas para tornar os textos de notificações de erros mais úteis para o cientista.

As classes *Relação, Campo, Valor, MáquinaVirtual* e *Arquivo* correspondem a artefatos segundo o OPM. Isto foi definido uma vez que tais classes designam estruturas digitais em sistemas computacionais (parâmetros, banco de dados, arquivos e

instâncias) sobre as quais se deseja controlar a proveniência. As classes *Atividade* e *CloudActivity* são mapeadas como processos OPM. Um processo OPM representa uma ou mais ações que consomem ou produzem artefatos. A classe *ProvedorNuvem* e *Usuário* representam um agente OPM.

As classes *Atividade* e *CloudActivity* são mapeadas como processos OPM. Um processo OPM representa uma ou mais ações que consomem ou produzem artefatos. A classe *ProvedorNuvem* e *Usuário* representam um agente OPM. Todos os dados de proveniência do SciCumulus são gerados de forma *on-line*, ou seja, durante o curso de execução do *workflow*. Esta proveniência *on-line* permite que o SciCumulus utilize os dados proveniência em tempo de execução para escalonamento e monitoramento. Nas abordagens existentes, tais como o VisTrails, os dados de proveniência são gerados somente no final da execução do *workflow*. Para exemplificar o esquema de proveniência do SciCumulus, a Figura 18 apresenta uma consulta sobre a base de proveniência. Esta consulta indica a duração em milissegundos de cada *cloud activity* para um determinado *workflow*.



```
SELECT t.taskid,
t.actid, w.tag,
t.exitstatus,
t.processor,
t.workspace,
t.status,
t.endtime,
t.starttime,
extract ('epoch' from (t.endtime-t.starttime))||',' as duration
from hworkflow w, hactivity a, hactivation t
where w.wkfid = a.wkfid
and a.actid = t.actid
and w.wkfid =76
order by t.endtime
```

taskid integer	actid integer	tag character varying(200)	exitstatus integer	processor integer	workspace character varying(150)	status character varying(25)	endtime timestamp with time zone	starttime timestamp with time zone	duration text
182471	475	SciHmm-SciPhy-30009	0	7	/root/exp/hmmsearchv4/1/	FINISHED	2011-11-25 17:01:28.212-03	2011-11-25 17:01:22.439-03	5.773,
182474	475	SciHmm-SciPhy-30009	0	9	/root/exp/hmmsearchv4/4/	FINISHED	2011-11-25 17:01:28.625-03	2011-11-25 17:01:22.55-03	6.075,
182473	475	SciHmm-SciPhy-30009	0	2	/root/exp/hmmsearchv4/3/	FINISHED	2011-11-25 17:01:29.003-03	2011-11-25 17:01:22.371-03	6.632,
182475	475	SciHmm-SciPhy-30009	0	3	/root/exp/hmmsearchv4/5/	FINISHED	2011-11-25 17:01:29.46-03	2011-11-25 17:01:22.472-03	6.988,
182472	475	SciHmm-SciPhy-30009	0	1	/root/exp/hmmsearchv4/2/	FINISHED	2011-11-25 17:01:33.413-03	2011-11-25 17:01:22.37-03	11.043,
182480	476	SciHmm-SciPhy-30009	0	9	/root/exp/mafft/5/	FINISHED	2011-11-25 17:07:08.861-03	2011-11-25 17:01:35.56-03	333.301,
182478	476	SciHmm-SciPhy-30009	0	3	/root/exp/mafft/3/	FINISHED	2011-11-25 17:07:26.232-03	2011-11-25 17:01:35.375-03	350.857,
182476	476	SciHmm-SciPhy-30009	0	1	/root/exp/mafft/1/	FINISHED	2011-11-25 17:07:30.683-03	2011-11-25 17:01:35.455-03	355.228,
182477	476	SciHmm-SciPhy-30009	0	2	/root/exp/mafft/2/	FINISHED	2011-11-25 17:07:50.067-03	2011-11-25 17:01:35.429-03	374.638,
182479	476	SciHmm-SciPhy-30009	0	7	/root/exp/mafft/4/	FINISHED	2011-11-25 17:07:51.483-03	2011-11-25 17:01:35.554-03	375.929,
182484	477	SciHmm-SciPhy-30009	0	5	/root/exp/readseq/4/	FINISHED	2011-11-25 17:07:55.184-03	2011-11-25 17:07:53.202-03	1.982,
182482	477	SciHmm-SciPhy-30009	0	3	/root/exp/readseq/2/	FINISHED	2011-11-25 17:07:55.225-03	2011-11-25 17:07:53.056-03	2.169,
182485	477	SciHmm-SciPhy-30009	0	1	/root/exp/readseq/5/	FINISHED	2011-11-25 17:07:56.037-03	2011-11-25 17:07:53.139-03	2.898,
182483	477	SciHmm-SciPhy-30009	0	9	/root/exp/readseq/3/	FINISHED	2011-11-25 17:07:56.193-03	2011-11-25 17:07:53.127-03	3.066,
182481	477	SciHmm-SciPhy-30009	0	7	/root/exp/readseq/1/	FINISHED	2011-11-25 17:07:57.132-03	2011-11-25 17:07:53.114-03	4.018,
182489	478	SciHmm-SciPhy-30009	0	5	/root/exp/modelgenerator/4	FINISHED	2011-11-25 17:18:43.684-03	2011-11-25 17:07:58.549-03	645.135,
182488	478	SciHmm-SciPhy-30009	0	2	/root/exp/modelgenerator/3	FINISHED	2011-11-25 17:34:54.318-03	2011-11-25 17:07:58.634-03	1615.684,

Figura 18 Consulta sobre a base de proveniência do SciCumulus

5.8 Modelo de Execução do SciCumulus

Depois de definido um cenário de paralelização de uma atividade do *workflow* por meio de varredura de parâmetros, o SciCumulus pode estabelecer um conjunto de *cloud activities* associadas a esta atividade. Uma *cloud activity* consome certa combinação de valores de parâmetros e dados. Desta forma, as *cloud activities* podem ser escalonadas e despachadas dentre os núcleos computacionais das máquinas virtuais que irão processá-las na nuvem de diferentes formas. No SciCumulus, da mesma forma que em experiências anteriores com o Hydra (Silva *et al.* 2011b), o conjunto das *cloud activities* pode ser distribuído seguindo as estratégias de despacho estática e dinâmica.

Na estratégia estática, as *cloud activities* de um determinado *workflow* são pré-alocadas para cada núcleo antes de serem consumidas. Ou seja, a distribuição do número de *cloud activities* por núcleo é feita *a priori*. A Figura 19(a) representa um esquema da distribuição estática de *cloud activities*. Cada *cloud activity* da atividade *A* do *workflow* (as *cloud activities* estão representadas pelos quadrados *a*) é referente a um mesmo programa com um conjunto diferente de valores de parâmetros. No início da execução paralela, o escalonador do SciCumulus atribui para cada núcleo computacional envolvido na execução um número fixo de quatro *cloud activities* da atividade *A* para serem processadas. O escalonador apresentado na figura é o componente do SciCumulus responsável pela distribuição, conforme dito anteriormente neste capítulo. Na estratégia de despacho estática, o escalonador do SciCumulus utiliza a abordagem *gather* e *scatter* do MPJ (Carpenter *et al.* 2000, Shafi *et al.* 2010) para transmitir as *cloud activities* aos núcleos computacionais. Como o ambiente de nuvem está sujeito a flutuações durante a execução do *workflow*, esta estratégia de despacho não é recomendada e nem pode ser utilizada pela abordagem adaptativa, pois não podemos definir *a priori* um escalonamento e garantir que não haverá problemas de desempenho.

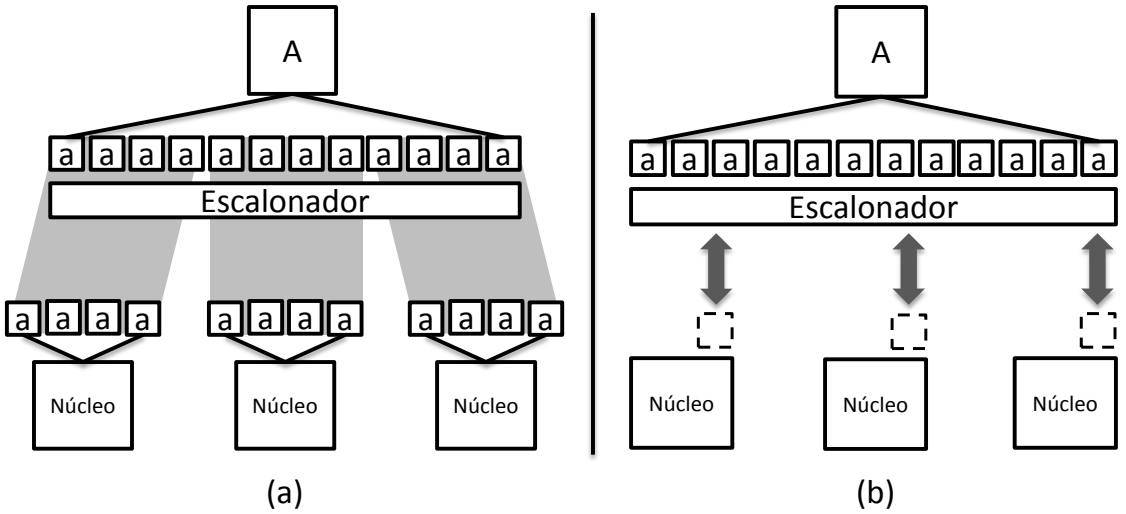


Figura 19 Estratégias de despacho de *cloud activities* no SciCumulus: estática (a) e a dinâmica (b)

Por outro lado, na estratégia de despacho dinâmica, cada *cloud activity* é alocada para um núcleo computacional conforme este esteja disponível e requisite uma *cloud activity* para processar. Ou seja, as *cloud activities* permanecem em uma fila única (também chamada de *bag*) e cada núcleo obtém e processa apenas uma *cloud activity* (ou grupo de *cloud activities*) por vez. Na Figura 19(b) é apresentado um esquema da estratégia dinâmica, onde cada núcleo requisita ao escalonador do SciCumulus uma *cloud activity* da atividade *A* por vez. Neste caso, o escalonador do SciCumulus precisa manter uma *thread* que aguarda requisições feitas pelos núcleos (*i.e. listener*) através de primitivas de comunicação do tipo *send* e *receive* do MPJ.

5.9 Detalhes de Implementação

Com o objetivo de desenvolver uma versão piloto para esta tese, implantamos a arquitetura do SciCumulus no ambiente Amazon EC2. O Amazon EC2 é um dos ambientes de computação em nuvem mais populares, e muitas aplicações científicas e comerciais estão sendo implantadas sobre ele (Armbrust *et al.* 2010, Hey *et al.* 2009, Hoffa *et al.* 2008, Matsunaga *et al.* 2008, Ocaña *et al.* 2011b, 2011a). A implantação de um mediador como o SciCumulus no ambiente Amazon EC2 não é algo trivial de ser realizado. Para tal, foram necessárias adaptações em alguns componentes da infraestrutura proposta.

A versão atual do SciCumulus foi desenvolvida utilizando o Java Versão 6 Update 15 e suas respectivas bibliotecas adicionais necessárias, como o HSQLDB (Simpson e Toussi 2002) e os *drivers* JDBC (Deitel e Deitel 2010). Os componentes da

camada de distribuição foram implementados utilizando o MPJ (Carpenter *et al.* 2000) (uma API MPI-*like* para Java). O MPJ adota como primitiva a bem conhecida técnica de paralelismo aninhado (Shafi *et al.* 2009), que mistura a distribuição entre processos por troca de mensagens em MPI e execução em vários núcleos das máquinas virtuais por meio de *threads*. Assim, o paralelismo aninhado do Java com MPJ é bem adequado, uma vez que há uma tendência cada vez maior de que as máquinas virtuais possuam vários núcleos (existem máquinas na Amazon que possuem 8 núcleos). Além disso, a maioria dos problemas de varredura de parâmetros é do tipo embaraçosamente paralelo (do inglês *embarrassingly parallel*) (Foster 1995), fazendo com que sejam aderentes a este tipo de paralelismo aninhado. O repositório de proveniência é mantido utilizando o banco de dados relacional PostgreSQL versão 8.4.6 que foi configurado em uma máquina dedicada na Amazon EC2 (http://mp4-4.dyndns.info). O executável do SciCumulus com as dependências embutidas encontra-se em fase de liberação e será disponibilizado em https://sourceforge.net/projects/scicumulus/. O sítio oficial do SciCumulus se encontra em http://gexp.nacad.ufrj.br/scicumulus. A seguir explicamos com mais detalhes passos importantes da implementação do SciCumulus.

5.9.1 Criação do *Cluster* Virtual

O SciCumulus é responsável por criar um *cluster* virtual no ambiente de nuvem. Já que é utilizado o MPJ, torna-se obrigatório a definir o nó principal (que é uma máquina virtual marcada com *rank* 0 - um número comum usado para identificar um processo específico de um conjunto de processos em execução) e os nós executores (*rank* 1, 2 e assim por diante). Essa abordagem que utiliza nós principais e nós executores não difere muito da abordagem MapReduce (Dean e Ghemawat 2010), já aplicada com sucesso em processamento paralelo na nuvem. Antes de configurar o *cluster* virtual, o SciCumulus consulta o repositório de proveniência para descobrir as imagens (semelhantes a imagens de CD, onde além dos arquivos de dados que estão contidos na imagem, a mesma também contém todos os metadados do sistema de arquivos, incluindo o código de inicialização do sistema operacional, estruturas e atributos) que podem ser utilizadas e os tipos de máquinas virtuais existentes. Para que esse processo funcione, uma imagem personalizada (*i.e.* AMI no caso do Amazon EC2) para a execução de máquinas virtuais tem que ter sido configurada *a priori*.

Para gerenciar o *cluster* na nuvem utilizamos um pacote de *software* chamado Vappio[3]. O Vappio fornece serviços *Web* para dinamicamente iniciar (*vp-add-instances*), buscar (*vp-search-instances*), descrever (*vp-describe-instances*) e encerrar (*vp-terminate-instances*) máquinas virtuais em seu ambiente de nuvem. Estes serviços *Web*, por sua vez utilizam as chamadas a API provida pelo Amazon EC2, incluindo *ec2-run-instances, ec2-terminate-instances* e *ec2-describe-instances*. Baseado no Vappio, o SciCumulus cria uma lista de IPs de máquinas virtuais instanciadas e essa lista é usada para criar o arquivo *machines.conf* que mantém a lista das máquinas virtuais disponíveis para o arcabouço do MPJ. A única desvantagem desta abordagem é que, quando uma mudança é necessária na imagem existente, uma nova imagem deve ser criada (um processo que geralmente leva um tempo considerável para ser realizado) e o repositório de proveniência tem que ser atualizado, a fim de usar essa nova imagem. Esta mudança na imagem se dá sempre que o cientista vislumbrar a necessidade de se utilizar um novo programa (ou uma nova versão) não antes pertencente ao *workflow*.

5.9.2 Conexão *Desktop*-Ambiente de Nuvem

Na camada cliente, os componentes de carga e *download* foram implementados para se conectar às máquinas virtuais em que os componentes de distribuição e execução foram implantados, e assim possibilitar enviar/receber os arquivos de dados diretamente para um sistema de arquivos compartilhado na nuvem. Tal conexão é realizada utilizando o protocolo SSH e, apesar de usarmos uma conexão segura para a transferência de dados (SSH e SCP), questões de segurança estão fora do escopo desta tese e são discutidas em outros trabalhos (Gadelha e Mattoso 2008, Jensen *et al.* 2009). O componente de despacho se conecta via SSH com o intermediador de execução na camada de distribuição para iniciar a execução paralela de atividades do *workflow*. Toda a comunicação entre as máquinas virtuais do *cluster* virtual também se dá através do SSH.

5.9.3 Configuração do Disco Compartilhado

Conceitualmente, o SciCumulus pode ser implementado para trabalhar com arquitetura *shared-nothing* onde não há nenhuma área de dados compartilhada. Entretanto, para a

[3] http://vappio.sf.net

implementação desta tese, optamos por utilizar uma área compartilhada para o SciCumulus. Esta opção foi adotada também pela equipe do SGWfC Pegasus para portar os *workflows* para a nuvem da Amazon. A fim de fornecer um sistema de arquivos compartilhado, configuramos um Bloco de Armazenamento Elástico da Amazon (do inglês *Elastic Block Storage* - EBS) (Amazon EC2 2010) por máquina virtual. Os EBS podem ser acessados por máquinas virtuais e seu ciclo de vida é independente do ciclo de vida das máquinas virtuais utilizadas pelo SciCumulus. Em outras palavras, este volume é persistente. No entanto, apesar de ser razoavelmente eficiente o EBS restringe o tamanho do dado a ser armazenado, o que representa uma séria limitação já que o volume de dados envolvidos em uma execução de um *workflow* científico pode ser grande.

Seguindo a abordagem proposta por Jackson *et al.* (2010), os dados de entrada são transferidos para o EBS e todos os dados de saída são armazenados no *Amazon Elastic Store* (S3), que é naturalmente menos eficiente, porém oferece uma área de armazenamento de 200 TB. Para conectar as máquinas virtuais com o Amazon S3 utilizamos o *Subcloud* (Sub*Cloud* 2011). O *Subcloud* é um sistema de arquivos compartilhado empresarial construído em cima do Amazon S3. Usando o *Subcloud*, somos capazes de montar um diretório em cada máquina virtual e apontar para uma única área (do inglês *bucket*) no Amazon S3, criando um sistema de arquivos virtual compartilhado.

5.9.4 Processo de Instrumentação

Para que o SciCumulus gere as *cloud activities* a serem executadas, o componente explorador de parâmetros se baseia em *templates* e arquivos de configuração com as linhas de comando padrão e instrumentadas para invocar os programas que estão associados às atividades do *workflow* (e consequentemente com as *cloud activities*). Nesses arquivos de *templates* e de configuração, essas linhas de comando não contem os valores reais de parâmetros para invocação dos programas, mas sim *tags* semelhantes às encontradas em JSP (*Java Server Pages*) (Bergsten 2003). Uma vez que uma *cloud activity* é gerada, é executado um processo de instrumentação.

Na instrumentação de uma *cloud activity*, as *tags* no arquivo *template* são substituídas por valores reais dos parâmetros encapsulados na *cloud activity*. Após essa etapa, cada arquivo *template* já modificado é copiado para um diretório específico onde

a *cloud activity* irá executar. Denominamos esse diretório como *workspace* da *cloud activity*. A Figura 20 apresenta exemplos de instrumentação de 3 *cloud activities* realizado no SciCumulus pelo explorador de parâmetros.

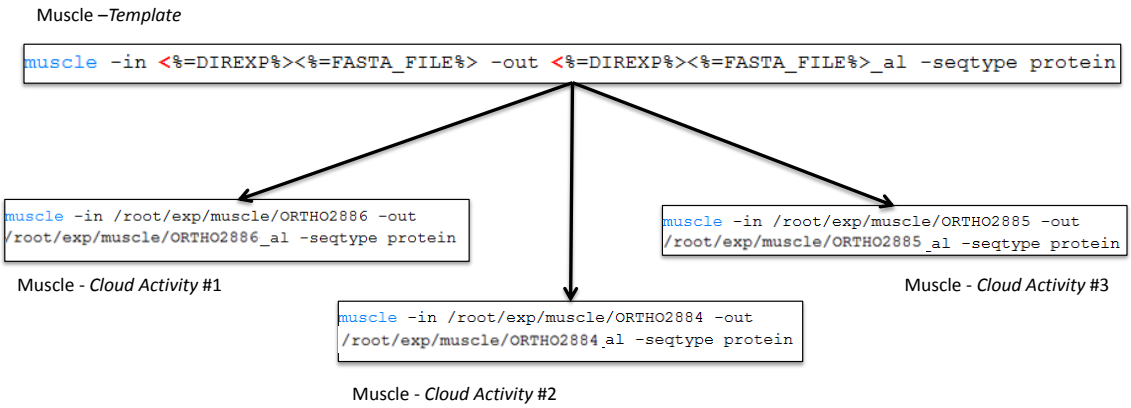


Figura 20 Processo de instrumentação no SciCumulus

No exemplo da Figura 20, apresentamos o conteúdo do arquivo *template* da atividade Muscle (programa de bioinformática que será apresentado no Capítulo 6 com mais detalhes). Nesse *template*, toda a chamada ao programa já está configurada com exceção do arquivo que será consumido (que é efetivamente o valor do parâmetro). O valor do parâmetro é representado através da *tag* *<%=FASTA_FILE%>*. Durante o processo de instrumentação, o SciCumulus substitui a *tag* pelos valores do parâmetro para cada *cloud activity*. No exemplo, existem três *cloud activities*, cada qual consumindo um arquivo diferente.

5.9.5 Acoplamento do SciCumulus com o VisTrails

Os componentes da camada cliente do SciCumulus foram implantados no SGWfC VisTrails para a execução dos experimentos apresentados nesta tese. Para facilitar a implantação no VisTrails, os componentes de carga, despacho e *download* foram implementados como um único módulo no VisTrails. Um módulo no VisTrails, assim como o próprio SGWfC VisTrails, é desenvolvido utilizando a linguagem de programação Python (Python 2010). De uma maneira geral, todo módulo possui portas de entrada e saída e são através delas que é realizada a comunicação com outros módulos presentes no *workflow*. Esses módulos são abstrações ilustradas de uma atividade que o cientista deseja executar dentro de um *workflow*. A Figura 21 ilustra um módulo no VisTrails, chamado *Soma*, em que estão presentes duas portas de entrada, localizadas na parte superior esquerda e uma porta de saída, localizada na parte inferior

direita. O módulo tem sua execução iniciada ao receber nas portas de entrada todos os valores necessários para realizar sua operação. Após esse valor ser obtido, ele é repassado à porta de saída, determinando o fim das operações do módulo.

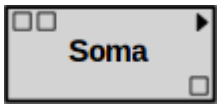


Figura 21 Exemplo de um Módulo no VisTrails

Como o Python é uma linguagem orientada a objetos, todo módulo criado deve herdar de uma classe pai chamada *Module*, pois é através dessa classe que serão criadas as portas de entrada e saída, além de outras funções como acessar os valores passados as essas portas, instanciação do módulo entre outras operações. Para integrar o SciCumulus com o VisTrails foi desenvolvido um módulo chamado SciCumulusCall, que tem como principal função o despacho da atividade local para a nuvem através do VisTrails usando o SciCumulus. Para isso, o cientista precisa configurar uma série de parâmetros necessários para a correta execução do SciCumulusCall e invocação do SciCumulus. A lista de parâmetros do módulo segue abaixo (cada parâmetro do módulo está relacionado com um elemento do arquivo de especificação do SciCumulus, que é descrito nas seções a seguir neste capítulo):

i. *Tag*: nome do programa a ser executado na nuvem;
ii. *Description*: pequena descrição sobre o programa que será executado na nuvem;
iii. *Template Directory*: diretório com o arquivo de *template* que contém a chamada do programa a ser executado e seus recursos necessários;
iv. *Activation*: arquivo instrumentado com a chamada do programa a ser executado na nuvem;
v. *Parameter File*: arquivo com os parâmetros para ser consumido pelo programa na nuvem;
vi. *Bucket Name*: nome do *bucket* (área compartilhada na nuvem) em que os dados de entrada e de saída serão armazenados
vii. *Database Host*, *Database Port*, *Database User* e *Database Password*: endereço, porta, usuário e senha para acesso ao servidor de banco de dados onde são armazenados os dados de proveniência.

viii. *Machine Image:* ID da imagem que será usada para instanciar as máquinas virtuais envolvidas no experimento
ix. *Number of Machines*: número de máquinas a serem instanciadas[4]
x. *Machine Type*: tipo de máquina virtual a ser utilizada[5]
xi. *Security Access Key*: chave de acesso para criação das máquinas virtuais através da API da Amazon.

A Figura 22 apresenta uma visão geral da interface do VisTrails com o módulo desenvolvido ao centro e à direita os parâmetros que devem ser preenchidos pelo cientista. A ideia principal é que o cientista utilize o módulo de invocação do SciCumulus como se fosse um módulo comum do VisTrails. Com todos os parâmetros configurados, o SciCumulusCall inicia sua execução seguindo os seguintes passos:

i. Criação do arquivo de configuração do SciCumulus (*sciCumulus.xml*) em um diretório temporário – este arquivo tem como objetivo informar ao SciCumulus os parâmetros de acesso à nuvem, como por exemplo, a chave de segurança, a quantidade de máquinas a serem instanciadas e os tipos das mesmas;
ii. Criação do arquivo XML de definição do *workflow* (vide Subseção 5.6 para maiores detalhes);
iii. Transferência de dados de entrada e diretórios para o ambiente de nuvem;
iv. Despacho da atividade para ser executada em paralelo com o SciCumulus;
v. Verificação do fim da execução da atividade na nuvem;
vi. Sinalização do fim da execução do SciCumulusCall – o fluxo de execução e controle retorna para o VisTrails.

---

[4] Este parâmetro é opcional. Caso o cientista opte por não preenchê-lo, o SciCumulus automaticamente dimensiona o conjunto de máquinas virtuais para melhor atender ao cientista (vide Subseção 5.2 para maiores detalhes).

[5] Este parâmetro é opcional. Caso o cientista opte por não preenchê-lo, o SciCumulus automaticamente dimensiona a quantidade de máquinas virtuais por tipo para melhor atender ao cientista (vide Subseção 5.2 para maiores detalhes).

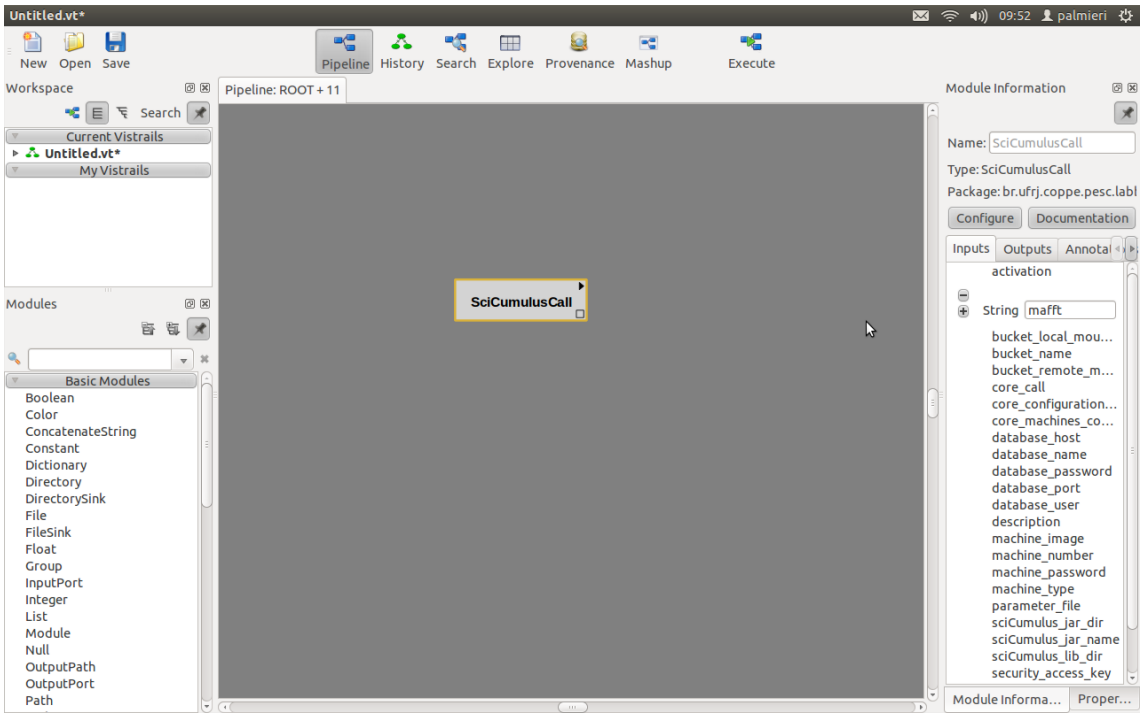


Figura 22 Implantação do módulo SciCumulusCall no SGWfC VisTrails

Capítulo 6 - Estudo de Caso: *Workflow* de Análise Filogenética

Este capítulo descreve as características do *workflow* científico modelado e executado com o SciCumulus no ambiente de nuvem Amazon EC2 para a avaliação experimental apresentada nesta tese. O objetivo principal deste capítulo é prover os principais conceitos de biologia evolutiva e análise filogenética, necessários para a compreensão do exemplo de experimento em larga escala escolhido.

6.1 Contextualização

Durante os últimos dois séculos, cientistas como Jean-Baptiste Lamarck (Delange 2002) e Charles Darwin (Darwin 2003) estudaram biologia evolutiva (Futuyma e Futuyma 1997, Muehlenbein 2010, Pigliucci e Kaplan 2006) para explicar o processo de evolução dos seres vivos em nosso planeta. Com a descoberta da estrutura do DNA e com o desenvolvimento contínuo da nova geração de técnicas de sequenciamento de DNA, os cientistas se tornaram capazes de estudar em mais detalhes a base para a herança genética. Devido a estas novas descobertas, a área de biologia evolutiva cresceu em um ritmo acelerado, tornando possível mapear a transição de características na história evolutiva de qualquer organismo vivo, principalmente, através do uso de filogenia e árvores filogenéticas para análise evolutiva. Filogenia (ou filogênese) é o termo comumente utilizado para hipóteses de relações evolutivas (ou seja, relações filogenéticas) de um grupo de organismos, isto é, determinar as relações ancestrais entre espécies conhecidas.

Por sua vez, uma árvore filogenética, também chamada de árvore da vida (Zvelebil e Baum 2007), é uma representação gráfica, em forma de uma árvore, das relações evolutivas entre várias espécies ou outras entidades que possam ter um ancestral comum, conforme o exemplo apresentado na Figura 23. Em uma árvore filogenética, cada nó com descendentes representa o mais recente antepassado comum, e os comprimentos dos ramos podem representar estimativas do tempo evolutivo. Cada nó terminal em uma árvore filogenética é chamado de "unidade taxonômica" ou táxon (Daubin *et al.* 2003, Yang 1994).

Durante a última década, tem havido um aumento sem precedentes no volume de dados com o objetivo de sequenciamento para categorizar todos os genes de genomas de diversos organismos, por exemplo, o genoma humano (Lander 2001), bem como outros organismos. Resultados obtidos a partir de tecnologias filogenéticas podem contribuir para outras áreas de bioinformática como o desenvolvimento de novas drogas (Anderson 2003), uma vez que se soubermos o tratamento para uma determinada doença causada por um determinado agente, podemos estender esse tratamento para doenças causadas por agentes que sejam descendentes do mesmo.

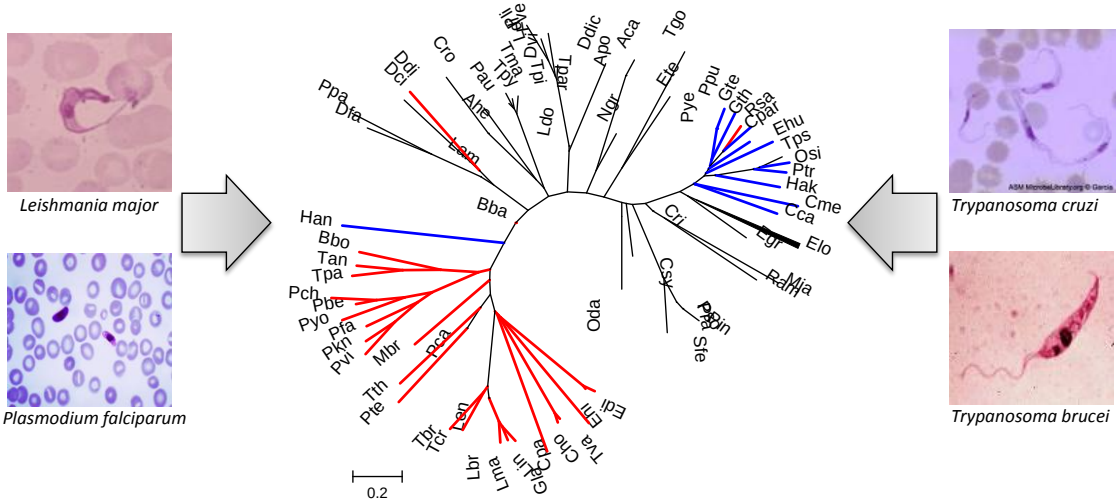


Figura 23 Exemplo de árvore filogenética adaptado de Ocaña e Dávila (2011)

No entanto, reconstruir as relações entre os organismos vivos utilizando árvores filogenéticas está longe de ser uma tarefa trivial e se apresenta como um desafio fundamental em biologia permitindo uma melhor compreensão dos eventos evolutivos envolvidos em determinados genes de interesse. A fim de gerar uma árvore filogenética com qualidade aceitável alguns passos devem ser realizados antes do objetivo final que é a geração da árvore propriamente dita. Esses passos são partes do protocolo (experimento) de análise filogenética.

O alinhamento múltiplo de sequências (do inglês, *Multiple Sequence Alignment*, ou simplesmente MSA, como utilizaremos a partir de agora) é um desses passos, e pode ser considerado fundamental em muitos experimentos de bioinformática, inclusive em experimentos de genômica comparativa e filogenia (Eddy 2009). Um MSA é um alinhamento de pelo menos duas sequências biológicas (sequências de guanina, adenina, citosina e timina, que são a base para formar cadeias polinucleotídeas que, por sua vez, formam o DNA), geralmente proteínas, DNA ou RNA. De um modo geral, assume-se

que o conjunto de sequências dadas como entrada são evolutivamente relacionadas, *i.e.* as sequências compartilham ancestrais. Do alinhamento produzido baseado na sequência de entrada, podemos inferir uma homologia, e estamos aptos então a utilizar esses dados como insumos para a análise filogenética. Para efeito de ilustração, consideremos as sequências TAGGTCA e TAGCTA (Mattos 2008, Zvelebil e Baum 2007). Um possível alinhamento entre elas é mostrado na Figura 24.

```
TAGGTCA␣␣␣␣␣␣␣
␣␣␣␣␣␣␣TAGCTA
```

Figura 24 Um alinhamento entre TAGGTCA e TAGCTA

Uma análise rápida nas duas sequências, nos mostra que elas são bastante parecidas, embora o alinhamento realizado na Figura 24 não tenha mostrado essa semelhança de modo satisfatório. Um alinhamento com mais qualidade poderia ser o alinhamento mostrado na Figura 25.

```
TAGGTCA
TAGCT␣A
```

Figura 25 Um alinhamento com mais qualidade entre TAGGTCA e TAGCTA

O alinhamento produzido e exemplificado na Figura 25 nos evidencia que as sequências são bastante semelhantes e que suas únicas diferenças são uma base G na primeira sequência ocupando o lugar de uma base C na segunda sequência e a penúltima base (um C) da primeira sequência que não está presente na segunda sequência.

Existem vários algoritmos de alinhamento disponíveis para identificar as regiões de similaridade entre sequências. O ClustalW (Thompson *et al.* 1994) é um método progressivo de alinhamento que é amplamente utilizado em bioinformática, assim como o Kalign (Lassmann e Sonnhammer 2005), que é um método altamente eficiente de MSA e que emprega o algoritmo Wu-Manber de correspondência de *strings*. O MAFFT (Katoh e Toh 2008) fornece vários procedimentos classificados em três tipos, o método progressivo, o refinamento iterativo, e o algoritmo de refinamento iterativo de valores de consistência. Em geral, há um equilíbrio entre velocidade e precisão quando usamos o MAFFT. O método Muscle (Edgar 2004) estima rapidamente distâncias usando

contagem kmer, enquanto que o Probcons (Do *et al.* 2005) utiliza cadeias de Markov ocultas durante o alinhamento.

Além do passo de alinhamento das sequências, outros passos são fundamentais em uma análise filogenética são a escolha do modelo de evolução a ser usado e a geração da árvore propriamente dita. Uma vez que o experimento de análise filogenética é baseado em simulações diversas, era natural que fossem desenvolvidas ferramentas computacionais para realizar cada um dos passos das análises. Além dos programas de alinhamento, existem vários outros programas bem conhecidos e testados que podem ser naturalmente orquestrados para desenvolver um experimento de análise filogenética (Busset *et al.* 2011, Chen *et al.* 2009, Dereeper *et al.* 2008, 2010, Guindon *et al.* 2005, Jones *et al.* 2011, Keane *et al.* 2007, Lin *et al.* 2005, Sánchez *et al.* 2011).

Grande parte das análises filogenéticas depende de grandes volumes de dados e um aparato científico de alta tecnologia. Esses experimentos comumente demandam computação intensiva e uma gerência complexa dos dados. Essa gerência era muitas vezes realizada de forma *ad hoc*, seja manualmente ou através de *scripts*, o que acarretava uma série de problemas como erros na execução do experimento, perda no controle das versões dos programas utilizados, problemas no controle dos dados produzidos, etc. Além disso, em cada experimento citado anteriormente existe a real necessidade de se explorar os parâmetros de variabilidade (por exemplo, valores de *bootstrap*) para o mesmo conjunto de dados de entrada. Neste contexto, técnicas e metodologias de *workflows* científicos podem melhorar a gerência dos experimentos. Como estudo de caso desta tese, modelamos o *workflow* de análise filogenética chamado SciPhy (Ocaña *et al.* 2011b) que tem como objetivo gerar uma árvore filogenética baseado em um conjunto de sequências biológicas de entrada.

6.2 O SciPhy

O SciPhy (Ocaña *et al.* 2011b) é um *workflow* científico que foi projetado para gerar árvores filogenéticas com máxima verossimilhança (Yang 1994). Ele foi projetado inicialmente para trabalhar com sequências de aminoácidos, porém seu uso pode ser extrapolado para outros tipos de sequências biológicas. O *workflow* SciPhy é composto por sete atividades principais que são:

i. A construção do alinhamento genético (MSA);
ii. A conversão de formato do alinhamento;

iii. A pesquisa sobre o melhor modelo evolutivo a ser usado;

iv. A construção da árvore filogenética;

v. A concatenação dos alinhamentos produzidos de forma a gerar um super-alinhamento que servirá para inferir a relação do genoma completo;

vi. A escolha do melhor modelo evolutivo para o super-alinhamento e;

vii. A construção da árvore filogenômica que contém informações sobre todos os organismos associados.

Estas atividades, respectivamente, executam as seguintes aplicações de bioinformática: programas de alinhamento genético (permitindo ao cientista a escolher entre o MAFFT, o Kalign, o ClustalW, o Muscle, ou o ProbCons), o ReadSeq (Gilbert 2003), o ModelGenerator (Keane *et al.* 2006), o RAxML (Stamatakis 2006), um *script* Perl, o ModelGenerator e o RAxML. A Figura 26 apresenta a visão conceitual do SciPhy.

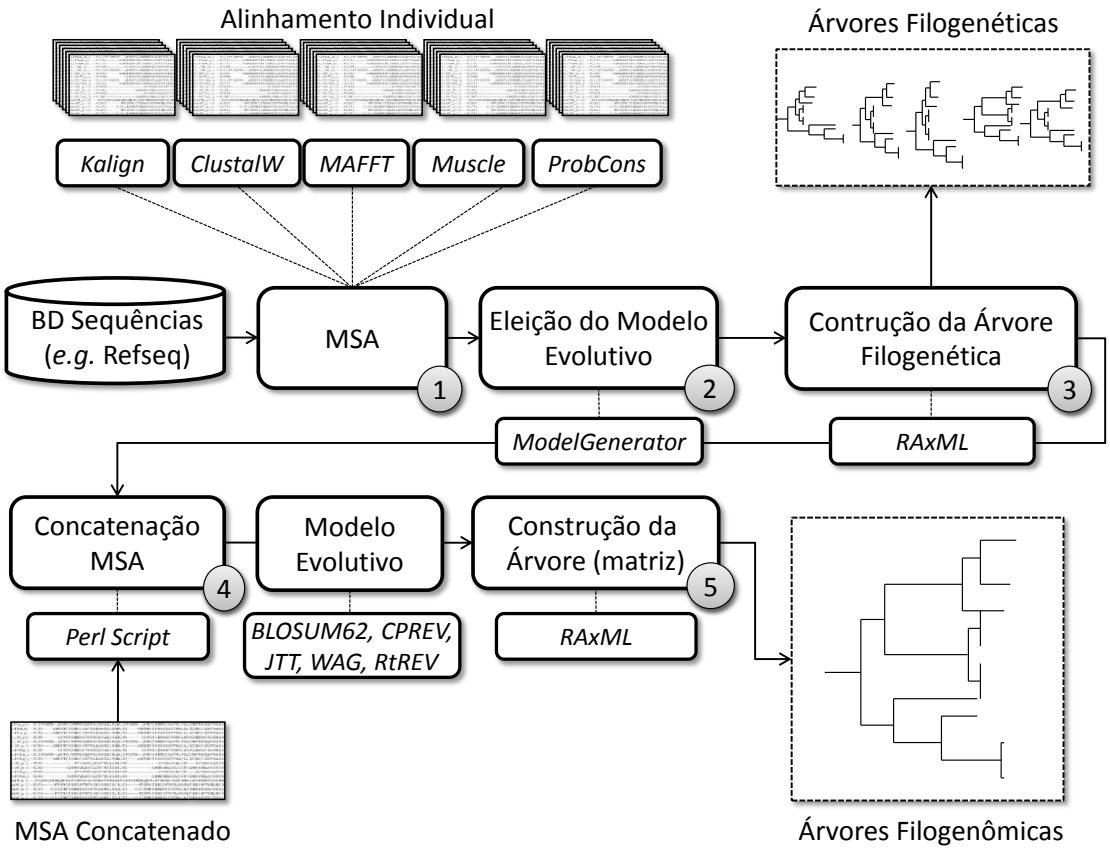


Figura 26 Especificação conceitual do *workflow* SciPhy

A primeira atividade do SciPhy (MSA) constrói alinhamentos individuais utilizando um dos cinco programas disponíveis para alinhamento genético: o ClustalW,

o Kalign, o MAFFT, o Muscle, ou o ProbCons. Cada programa de alinhamento recebe um arquivo multi-fasta (que pode representar um aminoácido ou não) como entrada (a partir de um conjunto de arquivos multi-fasta existentes), produzindo como saída um alinhamento (MSA). Neste ponto, o SciPhy obtém o MSA individual (produzido pelo programa de alinhamento eleito para a atividade). Na segunda atividade, o alinhamento é convertido para o formato PHYLIP (Felsenstein J 1989). Formato este que é alcançado através da utilização do programa ReadSeq.

Cada MSA é testado na terceira atividade (Eleição do Modelo Evolutivo) para encontrar o melhor modelo evolutivo utilizando o ModelGenerator e ambos (MSA individual e modelo evolutivo) são utilizados na quarta atividade (Geração da Árvore Filogenética) para gerar árvores filogenéticas utilizando o RAxML com parâmetro de *bootstrap* configurável. Consequentemente, várias árvores diferentes podem ser obtidas para cada um dos programas de MSA que foram eleitos. Uma vez que os cientistas não conhecem *a priori* qual o método de alinhamento que produz o melhor resultado final (a melhor árvore), eles têm que realizar uma análise filogenética exploratória (varredura de parâmetros), ou seja, executar o SciPhy várias vezes (uma vez para cada método MSA).

Na quinta atividade os alinhamentos produzidos são concatenados em um super-alinhamento utilizando um *script* Perl (Wall *et al.* 2000) próprio. Esse super-alinhamento deve ter seu modelo processado com o ModelGenerator na sexta tarefa, sendo que o super-alinhamento deve ser processado com cinco modelos diferentes (BLOSUM62, CPREV, JTT, WAG e RtREV). Para cada um dos modelos processados, uma nova árvore é gerada na sétima atividade através da execução do programa RAxML. Mais detalhes sobre o SciPhy podem ser obtidas no artigo de Ocaña *et al.* (2011b).

6.3 A Varredura de Parâmetros no SciPhy

Apesar de conceitualmente o SciPhy ser computacionalmente simples (uma vez que são “apenas” seis programas orquestrados), na prática ele pode ser demasiadamente complexo de ser gerenciado devido ao volume de dados trabalhados e a quantidade de parâmetros que devem ser explorados, uma vez que o cientista não conhece *a priori* qual a configuração que levará a árvore filogenética com melhor qualidade. Um experimento típico de filogenia pode analisar centenas ou milhares de arquivos multi-fasta, cada um contendo centenas ou milhares de sequências biológicas. Como o

processamento destes arquivos e das combinações de parâmetros é independente, o SciPhy se torna um candidato interessante para explorarmos a execução paralela em experimentos de bioinformática. No contexto do SciCumulus, cada combinação de parâmetros mais dados de entrada no SciPhy gerará uma *cloud activity* diferente, que poderá ser processada em paralelo, aumentando assim a eficiência na execução do *workflow*.

Outro fator que faz com que o SciPhy seja um candidato interessante para estudo de caso é o fato do *workflow* ter um tempo de execução relativamente grande. Por exemplo, com uma configuração de 200 arquivos multi-fasta de entrada (este valor pode ser muito superior), a execução do SciPhy em uma máquina com dois núcleos, apenas variando os métodos de alinhamento (5 execuções) necessita em média 9 dias para terminar. Sem a aplicação de técnicas de paralelismo e com o aumento da combinação de parâmetros e dados de entrada a serem explorados, esse tempo pode se tornar proibitivo para a execução do experimento.

Uma vez que visamos à execução de uma varredura de parâmetros em paralelo no SciPhy, cada uma das atividades é executada para um arquivo de entrada contendo várias sequências diferentes (arquivo multi-fasta). Cada uma dessas execuções pode ser realizada em paralelo. Apesar de não fazer parte do *workflow* conceitual, como estamos executando uma varredura de parâmetros para analisar qual resultado tem mais qualidade, devemos introduzir uma atividade surrogata de verificação do melhor resultado ao final da execução. No entanto, essa verificação que é realizada é mapeada como uma *cloud activity* não paralela, ou na proposta de Ogasawara *et al.* (2011), como uma atividade bloqueante. A representação da execução paralela de SciPhy é apresentada na Figura 27. É Importante ressaltar que existem mudanças no grau de paralelismo durante o curso de execução do *workflow*.

Por exemplo, se nós processarmos os mesmos 200 arquivos multi-fasta como entrada, para as quatro primeiras atividades do *workflow* o SciCumulus irá gerar 200 *cloud activities* associadas a cada atividade. Em seguida, na segunda parte (duas últimas atividades) o SciCumulus irá gerar apenas cinco *cloud activities* para cada atividade (uma para cada método escolhido).

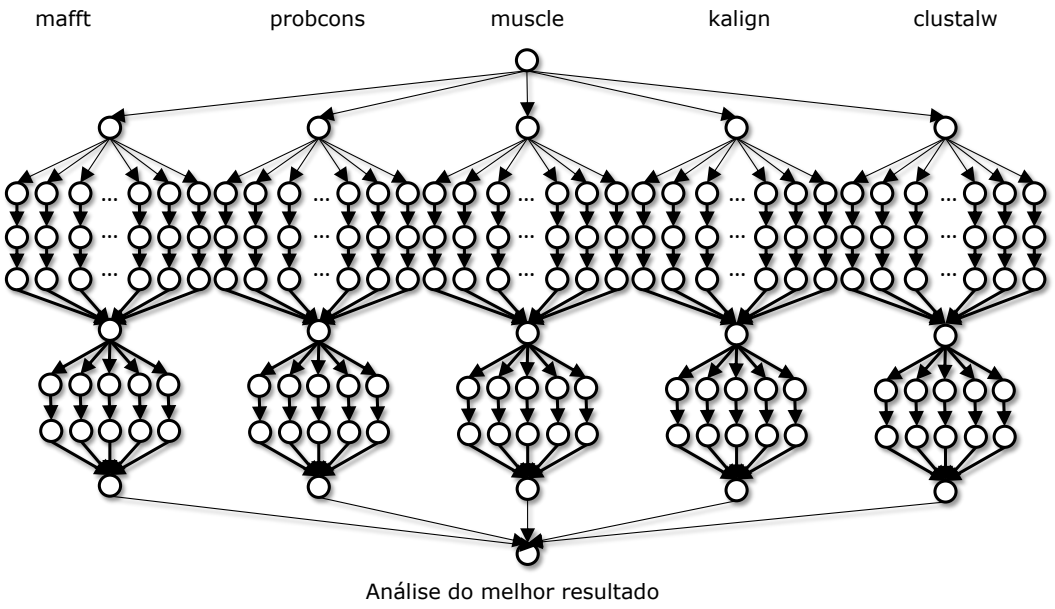


Figura 27 Visualização das *cloud activities* para exploração de parâmetros no SciPhy

Essas mudanças de grau de paralelismo no *workflow* impactam diretamente na definição das *cloud activities* para processamento. Por exemplo, a atividade de escolha da melhor árvore filogenética produzida não pode ser executada em paralelo já que ela é uma atividade bloqueante, ou seja, só poderá existir uma *cloud activity* para executar esta atividade. A Figura 28 apresenta dois possíveis mapeamentos para criação de *cloud activities* no SciPhy, porém podem existir outros não apresentados na Figura 28.

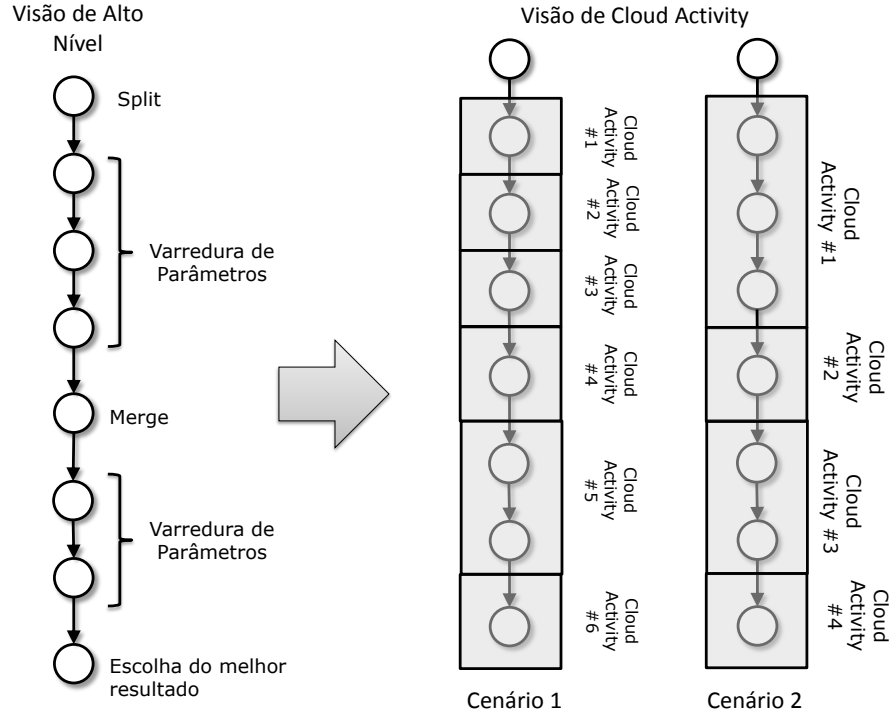


Figura 28 Cenários de mapeamento das *cloud activities* no SciPhy

Capítulo 7 - Avaliação Experimental

Este capítulo apresenta as configurações utilizadas para modelar e executar o *workflow* SciPhy utilizando o SciCumulus bem como os resultados experimentais obtidos. A ideia central deste capítulo é analisar o desempenho e a escalabilidade do SciCumulus frente a soluções existentes que podem ser utilizadas para paralelizar *workflows* que necessitam de PAD em nuvens, tais como implementações baseadas no Hadoop (Hadoop 2012). Para tal, devemos avaliar todos os componentes apresentados nesta tese, tais como a abordagem de escalonamento adaptativo (modelo de custos e algoritmo de escalonamento guloso), as consultas de proveniência e os serviços adicionais como o SciCumulus-ECM, SciMultaneous e SciLightning. Para avaliar o SciCumulus, modelamos o *workflow* SciPhy conforme apresentado no Capítulo 6 em um ambiente de nuvem distribuído e comparamos a abordagem proposta com o Hadoop. O Hadoop foi o escolhido como alvo de comparações, pois a grande maioria das abordagens existentes (e disponíveis para utilização) para paralelizar *workflows* científicos em nuvens é baseada no Hadoop como, por exemplo, o Kepler + Hadoop (Wang *et al.* 2009).

Este capítulo está organizado como se segue. A Seção 7.1 explica o conceito de MapReduce e Hadoop, com o qual iremos comparar o SciCumulus. A Seção 7.2 apresenta a configuração do ambiente utilizado para execução dos experimentos. A Seção 7.3 nos traz a configuração do experimento, como versões utilizadas dos programas, origem dos dados, etc. A Seção 7.4 apresenta o histograma de distribuição de tempo das *cloud activities* do *workflow* SciPhy. A Seção 7.5 apresenta a análise de desempenho da execução não-adaptativa enquanto que a Seção 7.6 apresenta um estudo de escalabilidade com o SciCumulus. A Seção 7.7 discute sobre o custo monetário e a Seção 7.8 sobre a questão dos agrupamentos de *cloud activities* e o impacto gerado na execução do *workflow*. A Seção 7.9 apresenta os resultados obtidos com a execução adaptativa enquanto que a Seção 7.10 discute as melhorias alcançadas com o serviço SciCumulus-ECM na abordagem adaptativa. A Seção 7.11 apresenta a avaliação do SciLightning. A Seção 7.12 discute sobre a vazão no acesso aos dados e o impacto das transferências e na Seção 7.13 avalia uma série de consultas de proveniência e como as mesmas são respondidas pelo SciCumulus. Finalmente na Seção 7.14 apresentamos as considerações finais acerca deste capítulo.

7.1 MapReduce e Hadoop

Como focamos a comparação do SciCumulus com o Hadoop, nesta seção nos preocupamos em apresentar o modelo MapReduce (Dean e Ghemawat 2010) e suas implementações. O MapReduce é um modelo de programação que visa facilitar a paralelização de aplicações em ambientes distribuídos como *clusters* e nuvens de computadores. O MapReduce foca no tratamento de grandes volumes de dados a serem processados em paralelo. Usando o MapReduce, os dados podem ser particionados *a priori* e gerados conjuntos de pares chave-valor para serem processados por uma função *Map* em paralelo, e os resultados intermediários são incorporadas ao resultado final usando a função chamada *Reduce* de acordo com um critério específico. A principal ideia por trás MapReduce é fazer com que seja transparente para os usuários a programação, o balanceamento de carga, e os mecanismos de tolerância a falhas na execução paralela de programas.

Para usar o MapReduce, duas funções principais devem ser programadas: (i) uma função de *Map* na forma *map(in_key, in_val)* → *lista(out_key, intermediate_val)* e (ii) uma função de *Reduce* na forma *reduce(out_key, lista (intermediate_val))* → *lista (out_val)*. Cada uma destas funções é específica de domínio. No contexto de *workflows* científicos, uma nova função de *Map* e de *Reduce* deve ser implementada para cada tipo diferente de atividade. O MapReduce foi inicialmente proposto para aplicações comerciais, tais como indexação de documentos, porém nos últimos anos o seu uso está sendo extrapolado para domínios científicos (Matsunaga et al. 2008, Vega et al. 2010).

Uma das implementações mais famosas e utilizadas do MapReduce é o Hadoop, um arcabouço desenvolvido pelo grupo Apache que fornece o componente central MapReduce e o Sistema de Arquivos Distribuídos do Hadoop (do inglês *Hadoop Distributed File System* - HDFS) nativamente. Na arquitetura do Hadoop um processo central atua como mestre e coordena tarefas do tipo *Map* e *Reduce*, enquanto todos os outros processos atuam como trabalhadores em diferentes nós de computação. Processos trabalhadores executam tarefas que são geradas e distribuídas pelo nó mestre. O mesmo processo trabalhador é capaz de executar vários *Maps* e *Reduces* ao mesmo tempo. Além disso, as implementações de MapReduce como o Hadoop estão sendo usados em nuvem em ambientes como o Amazon EC2.

Embora essas funções de *Map* e *Reduce* em conjunto com as variáveis de ambiente do Hadoop possam ser implementadas e configuradas (respectivamente) por especialistas em computação, pode ser uma tarefa muito complexa para ser desempenhada por cientistas sem uma bagagem mais computacional. Além disso, como a execução das tarefas no MapReduce está dissociada do conceito de SGWfC, não existe apoio algum a descritores de proveniência. A Figura 29 apresenta uma visão esquemática do funcionamento do Hadoop com o *workflow* SciPhy. Nesse caso, cada arquivo multi-fasta de entrada foi processado por um *Map* diferente nas várias máquinas virtuais em um *cluster* virtual no Amazon EC2.

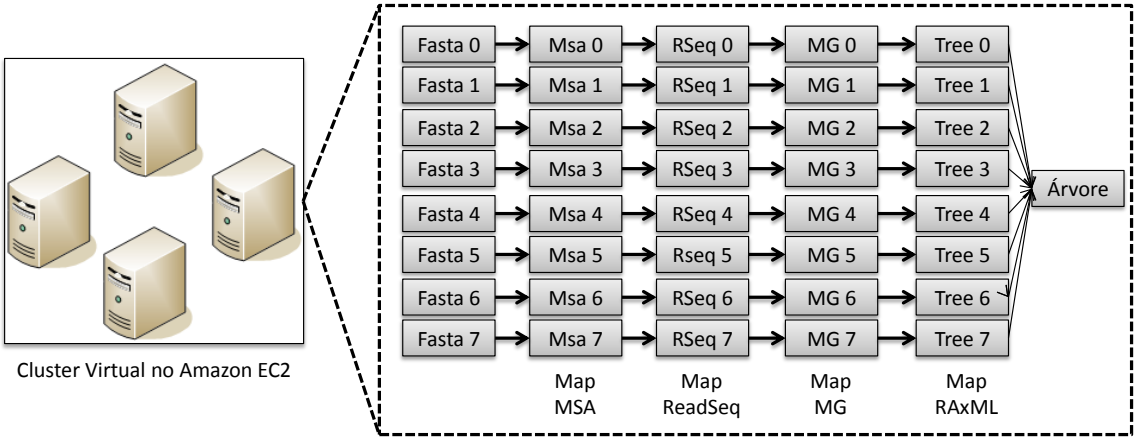


Figura 29 Esquematização do funcionamento do SciPhy com o Hadoop

7.2 Configuração do Ambiente

Para os experimentos executados nesta tese, implantamos o SciCumulus e o Hadoop no ambiente de nuvem Amazon EC2. Conforme discutido anteriormente, o Amazon EC2 fornece vários tipos diferentes de máquinas virtuais para os cientistas para instanciação e uso. Cada um destes tipos possui características únicas (capacidade de CPU, capacidade de memória RAM e capacidade de armazenamento).

Existem vários tipos de máquinas virtuais, tais como a do tipo *micro* (EC2 ID: t1.micro - 613 MB RAM, 1 núcleo, volume EBS de armazenamento de 30 GB), do tipo *large* (EC2 ID: m1.large - 7.5 GB RAM, 850 GB de armazenamento, 2 núcleos) , do tipo *extra-large* (EC2 ID: m1.xlarge - 15 GB de RAM, 1.690 GB de armazenamento, 4 núcleos), *extra-large de alta capacidade* (EC2 ID: c1.xlarge - 7.5 GB RAM, 850 GB de armazenamento, 8 núcleos), e *extra-large quádrupla* (EC2 ID: cc1.4xlarge - 23 GB de RAM, 1.690 GB de armazenamento, 8 núcleos). Nos experimentos apresentados nesta tese, consideramos *clusters* de tamanho de até 128 instâncias do tipo *micro* e *large*

apenas. A Tabela 1 resume os dados de desempenho e precificação dos tipos de máquinas virtuais passíveis de uso pelo SciCumulus.

Tabela 1 Configuração de *hardware* e preços de utilização dos tipos de máquinas virtuais disponíveis

Tipo de Máquina	*Memória*	*Disco*	*UC*[6]	*Núcleos*	*Arquitetura*	*Preço*[7]
Micro	613 MB	-	1 UEC2	1	32 Bits	0,02
Large	7,5 GB	850 GB	2 UEC2	2	64 Bits	0,34
Extra-large	7,5 GB	1.690 GB	4 UEC2	4	64 Bits	0,68
Extra-large de alta capacidade	7 GB	1.690 GB	20 UEC2	8	64 Bits	1,16
Extra-large quádrupla	68,4 GB	1.690 GB	26 UEC2	8	64 Bits	2,00

Cada uma das instâncias utiliza processadores equivalentes ao Intel Xeon quad-core. Cada máquina virtual instanciada nos experimentos de análise filogenética nesta tese é baseada no sistema operacional Linux Cent OS 5 (64 bits), e foi configurada com todos os *software*s necessários e bibliotecas como MPJ (Carpenter *et al.* 2000) e as aplicações da bioinformática. Todas as máquinas virtuais foram configuradas para serem acessadas utilizando SSH sem verificação de senha (embora isso não seja recomendado devido a questões de segurança, que fogem ao escopo desta tese).

Além disso, as imagens das máquinas virtuais (EC2 AMI IDs: ami-e4c7368d e ami-ceb949a7) foram armazenadas na nuvem e o SciCumulus cria o *cluster* virtual para executar o experimento com base nestas imagens. As mesmas imagens foram utilizadas para executar o experimento com o Hadoop. Em termos de *software*, todas as instâncias, não importando seu tipo, executam os mesmos programas e configurações. De acordo com a Amazon EC2, todas as máquinas virtuais foram instanciadas na região leste dos EUA - N. Virginia e seguem as regras de preços daquela localidade[8].

---

[6] Uma unidade de computação (*EC2 Compute Unit*) é equivalente a um processador com relógio entre 1,00 e 2,33 GHz.

[7] Preço calculado por hora em dólares

[8]Atualmente, a Amazon EC2 disponibiliza um *datacenter* em São Paulo que pode ser utilizado de forma a diminuir a latência de comunicação dos usuários brasileiros, entretanto, na época da execução dos experimentos (Julho-Dezembro de 2011), esta região ainda não se encontrava disponível para uso.

7.3 Configuração do Experimento

Para executar o SciPhy, variamos o programa de alinhamento e o modelo evolucionário utilizado no *workflow* científico (ver a possível variabilidade da atividade de alinhamento e da eleição do modelo evolucionário na Figura 26). Nossas execuções utilizam como entrada um conjunto de arquivos multi-fasta de entrada contendo sequências de proteínas extraídas do banco de dados biológicos RefSeq *release* 48 (Pruitt *et al.* 2009) e do ProtozoaDB (Dávila *et al.* 2008), disponibilizado pelo Prof. Dr. Alberto Martín Rivera Dávila da Fundação Oswaldo Cruz. Este conjunto de dados é formado por 1.600 arquivos multi-fasta de aminoácidos e cada arquivo multi-fasta é constituído por uma média de 20 sequências. Essa base dados de entrada é considerada de tamanho médio para experimentos de bioinformática de larga escala.

Para executar o SciPhy, uma vez baixado, cada arquivo multi-fasta de entrada é alinhado para obter um MSA utilizando as seguintes versões de programas: ClustalW versão 2.1, Kalign versão 1.04, MAFFT versão 6,857, versão Muscle 3.8.31, e ProbCons versão 1.12. Cada alinhamento é usado como entrada para o programa ModelGenerator versão 0.85 que elege um modelo evolutivo como saída. Então, ambos, o alinhamento e o modelo evolutivo, são utilizados como entrada para construir árvores filogenéticas utilizando o RAxML-7.2.8-alpha.

Todos os alinhamentos resultantes (de cada programa de alinhamento), obtidos a partir de análises filogenéticas, são concatenados para produzir um arquivo contendo um super-alinhamento. Então, este super-alinhamento é testado com cinco modelos evolutivos (BLOSUM62, CPREV, JTT, WAG, e RtREV) e ambos (super-alinhamento e modelos evolutivos) são utilizados como entrada para construir árvores filogenômicas usando o RAxML-7.2.8-alpha. O *workflow* SciPhy foi modelado utilizando o SciCumulus e o Hadoop versão 0.21.0.

A fim de executar cada uma das atividades do SciPhy no Hadoop tivemos que implementar funções *Map* e *Reduce* específicas. Uma vez que a entrada dos programas de alinhamento é um arquivo multi-fasta único ou muitos arquivos multi-fasta, para as atividades de alinhamento modeladas no Hadoop a entrada é sempre um conjunto de arquivos multi-fasta. Desta forma, coletamos os arquivos de entrada e os carregamos no HDFS (do inglês *Hadoop Distributed File System*) para criar pares de chave-valor para o componente *Mapper* (responsável pela execução dos *Maps* no Hadoop). O *Mapper*

cria um processo Java para invocar o programa específico (ClustalW, Kalign, MAFFT, Muscle ou ProbCons). A chave do *Map* é o nome do arquivo, e o valor do *Map* inclui o caminho completo para cada arquivo de entrada que foi carregado no HDFS.

Cada tarefa *Map* baixa os arquivos multi-fasta de entrada a partir HDFS, e transfere essa entrada de forma a executar o programa associado. As próximas atividades no *workflow* seguem a mesma abordagem. Para cada programa que é invocado, novas funções de *Map* e *Reduce* devem ser implementadas (para o ModelGenerator, o RAxML e para o *script* de geração do super-alinhamento), mas essas novas atividades vão consumir os dados produzidos a partir da atividade anterior, criando assim a dependência de dados entre as mesmas. Essa dependência teve de ser implementada uma vez que o Hadoop não oferece apoio para execução de *workflows*. A função de *Reduce* é responsável por coletar todas as saídas intermediárias e agrupá-las em um único arquivo de saída a ser transferida para a máquina do cientista.

7.4 Distribuição do Tempo e Execução das *Cloud Activities*

Uma análise que devemos realizar antes de executar qualquer experimento é a questão da distribuição dos tempos de execução das *cloud activities*. Baseado nesta distribuição, podemos analisar o comportamento dos experimentos a seguir apresentados e desenhar as conclusões apropriadamente. Analisando o repositório proveniência dos SciCumulus podemos construir o histograma do tempo de execução para todas as *cloud activities* de execuções passadas do *workflow* SciPhy, conforme apresentado na Figura 30. Com base nas informações obtidas a partir deste repositório, foi possível calcular o a média (1.703,5 segundos) e desvio padrão (108,3 segundos) dos tempos de execução das *cloud activities*.

A principal vantagem de termos tal distribuição de tempos de execução é que podemos traçar heurísticas diferentes para distribuição como escalonar *cloud activities* computacionalmente intensivas para máquinas virtuais mais poderosas se estamos focando em desempenho. Por outro lado, é possível escalonar *cloud activities* computacionalmente intensivas para máquinas virtuais mais baratas se estamos nos concentrando em redução do custo monetário. Um *workflow* com uma distribuição como esta pode se beneficiar do modelo de custos proposto uma vez que temos tempos diferentes de execução, desta maneira possibilitando traçar diferentes heurísticas de escalonamento. Na próxima seção, apresentamos os resultados de desempenho do

SciCumulus executando o SciPhy com diferentes configurações do modelo de custo que representam cada um dos cenários anteriormente apresentados (C1, C2, C3 e C4).

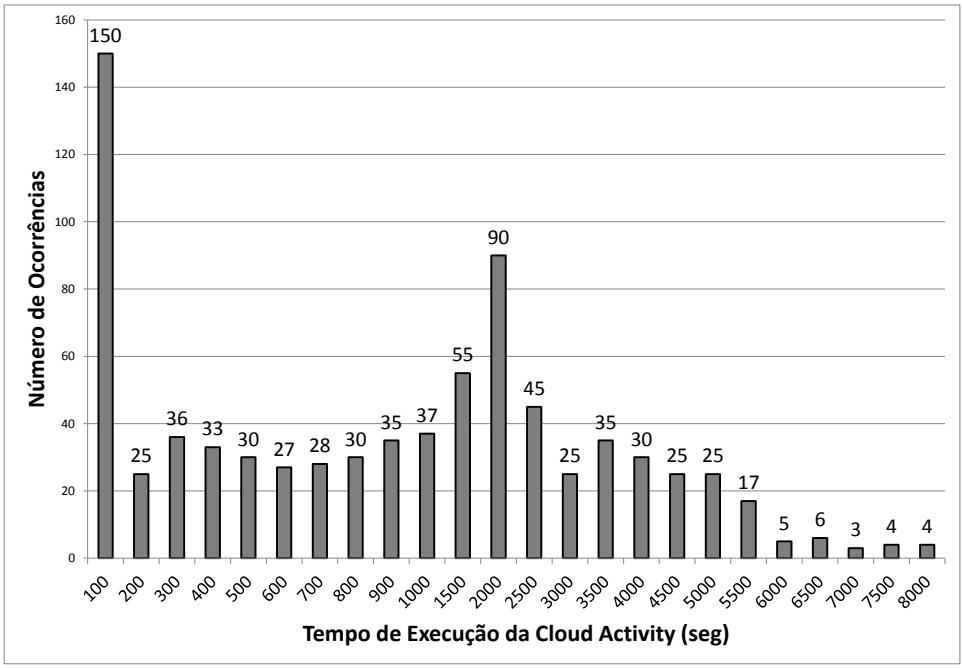


Figura 30 Distribuição de tempo das *cloud activities*

7.5 Análise de Desempenho da Execução Não-Adaptativa

A fim de realizarmos uma comparação justa da abordagem de escalonamento do SciCumulus com o Hadoop, primeiramente executamos o SciPhy no SciCumulus sem introduzir nenhuma variação no número de máquinas envolvidas na execução (uma vez que o Hadoop não escala o número de máquinas virtuais durante a execução do *workflow*) e consumindo um total de 200 arquivos multi-fasta de entrada (cada um dos os arquivos contêm cerca de 20 sequências a serem alinhadas). Optamos por processar uma fração da coleção de entrada devido a limitações de tempo uma vez que o SciPhy foi executado 240 vezes para esta primeira análise, e seria inviável executar esta quantidade de vezes processando a coleção inteira.

Assim, nesta primeira parte da análise experimental, executamos varreduras de parâmetros reais, utilizando um cenário "estático", onde não existem novas máquinas virtuais ou destruições de máquinas virtuais durante o curso de execução do *workflow*. Desta forma, pretendemos analisar o desempenho do escalonamento do SciCumulus baseado no modelo de custos proposto, fixando o número de máquinas virtuais utilizados em cada execução. Em cada execução do *workflow* para comparar o SciCumulus com o Hadoop, metade das máquinas virtuais instanciadas é do tipo *large* e

a outra metade é do tipo *micro*. Desta forma, tentamos simular um cenário real (onde as máquinas disponíveis podem não ser sempre do mesmo tipo) e explorar as vantagens do modelo de custo proposto no escalonamento e execução das atividades de *workflows* científicos, variando o tipo de recursos utilizados.

Para cada cenário apresentado, calculamos o tempo de execução do SciPhy utilizando o SciCumulus e o Hadoop. Além disso, calculamos o valor da aceleração (*speedup*) para cada cenário estudado. A aceleração pode ser definida como a relação entre o tempo gasto para executar uma tarefa (no nosso caso uma *cloud activity*, uma atividade ou o *workflow* completo) com um único processador e o tempo gasto com *n* processadores, ou seja, a aceleração é a medida do ganho em tempo alcançado.

Em primeiro lugar, ajustamos o modelo de custo proposto para focar em desempenho (cenário C1) definindo $\alpha_1 = 0,9$, $\alpha_2 = 0,05$ e $\alpha_3 = 0,05$. Por motivos que ainda estão sendo estudados, não podemos zerar os critérios do modelo de custo, pois o SciCumulus apresenta um comportamento anômalo no escalonamento quando configurado desta forma. Desta forma, optamos por atribuir um percentual de impacto mínimo (5%) para os critérios menos importantes. O tempo de execução para esta configuração e os resultados de aceleração são apresentados na Figura 31 e Figura 32, respectivamente. Podemos observar a partir da Figura 31, que o tempo total de execução do *workflow* SciPhy utilizando tanto o SciCumulus quanto o Hadoop diminui, como esperado, quando provemos uma quantidade maior de máquinas virtuais para processamento (e, consequentemente, mais núcleos virtuais).

Os dados brutos de tempo podem ser visualizados também na Tabela 2. Nesta tabela apresentamos os dados da execução do SciPhy para cada um dos métodos de alinhamento isoladamente (linhas ClustalW, Mafft, Muscle, ProbCons e Kalign) e a execução dos 5 métodos de uma única vez. Como o cientista tem que executar os cinco métodos sempre, uma vez que não sabe *a priori* qual dará o melhor resultado, todas as execuções consideradas nos experimentos a seguir se referem à execução do SciPhy com todos os cinco métodos de alinhamento.

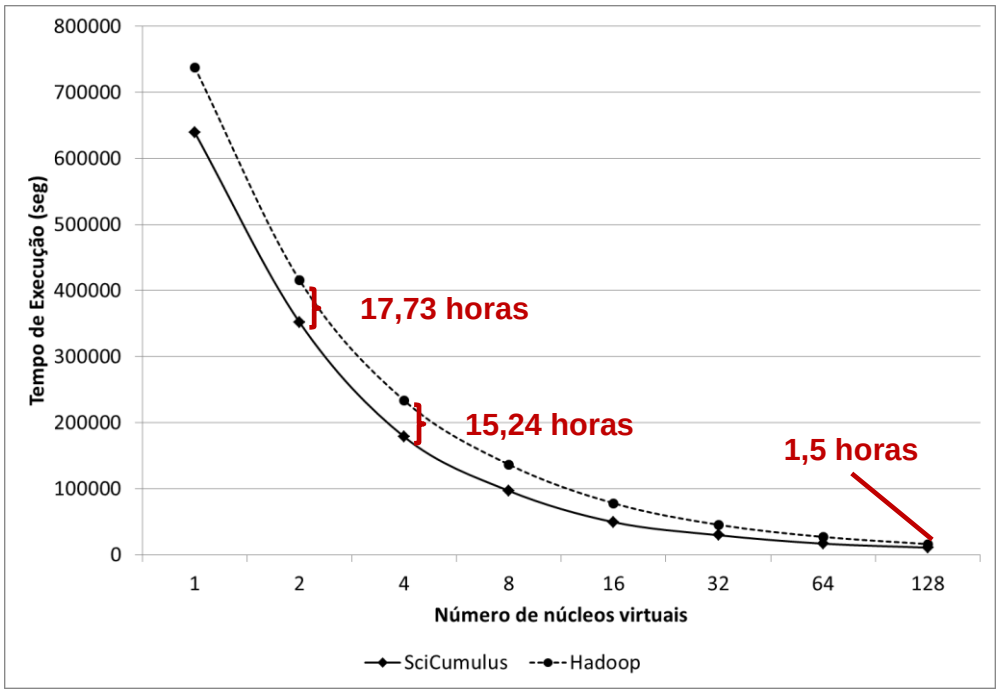


Figura 31 Tempo total de execução para o cenário C1

O desempenho do SciCumulus superou o Hadoop quando utilizou de 1 a 128 núcleos virtuais. Este comportamento se dá devido a duas razões principais. A primeira é que o Hadoop apresenta uma sobrecarga computacional consideravelmente alta para a geração das tarefas de *Map*. Esta fase da geração dos *Maps* consome um tempo considerável de cada execução de cada atividade no *workflow*. A segunda razão, é que o escalonamento efetuado no SciCumulus foi baseado no modelo de custos proposto e *cloud activities* que necessitavam de mais CPU foram escalonadas para máquinas virtuais mais potentes, enquanto que *cloud activities* que demandavam menos poder de CPU foram escalonadas para máquinas virtuais com menor capacidade. Por outro lado, o Hadoop fez um escalonamento aleatório sem considerar o peso computacional de cada *cloud activity* e a capacidade da máquina virtual.

Tabela 2 Tempos da execução do SciPhy para o cenário C1

Abordagem	*Execução*	*Quantidade de Núcleos Utilizados*							
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**
Hadoop	Clustalw	190624,00	103600,00	56000,00	34160,00	18788,00	10333,40	6200,04	3720,02
	Mafft	140017,41	74875,62	40040,44	21412,00	13917,80	9046,57	6061,20	4061,01
	Muscle	106555,64	63805,77	37096,38	22899,00	12594,45	6926,95	3809,82	2095,40
	ProbCons	125751,50	75300,30	45090,00	27000,00	14850,00	8910,00	5346,00	3207,60
	Kalign	174832,31	98220,40	55180,00	31000,00	17670,00	10071,90	5539,55	3046,75
	Experimento	**737780,87**	**415802,10**	**233406,82**	**136471,00**	**77820,25**	**45288,82**	**26956,61**	**16130,78**
SciCumulus	Clustalw	165023,00	90001,00	44520,00	25400,00	12954,00	7901,94	4346,07	2868,40
	Mafft	121400,00	66125,00	31452,00	18002,00	9181,02	5508,61	3084,82	2190,224131
	Muscle	88541,00	53173,80	29541,00	16101,00	8211,51	4680,56	2667,92	1467,355779
	ProbCons	106254,25	61660,80	34256,00	18054,00	9207,54	5432,45	3313,79	2021,414124
	Kalign	140874,00	81004,00	38744,00	19078,00	9729,78	6032,46	3317,85	2156,605737

	Experimento	**639176,00**	**351964,60**	**178513,00**	**96635,00**	**49283,85**	**29556,02**	**16730,46**	**10704,00**

Desta forma, uma vez que o valor de *csi* de máquinas virtuais do tipo *large* é maior do que o valor de *csi* de máquinas virtuais do tipo *micro*, o tempo total de execução é reduzido quando utilizamos o SciCumulus. Por exemplo, quando executamos o SciPhy com 64 núcleos virtuais, o SciCumulus necessita de 16.730,46 segundos em media enquanto que o Hadoop executa em 26.956,61 segundos. Essa diferença de desempenho, levou a um pico de melhora de 37,9% do SciCumulus sobre o desempenho do Hadoop.

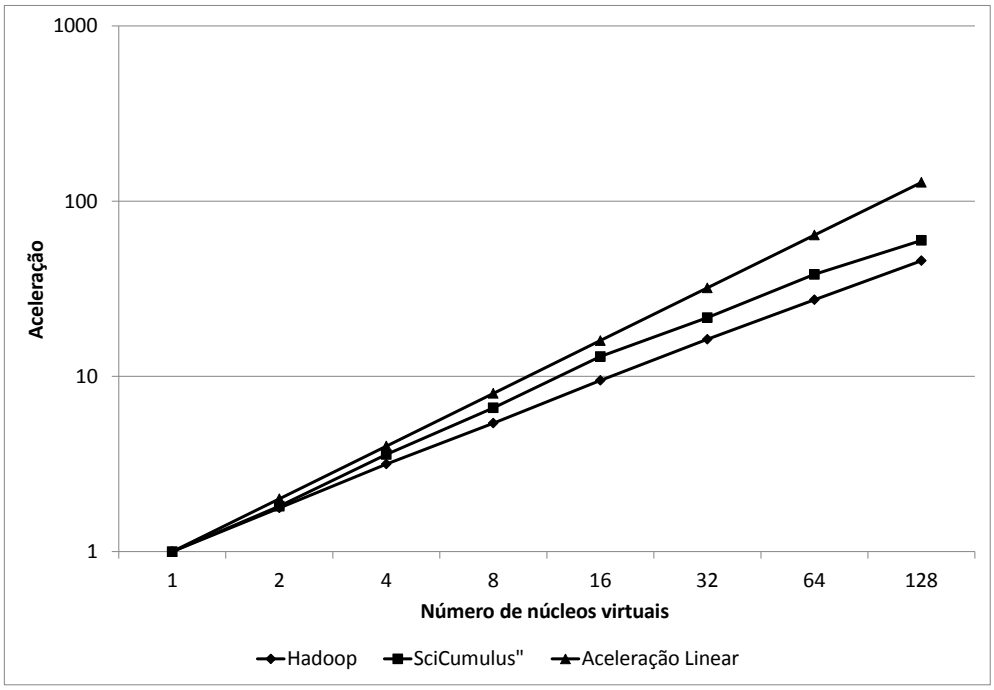


Figura 32 Aceleração para o cenário C1

A execução do SciPhy focando em desempenho nos levou a uma aceleração de 27,36 (no caso do Hadoop utilizando 64 núcleos disponíveis) e 38,20 (no caso do SciCumulus considerando 64 núcleos disponíveis). Ao analisar o gráfico de aceleração apresentado na Figura 32, podemos afirmar que o SciCumulus atingiu uma aceleração quase linear quando utilizamos entre 2 e 32 núcleos no processamento, e sofreu uma degradação pequena quando utilizou de 64 até 128 núcleos. A aquisição de mais de 64 núcleos virtuais para execução não pôde trazer maiores benefícios ou gerar um aumento considerável de desempenho, principalmente quando o número de *cloud activities* aproximou-se do número de núcleos envolvidos na computação. Isto se deu por motivos de implementação e da estrutura interna do SciCumulus na gerência das *cloud activities*.

Como para cada *cloud activity* um novo *workspace* (diretório no ambiente compartilhado) é criado, isso implica em aumentar a árvore de diretórios que deve ser gerenciada, gerando assim uma degradação no sistema operacional. Este problema já está sendo corrigido para a próxima versão do SciCumulus.

Em segundo lugar, ajustamos o modelo de custo para focar no custo monetário (cenário C2) definindo $\alpha_1 = 0{,}05$, $\alpha_2 = 0{,}05$ e $\alpha_3 = 0{,}9$. O tempo de execução e os resultados de aceleração para este cenário são apresentados na Figura 33 e Figura 34, respectivamente. Os dados brutos de execução podem ser também visualizados na Tabela 3. Diferentemente da execução do cenário C1 que possuía o foco no desempenho, neste caso optou-se por escalonar as *cloud activities* que necessitavam de maior poder de processamento para as máquinas virtuais do tipo *micro* (que são claramente mais baratas – 0,08 dólares por hora de utilização), enquanto que atividades menos computacionalmente intensivas foram escalonadas para máquinas virtuais do tipo *large* (que são mais caras – 0,35 dólares por hora de utilização).

No entanto, o SciCumulus não mostrou um desempenho melhor do que o Hadoop quando utilizamos de 2 a 128 núcleos virtuais, o que é compreensível já que o foco é no custo monetário ao invés de desempenho. O SciCumulus só superou o Hadoop quando foi utilizado apenas um núcleo, uma vez que existia apenas uma única opção de escalonamento de *cloud activities* e a sobrecarga de geração de *Maps* do Hadoop fez com que o mesmo tivesse um desempenho pior. De 2 até 128 núcleos o desempenho utilizando o SciCumulus foi degradado porque as máquinas virtuais menos poderosas foram escolhidas preferencialmente (já que elas são mais baratas) para executar *cloud activities* que demandavam mais CPU, enquanto que no Hadoop o escalonamento não leva em conta o custo monetário envolvido na execução.

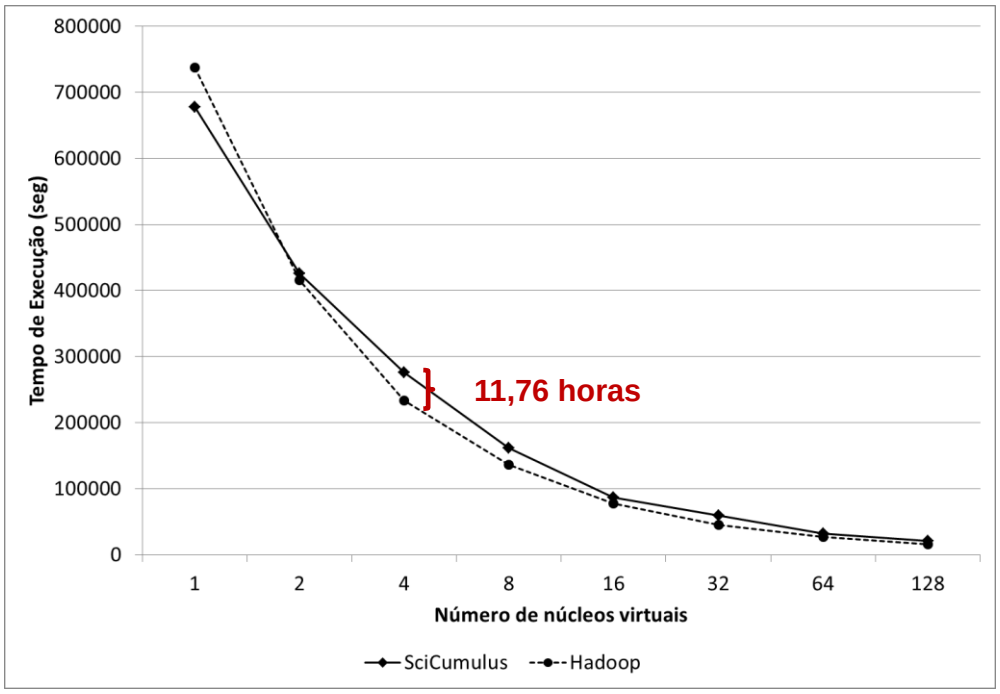


Figura 33 Tempo total de execução para o cenário C2

A abordagem com foco no custo monetário proposta pelo SciCumulus é principalmente benéfica no caso de estarmos executando o experimento em um provedor em que a janela de tempo é de 1 minuto ou em modelos de cobrança em que se pague apenas pelo utilizado e não por janela de tempo. Neste caso, tendemos a gastar menos porque as *cloud activities* computacionalmente intensivas são executadas em máquinas virtuais mais baratas e a tendência de termos tempo ocioso é reduzida.

Por outro lado, se a janela de tempo utilizada pelo provedor for de 1 hora, não gastaremos necessariamente menos dinheiro utilizando este modelo, pois já foi pago o período de uma hora, sendo este usado ou não. No entanto a vantagem é que quando o algoritmo de dimensionamento de recursos é executado, o mesmo tenderá a destruir máquinas virtuais mais caras, assim, fazendo com que algumas *cloud activities* computacionalmente menos intensivas cessem sua execução (essas *cloud activities* tem de ser escalonadas novamente). Neste caso, quanto menos "pesada" for a *cloud activity* a ser re-escalonada, melhor.

Tabela 3 Tempos da execução do SciPhy para o cenário C2

Abordagem	*Execução*	*Quantidade de Núcleos Utilizados*							
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**
Hadoop	Clustalw	190624,00	103600,00	56000,00	34160,00	18788,00	10333,40	6200,04	3720,02
	Mafft	140017,41	74875,62	40040,44	21412,00	13917,80	9046,57	6061,20	4061,01
	Muscle	106555,64	63805,77	37096,38	22899,00	12594,45	6926,95	3809,82	2095,40
	ProbCons	125751,50	75300,30	45090,00	27000,00	14850,00	8910,00	5346,00	3207,60
	Kalign	174832,31	98220,40	55180,00	31000,00	17670,00	10071,90	5539,55	3046,75

SciCumulus	**Experimento**	**737780,87**	**415802,10**	**233406,82**	**136471,00**	**77820,25**	**45288,82**	**26956,61**	**16130,78**
	Clustalw	227562,00	114521,00	58412,00	36874,00	23547,00	12400,00	7452,00	4005
	Mafft	170458,00	85412,00	41542,00	22415,00	15471,00	11015,00	7510,00	6005,00
	Muscle	145874,00	75412,00	38447,00	25874,00	13005,00	9046,00	4874,00	2547,00
	ProbCons	147878,00	85478,00	46012,00	29014,00	15874,00	13254,00	5002,00	3500,00
	Kalign	201879,00	110452,00	57080,00	38471,00	19005,00	14005,00	7541,00	4520,00
	Experimento	**678122,00**	**425789,00**	**275744,00**	**161547,00**	**86902,00**	**59720,00**	**32379,00**	**20577**

Quando executamos o SciPhy focando no custo monetário com 64 núcleos, o SciCumulus executou em 32.379,00 segundos em média enquanto que o Hadoop executou em 26.956,61 segundos. Devido aos fatores já explicados evidenciamos uma diferença "negativa" de 16,7% na comparação do desempenho. Essa execução levou a um aumento de velocidade de 27,36 (no caso do Hadoop considerando 64 núcleos virtuais disponíveis) e de 20,94 (no caso do SciCumulus considerando 64 núcleos virtuais disponíveis). Ao analisarmos a aceleração na Figura 34, podemos afirmar que o Hadoop alcançou uma melhor aceleração do que SciCumulus, e ambos sofreram uma degradação pequena, quando utilizamos de 64 até 128 núcleos virtuais. No entanto, a diferença de aceleração é aceitável já que o foco principal aqui é o custo monetário e não o desempenho propriamente dito. Essa análise foi feita para mostrar que mesmo focando no custo monetário não negligenciamos o aumento de desempenho na execução paralela do *workflow*.

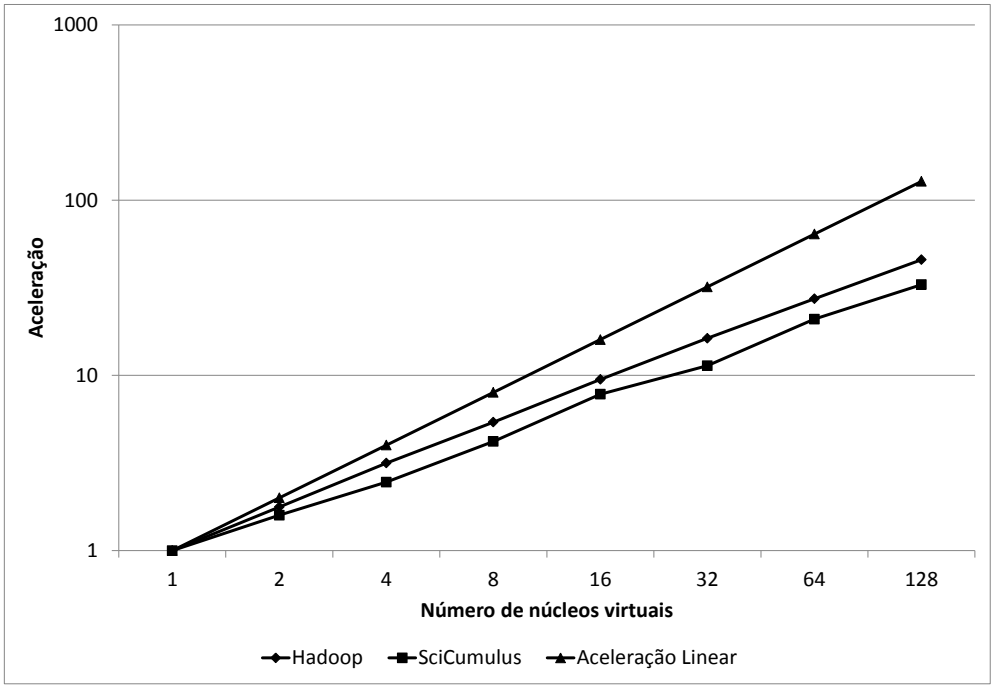


Figura 34 Aceleração para o cenário C2

Em terceiro lugar, ajustamos o modelo de custos para manter o foco na confiabilidade (cenário C3) definindo $\alpha_1 = 0,05$, $\alpha_2 = 0,9$ e $\alpha_3 = 0,05$. O tempo de execução e os resultados aceleração são apresentados na Figura 35 e Figura 36, respectivamente. Além disso, os dados brutos de tempo de execução são apresentados na Tabela 4. Uma vez que estamos nos concentrando na confiabilidade, temos que calcular a probabilidade de ocorrer um determinado número de falhas nas máquinas virtuais envolvidas na execução.

Como podemos observar, consultando os dados proveniência no repositório do SciCumulus, máquinas virtuais do tipo *micro* degradam aproximadamente 3,4 vezes mais rápido do que máquinas virtuais do tipo *large*. Além disso, verificamos que a probabilidade de uma *cloud activity* falhar em uma máquina do tipo *micro* chega a 9,1% enquanto que em máquinas do tipo *large* esta probabilidade de falha é de 0,4%. Desta forma, o modelo de custo e o algoritmo de escalonamento de *cloud activities* escalonaram *cloud activities* computacionalmente mais intensivas para máquinas virtuais do tipo *large* e *cloud activities* computacionalmente menos intensivas para máquinas virtuais do tipo *micro*.

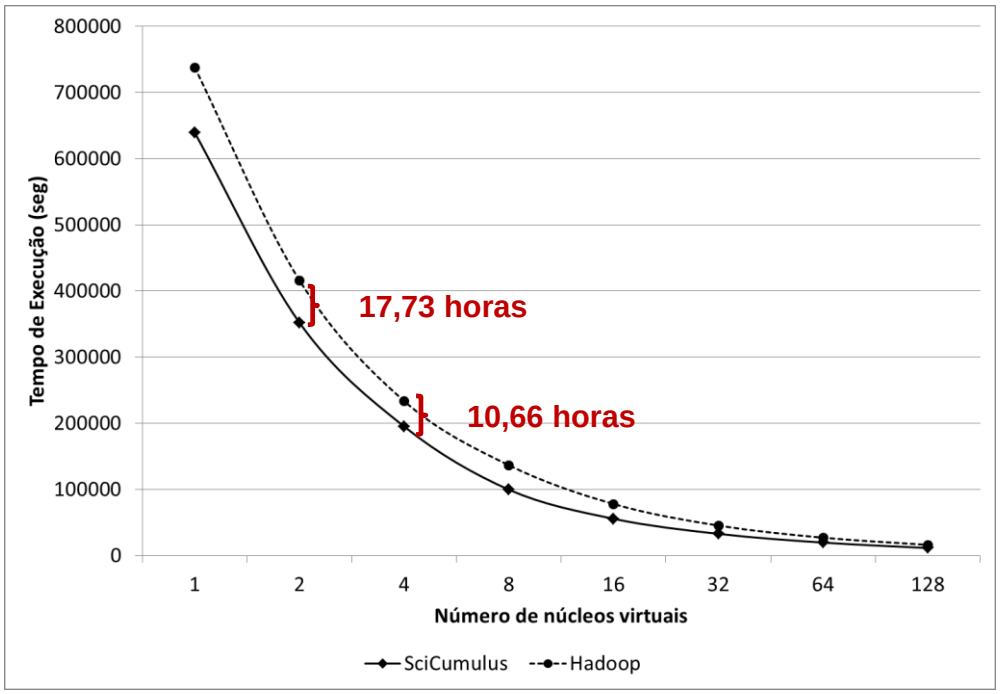


Figura 35 Tempo total de execução para o cenário C3

De fato, utilizando o foco em confiabilidade evitamos re-executar *cloud activities* computacionalmente intensivas, pois elas são escalonadas para máquinas que apresentam uma probabilidade de falha mais baixa. Desta forma, uma vez que máquinas

virtuais do tipo *large* são mais confiáveis do que máquinas virtuais do tipo *micro*, o comportamento deste escalonamento se degenerou ao primeiro cenário (C1). Da mesma forma que o primeiro cenário o SciCumulus superou o Hadoop quando utilizamos de 2 a 128 núcleos virtuais na execução.

Tabela 4 Tempos da execução do SciPhy para o cenário C3

Abordagem	*Execução*	*Quantidade de Núcleos Utilizados*							
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**
Hadoop	Clustalw	190624,00	103600,00	56000,00	34160,00	18788,00	10333,40	6200,04	3720,02
	Mafft	140017,41	74875,62	40040,44	21412,00	13917,80	9046,57	6061,20	4061,01
	Muscle	106555,64	63805,77	37096,38	22899,00	12594,45	6926,95	3809,82	2095,40
	ProbCons	125751,50	75300,30	45090,00	27000,00	14850,00	8910,00	5346,00	3207,60
	Kalign	174832,31	98220,40	55180,00	31000,00	17670,00	10071,90	5539,55	3046,75
	Experimento	**737780,87**	**415802,10**	**233406,82**	**136471,00**	**77820,25**	**45288,82**	**26956,61**	**16130,78**
SciCumulus	Clustalw	140587,00	72858,14	42520,79	27510,89	13040,00	8112,00	4806,42	2643,53
	Mafft	85478,00	45902,87	29555,32	15478,00	10254,00	5877,00	3482,08	1915,14
	Muscle	88201,00	47004,34	25140,87	16520,78	9542,00	4998,26	3293,02	1811,16
	ProbCons	111205,36	59898,86	32478,00	19150,00	10100,58	6665,58	3852,00	2332,58
	Kalign	127002,59	69707,75	41004,00	19954,00	12547,00	7102,85	4125,00	2661,67
	Experimento	**639176,00**	**351964,60**	**195000,00**	**99541,00**	**55483,58**	**32755,69**	**19558,52**	**11364,08**

Uma vez que a probabilidade de falha de máquinas virtuais do tipo *large* é menor do que a probabilidade de falha de máquinas virtuais do tipo *micro*, o tempo total de execução é reduzido semelhantemente ao cenário C1. Por exemplo, quando executamos o SciPhy com 64 núcleos, o SciCumulus executou em 19.558,52 segundos enquanto que o Hadoop executou em 26.956,61 segundos em média. Foi constatado um pico de melhora de 34,7% no desempenho entre as abordagens. Essa execução focando na confiabilidade levou a um aumento de velocidade (aceleração) de 27,36 (no caso do Hadoop considerando 64 núcleos disponíveis) e de 32,68 (no caso do SciCumulus considerando 64 cores disponíveis). Ao analisarmos a aceleração na Figura 36, podemos afirmar que SciCumulus atingiu uma aceleração sublinear quando utilizou de 2 a 16 núcleos, e sofreu uma degradação pequena quando utilizou de 32 até 128 núcleos, pelos mesmo motivos apresentados anteriormente. Em relação à quantidade de falhas contabilizadas, comparamos a execução com foco em confiabilidade com o escalonamento aleatório no SciCumulus. Com o escalonamento aleatório o SciCumulus apresentou 3,70% de falhas em máquinas virtuais em média (30 falhas de *cloud activities* de um total de 810 *cloud activities* executadas em cada execução para cada método de MSA). Com a abordagem focando em confiabilidade, esse percentual de falhas foi reduzido para 0,5% em média (4 falhas de *cloud activities* de um total de 810 *cloud activities* executadas em cada execução para cada método de MSA).

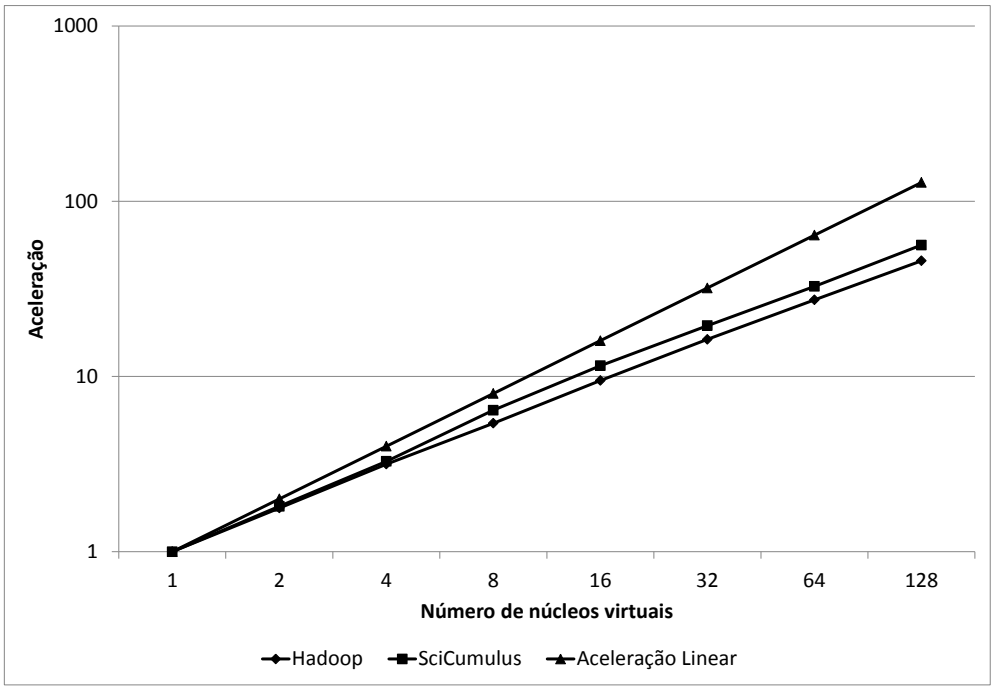


Figura 36 Aceleração para o cenário C3

No último cenário, foi ajustado o modelo de custos para equilibrar o foco em todos os critérios (cenário C4) definindo $\alpha_1 = 0{,}33$, $\alpha_2 = 0{,}33$ e $\alpha_3 = 0{,}33$. O tempo de execução e os resultados de aceleração são apresentados na Figura 37 e Figura 38, respectivamente. Os dados brutos são apresentados na Tabela 5. Uma vez que atribuímos pesos semelhantes para cada critério, o desempenho alcançado foi muito semelhante ao desempenho do Hadoop, como pode ser visualizado na Figura 37. A principal diferença no desempenho foi devida ao *overhead* que foi imposto pelo Hadoop para gerar *Maps* e *Reduces*.

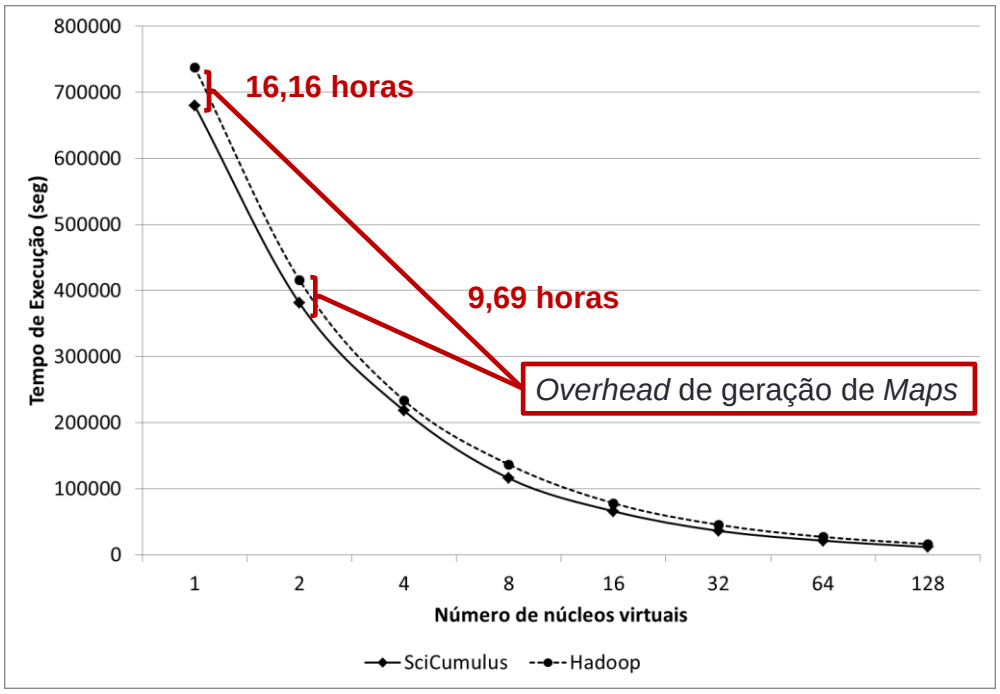


Figura 37 Tempo total de execução para o cenário C4

Quando executamos o SciPhy utilizando 64 núcleos e utilizando os parâmetros igualmente ponderados para o modelo de custo, o SciCumulus executou em 21.288,00 segundos enquanto que o Hadoop executou em 26.956,61 segundos. É uma diferença de 21,02% no desempenho global. Essa execução levou a um aumento de velocidade (aceleração) de 27,36 (no caso do Hadoop considerando 64 núcleos disponíveis) e 31,92 (no caso do SciCumulus considerando 64 núcleos disponíveis). Ambos SciCumulus e Hadoop apresentaram resultados muito similares neste cenário (C4).

Tabela 5 Tempos da execução do SciPhy para o cenário C4

Abordagem	*Execução*	*Quantidade de Núcleos Utilizados*							
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**
Hadoop	Clustalw	190624,00	103600,00	56000,00	34160,00	18788,00	10333,40	6200,04	3720,02
	Mafft	140017,41	74875,62	40040,44	21412,00	13917,80	9046,57	6061,20	4061,01
	Muscle	106555,64	63805,77	37096,38	22899,00	12594,45	6926,95	3809,82	2095,40
	ProbCons	125751,50	75300,30	45090,00	27000,00	14850,00	8910,00	5346,00	3207,60
	Kalign	174832,31	98220,40	55180,00	31000,00	17670,00	10071,90	5539,55	3046,75
	Experimento	**737780,87**	**415802,10**	**233406,82**	**136471,00**	**77820,25**	**45288,82**	**26956,61**	**16130,78**
SciCumulus	Clustalw	174999,83	96153,75	54945,00	29700,00	15660,00	8613,00	5002,00	2868,40
	Mafft	118799,34	65274,36	37514,00	18104,00	11478,00	6312,90	3845,00	2452,00
	Muscle	102074,49	58663,50	34918,75	18875,00	10547,00	5800,85	3552,00	1605,00
	ProbCons	127249,03	74852,37	41354,90	22354,00	12478,00	6862,90	3987,00	2021,41
	Kalign	156462,81	85968,58	49124,90	26554,00	15557,00	8556,35	4902,00	2547,00
	Experimento	**639176,00**	**351964,60**	**195000,00**	**99541,00**	**55483,58**	**32755,69**	**19558,52**	**11364,08**

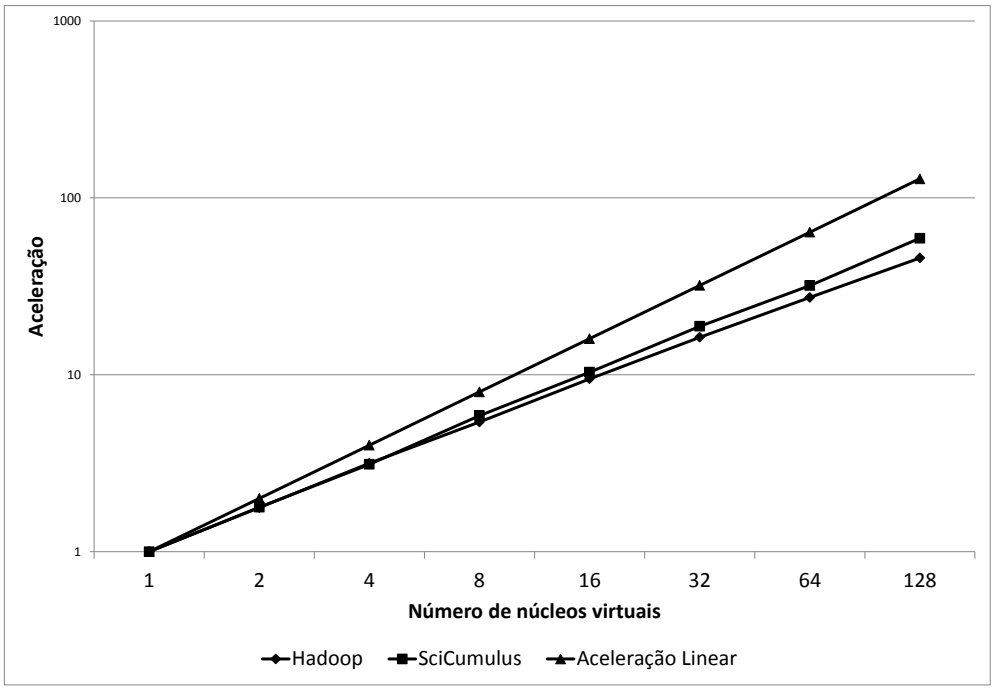


Figura 38 Aceleração para o cenário C4

7.6 Análise de Escalabilidade

A fim de analisar a sobrecarga de processamento imposta pelo SciCumulus quando processamos grandes volumes de dados, executamos um experimento para analisar a escalabilidade do SciCumulus de acordo com o aumento do volume de dados processados. Para este experimento, executamos o SciPhy limitando o tamanho do *cluster* a 16 núcleos virtuais, e variando o número de arquivos de entrada e os cenários de execução (*i.e.* os pesos associados a α_i no modelo de custo).

Este experimento consumiu de 100 a 1.600 arquivos multi-fasta de entrada (no caso dos 1.600 arquivos, processamos a coleção completa). Os resultados de escalabilidade são apresentados na Figura 39. Como podemos observar, como esperado, para cada um dos cenários (C1, C2, C3, C4 e Hadoop) conforme o volume de dados a ser processado no SciPhy aumenta o tempo de execução aumenta de forma análoga, mudando apenas a taxa de crescimento.

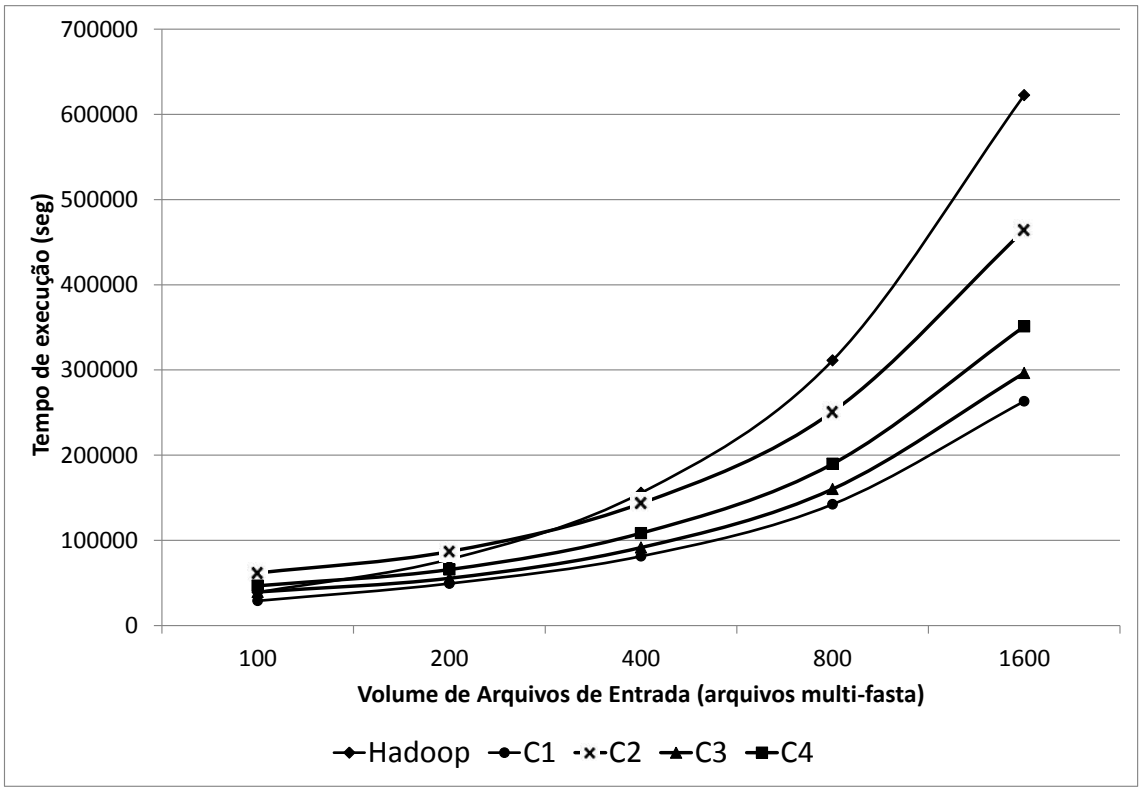


Figura 39 Estudo de escalabilidade de acordo com o aumento do volume de dados

Entretanto, pudemos comprovar que a sobrecarga imposta pelo Hadoop, neste caso, também acompanha o crescimento do volume de dados e consequentemente a quantidade de tarefas a ser processada, o que pode ser preocupante em experimentos em maior escala dos que os analisados nesta tese.

Os cenários C1, C3 e C4 apresentaram uma boa escalabilidade, alcançando uma escalabilidade sublinear uma vez que o tempo de execução teve um aumento quase que proporcional ao aumento no volume de dados (ao dobrarmos o volume de dados processados, o tempo de execução foi um pouco acima do dobro). No entanto, quando executamos o SciPhy utilizando o cenário C2 (foco no custo monetário) o tempo de execução aumenta gradativamente mais uma vez que distribuímos as *cloud activities* computacionalmente mais pesadas para máquinas com menor capacidade de processamento. Entretanto, mesmo com uma taxa de crescimento do tempo consideravelmente maior do que as outras políticas no SciCumulus, o cenário C2 ainda é melhor do que a execução com o Hadoop, que é penalizado pelo aumento acumulativo da sobrecarga de computação de geração de tarefas de *Map* e *Reduce*.

7.7 Análise de Custo Monetário

A fim de verificar o custo monetário envolvido na execução dos cinco cenários (Hadoop, C1, C2, C3 e C4), analisamos os dados de proveniência gerados e calculamos o custo final utilizando duas formas de cobrança e pagamento. Na primeira forma definimos o modelo de cobrança baseado em uma janela de tempo de 1 hora (método de pagamento da Amazon EC2), enquanto que na segunda forma definimos o modelo de cobrança baseado em cobrança por processamento (método de pagamento pré-pago do GoGrid em que se desconta apenas o tempo utilizado efetivamente de processamento da máquina por minuto de uso).

Para este experimento, em cada execução variamos a quantidade utilizada máquinas virtuais de cada tipo conforme explicado na última seção. Para cada execução do *workflow* SciPhy, metade das máquinas virtuais instanciadas é do tipo *large* enquanto que a outra metade é do tipo *micro*. O custo monetário total não varia muito durante a execução de quatro dos cinco cenários existentes (Hadoop, C1, C3 e C4), porque estas execuções não estão focadas em reduzir custos monetários, ou seja, o escalonamento é realizado sem considerar o custo monetário como objetivo principal a ser minimizado.

No entanto, é importante destacar que o custo monetário alcançado nestes cenários não é de forma alguma proibitivo para a execução do experimento, ou seja, a maioria dos cientistas é capaz de arcar com as despesas associadas a estas execuções. Neste caso, estamos mais interessados em analisar o custo monetário do segundo cenário (C2), onde focamos na redução do mesmo. Neste cenário, o custo monetário total varia bastante de acordo com a política de cobrança escolhida conforme apresentado na Tabela 6 e na Figura 40.

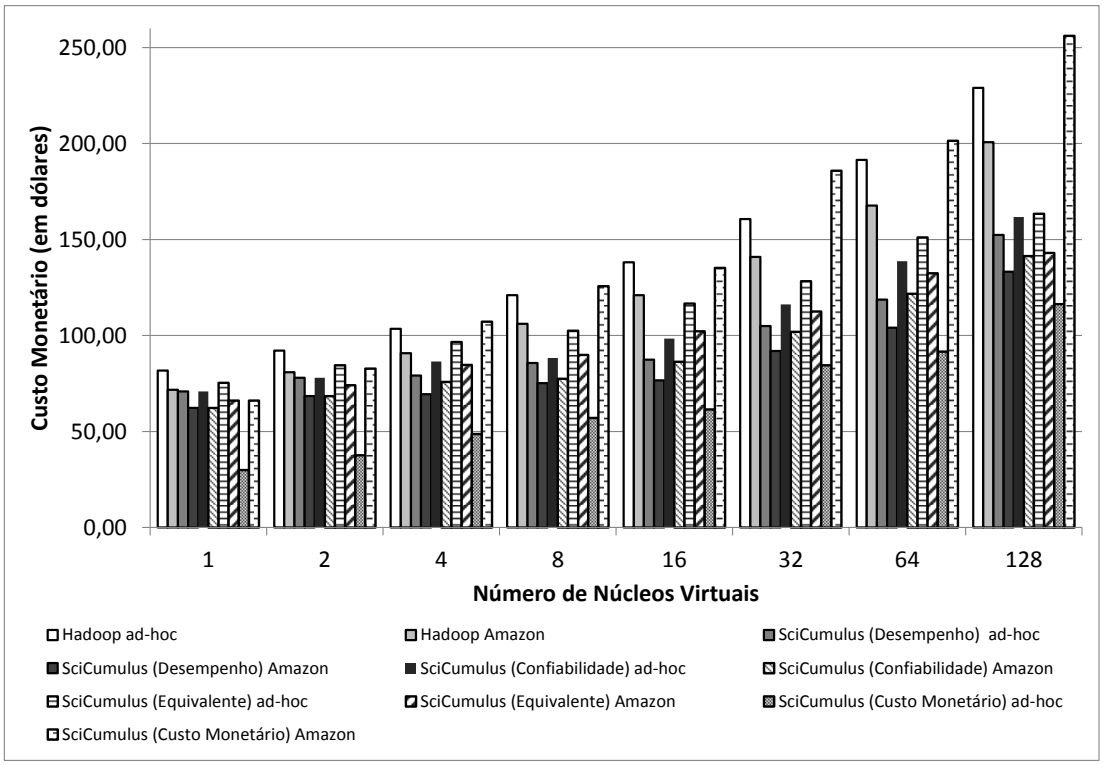


Figura 40 Análise de custo monetário para os cenários C1, C2, C3, C4 e Hadoop

Esta diferença de custo monetário é observada porque no caso em que estamos executando o experimento em um provedor em que a cobrança é realizada por minuto efetivamente processado, vamos escalonar *cloud activities* que são computacionalmente intensivas para máquinas virtuais que são mais baratas e pagar apenas por este processamento. Desta forma, grande parte do tempo de execução do *workflow* é gasta em máquinas virtuais do tipo *micro* (ou equivalente) e atividades computacionalmente menos intensivas apenas são executadas em máquinas virtuais de grande porte (caso existam). Nesse caso, só pagaremos pelo que efetivamente processamos e mesmo que uma maquina do tipo *large* não esteja sendo usada, não será cobrada nenhuma taxa de uso. Além disso, uma vez que atividades computacionalmente menos intensivas executam em alguns poucos segundos ou minutos, a cobrança por processamento é benéfica, pois há pouco (se não nenhum) desperdício de tempo.

Tabela 6 Custo monetário para o cenário C2

Número de Núcleos Virtuais	*Custo Monetário*[9]	
	Janela de Tempo de 1 Hora	**Janela de Tempo de 1 Minuto**
1	66,15	65,15
2	82,79	37,61
4	107,23	48,72
8	125,65	57,09
16	135,18	61,44
32	185,80	84,66
64	201,47	91,58
128	256,07	116,35

Por outro lado, se estamos utilizando o modelo de cobrança do Amazon EC2 os benefícios são menos tangíveis. Para um melhor entendimento, vamos imaginar o pior cenário que é quando uma atividade computacionalmente menos intensiva foi escalonada para uma máquina virtual de grande porte e ela não consome a janela de tempo completa. Esta atividade computacionalmente menos intensiva tem a duração de 16 minutos, mas o cientista já pagou por uma hora neste caso, desta forma, existe um desperdício de tempo e dinheiro caso o SciCumulus não escalone novas *cloud activities* para esta máquina. Além disso, no Amazon EC2, mesmo que o SciCumulus não escalone *cloud activities* para uma determinada máquina, estaremos pagando por hora desta máquina da mesma maneira, *i.e.* máquinas que estejam ociosas no modelo de cobrança do Amazon EC2 sofrem tarifação da mesma maneira. Estas características fazem com que a abordagem focando em custo tenha um comportamento ruim com este modelo de cobrança do Amazon, porém apresenta uma vantagem que é quando acontece o redimensionamento de recursos. Uma vez que estamos focando em custo monetário, o dimensionador de recursos fará ajustes para atender o orçamento informado e, quando o orçamento está prestes a ser alcançado, o SciCumulus deixa de escalonar *cloud activities* para máquinas virtuais mais caras pois estas tem maior chance de serem desligadas a fim de reduzir o custo monetário total. Caso uma máquina precise ser desligada a *cloud activity* que estava executando deverá ser reexecutada. Porém como existe a tendência de ser uma *cloud activity* computacionalmente pouco intensiva, o impacto na execução será minimizado.

[9] Em dólares americanos

7.8 Análise de Agrupamento de *Cloud Activities*

Nas execuções apresentadas até agora, o algoritmo de agrupamento de *cloud activities* não agrupou as mesmas de diferentes formas de acordo com o tipo de máquina e com o desempenho da mesma durante a execução, pois como o Hadoop não realiza este agrupamento, a comparação não seria justa. Assim, não conseguimos mensurar por meio dos experimentos já mostrados qual o impacto que o agrupamento de *cloud activities* na execução do *workflow*.

Nos últimos anos, tem sido discutido pela comunidade acadêmica a questão do agrupamento de tarefas em processamento paralelo (Iosup *et al.* 2007) para diminuição da sobrecarga de comunicação e transferência de dados. Seguindo os estudos prévios realizados por outros autores, adotamos a métrica de vazão de processamento de *cloud activities* a fim de apresentar a melhoria no desempenho das *cloud activities* alcançado por meio do agrupamento para mensurar a melhora (ou piora) no desempenho. Assim, neste estudo forçamos a utilização de determinados tamanhos fixos dos agrupamentos (de 1 a 16 *cloud activities* por grupo). Desta forma, executamos o SciPhy com modelo de custo focado em desempenho (α_1 = 0,9, α_2 = 0,05 e α_3 = 0,05) com 16 núcleos virtuais (equivalente a 8 máquinas virtuais do tipo *large*). A Figura 41 nos mostra que a vazão aumenta de aproximadamente 21,9 *cloud activities* por hora, sem o agrupamento, para um pico de vazão de 32,1 *cloud activities* por hora, com o agrupamento.

O desempenho do SciCumulus diminui após cerca de 5 *cloud activities* por agrupamento. Após uma ligeira queda com tamanho de agrupamento 6, se mantém praticamente constante até o máximo de tamanho de agrupamento configurado no experimento (16 *cloud activities* por grupo). Atribuímos essa queda ao problema de agrupar *cloud activities* que não tem muita relação de similaridade. Por exemplo, a proposta do agrupamento é reunir em uma mesma *cloud activity* outras instâncias que consomem dados semelhantes de forma a evitar transferências desnecessárias e sobrecarga na comunicação. Desta forma, o SciCumulus agrupa as *cloud activities* que preferencialmente tenham necessidades comuns e depois tenta agrupar as *cloud activities* que consomem dados não compartilhados, porém que não sejam demasiadamente grandes para evitar a sobrecarga de inicialização.

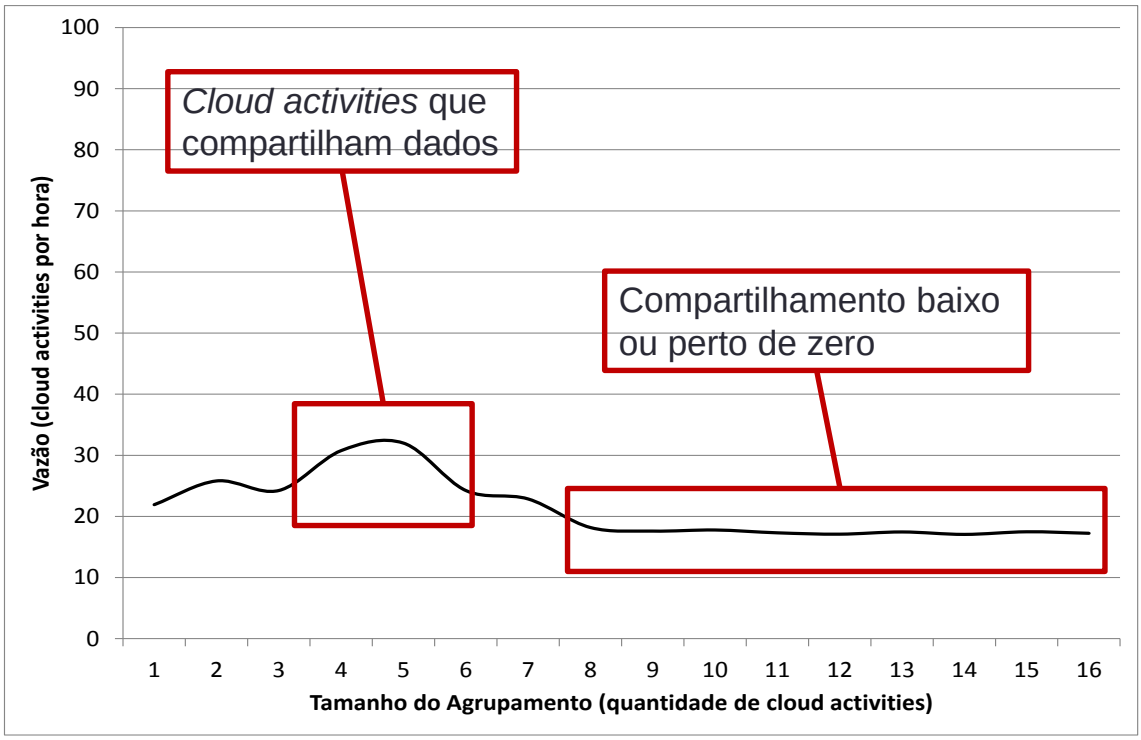


Figura 41 Análise do agrupamento das *cloud activities*

Entretanto, quando forçamos a criação de agrupamentos com tamanhos pré-definidos como neste experimento, corremos o risco de agrupar *cloud activities* que não compartilham dados e que possuem dados grandes para serem consumidos. Avaliando os dados biológicos de entrada, e o repositório de proveniência, pudemos constatar que foram produzidas diversas *cloud activities* com dados em comum o que fazia com que pudessem ser agrupadas idealmente em grupos de 4 a 5 *cloud activities*. Acima desse valor de tamanho de grupo o SciCumulus não conseguia mais aglutinar *cloud activities* com dados compartilhados no mesmo agrupamento o que acabou gerando o efeito contrário ao desejado (aumento da sobrecarga), desta forma não apresentando ganho de desempenho.

7.9 Análise de Desempenho da Execução Adaptativa

A fim de analisar o desempenho da execução adaptativa (na qual a quantidade de recursos varia de acordo com a previsão de demanda) no SciCumulus, realizamos uma série de experimentos reais baseados no *workflow* SciPhy. Estamos especialmente interessados em comparar o comportamento e o desempenho do SciCumulus utilizando a abordagem adaptativa e utilizando o algoritmo de escalonamento sem dimensionamento da quantidade de recursos utilizados, *i.e.* sem a execução do

algoritmo de dimensionamento de recursos. Os experimentos executados fixaram o número de arquivos multi-fasta de entrada consumidos em 100 e usaram apenas máquinas virtuais do tipo *large* (m1.large).

Como este experimento visa analisar a abordagem adaptativa o algoritmo de dimensionamento de recursos aumenta e reduz a quantidade de recursos utilizados dependendo da demanda, executamos o SciPhy em diversos cenários. Por motivos de espaço, escolhemos um cenário significativo e o apresentamos na Figura 42. Neste caso em particular pretende-se analisar o desempenho de acordo com os seguintes parâmetros de entrada: prazo de execução de 10 horas (36.000 segundos) e orçamento limite de US$ 150,00. O foco do modelo de custo foi no desempenho. Nesta execução, o número de máquinas virtuais aumenta e diminui ao longo do tempo para cumprir o prazo e o orçamento estipulados. Além disso, as máquinas virtuais podem ser destruídas sem interferência do SciCumulus.

Este cenário de destruição de máquinas virtuais sem o controle do usuário é encontrado quando usamos *spot instances* da Amazon EC2 (que pode ser destruída pelo provedor de nuvem sem aviso prévio e justamente por isso são mais baratas). Foram executados três casos diferentes neste experimento. O primeiro caso foi o que chamamos de "cenário ideal". Neste caso, assumimos que conhecíamos *a priori* que se executássemos o SciPhy utilizando 16 núcleos do início ao fim da execução poderíamos cumprir o prazo e o orçamento limite com o melhor desempenho possível. Desta forma, o "cenário ideal" começa e termina utilizando 16 núcleos. Por outro lado, nos outros dois casos, a quantidade de máquinas virtuais aumenta e diminui gradualmente. No segundo caso consideramos que não existem dados de proveniência disponíveis. Desta forma, o algoritmo de dimensionamento de recursos tem que esperar os dados de proveniência serem gerados em tempo real para poder calcular o desempenho e assim estimar a quantidade necessária de recursos. Na verdade, o algoritmo de dimensionamento de recursos apresenta uma latência até começar a aumentar a quantidade de máquinas virtuais envolvidas na execução.

No terceiro caso, assumimos que existem dados de proveniência disponíveis (de execuções passadas) no repositório desde o início da execução para realizar uma melhor estimativa. Desta forma, o algoritmo de dimensionamento de recursos começa a aumentar o número de máquinas virtuais mais cedo do que quando não há proveniência disponível. O algoritmo de dimensionamento de recursos escala os recursos

gradualmente porque as duas primeiras atividades do SciPhy são atividades computacionalmente menos intensivas (programa de alinhamento e Readseq). Quando alcançamos a terceira atividade (modelgenerator que é uma atividade computacionalmente intensiva) o algoritmo escala o número de máquinas virtuais mais rapidamente para poder atender a demanda. Podemos observar na Figura 42 que a quantidade total de máquinas virtuais aumenta e diminui gradualmente quando utilizamos a abordagem adaptativa do SciCumulus. No entanto, neste caso, o desempenho no cenário ideal (estático) supera o escalonamento adaptativo em 3,2% e 8,1% nos dois casos em que proveniência está disponível e quando não está, respectivamente. Estes resultados são muito próximos do cenário ideal, e cuja quantidade de núcleos a utilizar não é conhecida de antemão pelos cientistas.

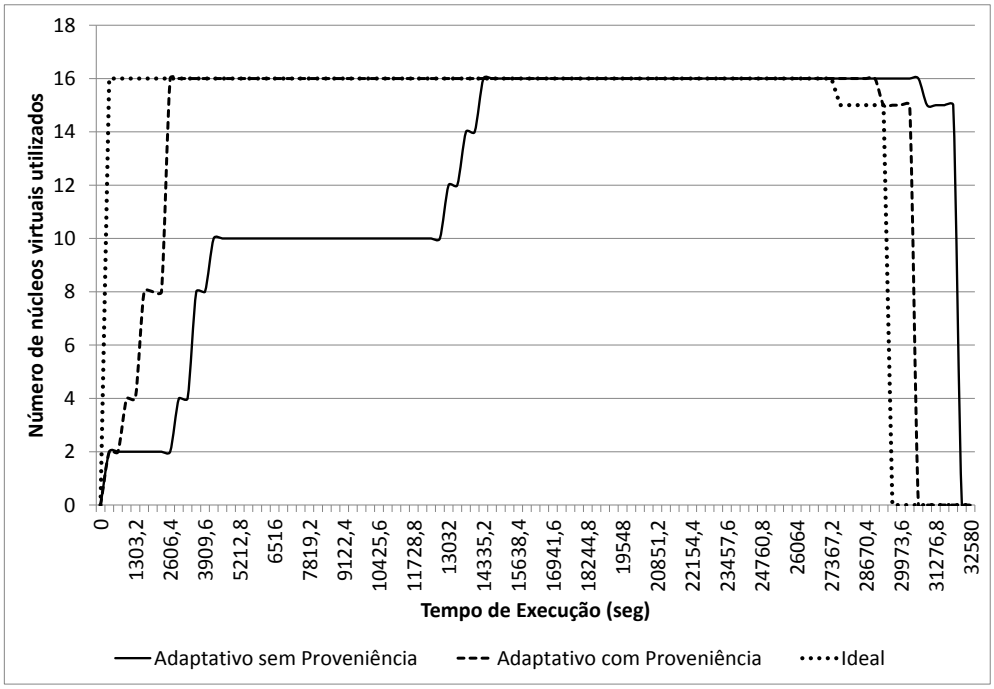


Figura 42 Acompanhamento da execução adaptativa

Apesar de a abordagem adaptativa com dimensionamento de recursos oferecer vantagens, ela também impõe uma sobrecarga que não pode ser descartada. Cada verificação que é realizada pelo SciCumulus antes de escalonar uma *cloud activity* ou de dimensionar os recursos implica em ociosidade por parte dos núcleos. Esta ociosidade deve ser analisada e não pode ser negligenciada.

Para tal, analisamos também o percentual de ociosidade dos núcleos das máquinas virtuais durante a execução do *workflow* SciPhy com a mesma configuração de experimento apresentada no início da seção (Figura 43). Uma vez que estamos

utilizando o escalonamento adaptativo, vários cálculos devem ser realizados (cálculo do modelo de custo, a estimativa para o número de máquinas virtuais a serem utilizadas, etc.) e o SciCumulus leva mais tempo para inicializar uma nova *cloud activity*. Desta forma, ao utilizar a abordagem adaptativa, o nível de ociosidade das máquinas virtuais tende a ser maior do que quando se usa uma abordagem estática onde não ocorrem mudanças. No entanto, essa diferença de nível de ociosidade é aceitável. A ociosidade média verificada no exemplo da Figura 42 foi de 11,34%, 12,60% e 14,26%, quando executamos utilizando um cenário estático, a abordagem adaptativa com proveniência e a abordagem adaptativa sem proveniência, respectivamente.

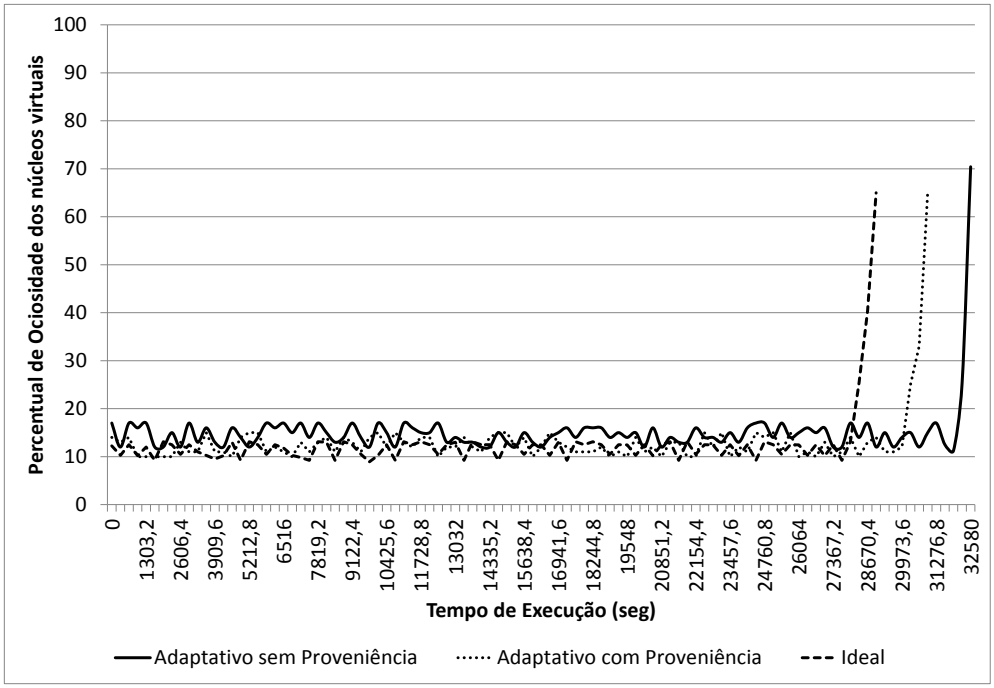


Figura 43 Percentual de ociosidade na execução adaptativa do SciCumulus

A ocorrência de um desbalanceamento da execução provoca uma inatividade dos núcleos computacionais assim que eles terminam a execução de suas últimas *cloud activities*. O fenômeno é proporcionalmente agravado à medida que o *workflow* aumenta o seu tamanho e o período de ociosidade é acumulado ao longo das atividades do *workflow*, conforme apresentado na Figura 43. Os níveis de ociosidade aumenta significativamente no final da execução, pois o SciCumulus apenas libera as máquinas virtuais ao fim da execução do *workflow*, fazendo assim com que algumas máquinas virtuais fiquem ociosas quando o número de *cloud activities* for menor do que o número de núcleos computacionais disponíveis.

7.10 Análise de Desempenho Utilizando o SciCumulus-ECM

Apesar da abordagem adaptativa do SciCumulus conseguir distribuir melhor as *cloud activities* de acordo com os recursos existentes e dimensionar a quantidade de recursos envolvidos na execução, ela ainda pode ser otimizada em vários sentidos. Um dos pontos de melhoria se refere à otimização da coleção inicial de máquinas virtuais que utilizamos. No exemplo da Figura 42 iniciamos a execução com nenhuma máquina virtual disponível. O SciCumulus então vai aumentando a quantidade de recursos gradativamente de acordo com a previsão de demanda baseada nas consultas de proveniência. Entretanto, esse aumento gradual tem uma sobrecarga de processamento embutida, pois perdemos tempo aumentando (ou diminuindo) a quantidade de máquinas envolvidas (uma vez que as máquinas tem um custo de inicialização que não pode ser negligenciado).

Essa sobrecarga poderia ser diminuída caso o conjunto inicial de máquinas virtuais já fosse suficientemente perto do ideal para o experimento. Como apresentado no Capítulo 6 desta tese, essa é a ideia do serviço SciCumulus-ECM (Viana *et al.* 2011b, 2011a): simular a execução do *workflow a priori* e criar um conjunto inicial de máquinas virtuais de modo que a adaptação posterior (caso necessária) seja menos custosa. Um exemplo da utilização do SciCumulus-ECM é apresentado na Figura 44.

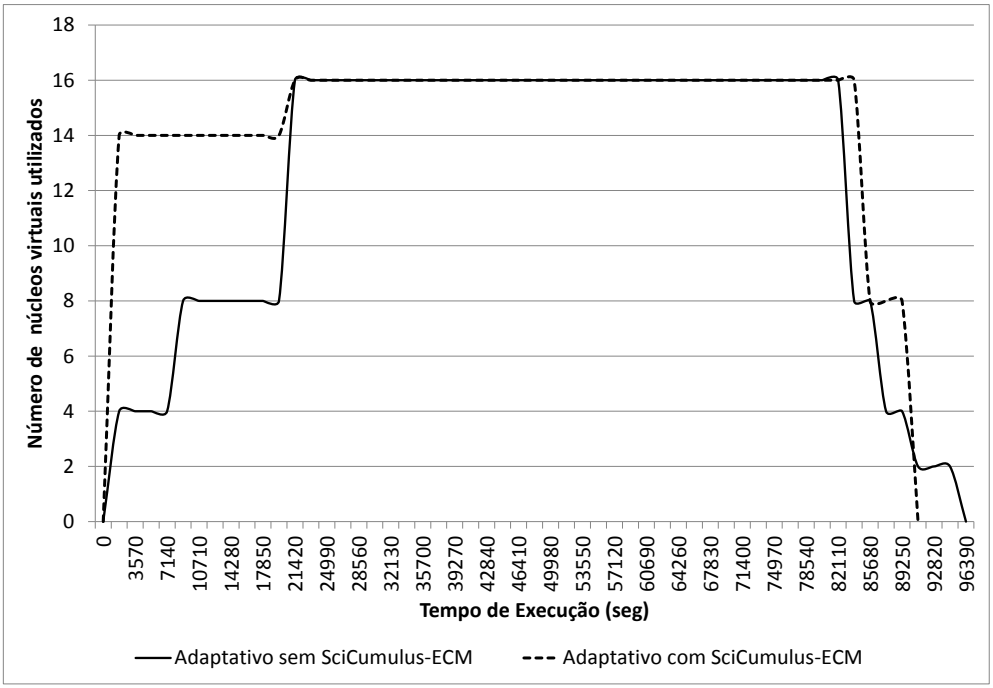


Figura 44 Análise experimental da utilização do SciCumulus-ECM

Neste exemplo, executamos o *workflow* SciPhy com foco em desempenho e processando 400 arquivos multi-fasta de entrada e limitando o orçamento em US$ 75,00. Sem a utilização do SciCumulus-ECM partimos do zero e o SciCumulus escala a quantidade de máquinas conforme analisa os dados de proveniência. Os aumentos de quantidade de máquinas ocorrem na análise de desempenho das atividades de alinhamento, de conversão (readseq) e de geração do modelo (modelgenerator).

Com a utilização do SciCumulus-ECM já iniciamos a execução com 14 máquinas virtuais e este valor só é aumentado para 16 no momento do término da primeira execução de *cloud activity* associada a atividade modelgenerator (que é uma atividade computacionalmente intensiva). Podemos perceber que existe uma diferença de 6% entre a execução com o SciCumulus-ECM (91.035,00 segundos) e sem a utilização do SciCumulus-ECM (96.390,00 segundos). Apesar de percentualmente a diferença não ser grande, ela tende a escalar de acordo com o aumento da complexidade do *workflow*, *i.e.* quanto maior for o tempo de execução do *workflow*, maior será a diferença de tempo absoluta. É importante ressaltar que o tempo de otimização gasto com o algoritmo genético do SciCumulus-ECM não se encontra contabilizado neste gráfico. Porém, em todos os testes realizados, para processar as 200 gerações no algoritmo genético o SciCumulus-ECM levou em média 5,5 minutos, tempo este que pode ser considerado desprezível frente ao tempo total de execução do *workflow* e frente ao ganho na execução proporcionado pelo SciCumulus-ECM. Dados mais completos sobre o SciCumulus-ECM são apresentados por Viana *et al.* (2011b, 2011a) em seu trabalho.

7.11 Análise do SciLightning

Com o intuito de analisar a capacidade do SciLightning para monitorar a execução dos *workflows* executados em paralelo pelo SciCumulus, executamos um experimento controlado visando analisar se todas as situações a serem monitoradas foram efetivamente capturadas pelo SciLightning. Nesse experimento executamos o SciPhy com 16 núcleos virtuais, focando em desempenho e consumindo 200 arquivos multi-fasta de entrada. Dessa coleção de arquivos, 15% (30 arquivos no total) foram intencionalmente modificados para conter menos de 3 sequências biológicas cada. Desta forma, a atividade de alinhamento pôde ser executada com êxito, mas as atividades de geração do modelo e das árvores geraram erros que deveriam ser informados aos cientistas.

Além do monitoramento de erros, foram configurados pontos de monitoramento de fim de atividade (*i.e. milestones*) em cada atividade do *workflow* para que ao seu término o cientista fosse devidamente informado e pontos de monitoramento de geração de cada arquivo **.mafft*. A Tabela 7 apresenta um resumo com as estatísticas dessas execuções.

Tabela 7 Estatísticas da execução do SciLightning

Aviso	*Total real*	*Total monitorado*	*Revocação*	*Precisão*
Erro	30	28	93,3%	100%
Ponto de monitoramento de fim de atividade	5	5	100%	100%
Ponto de monitoramento de geração de arquivo	400	387	96,8%	100%

Para avaliar o desempenho do SciLightning utilizamos critérios da área de busca e recuperação da informação (Baeza-Yates e Ribeiro-Neto 2011). As medidas mais comuns para avaliar a qualidade nesta área são conhecidas com revocação e precisão (ou do inglês *recall* e *precision*) e as adaptamos para o contexto do experimento. Assim, a revocação mede a proporção de avisos relevantes dados dentro do total de avisos que deveriam ser dados. A precisão mede quantos avisos de fato relevantes foram dados.

Analisando os dados podemos perceber que o SciLightning conseguiu atingir níveis de revocação de mais de 90% em todos os tipos de aviso. Os casos de erro que não foram avisados foram os que a execução do programa não falhou do ponto de vista do sistema operacional, mas na prática não produziu os dados que deveria ter produzido. Este tipo de erro não tem como ser capturado pelo SciLightning já que ele considera como "erro" quando o sistema operacional indica algum erro na invocação ou execução do programa propriamente dito. Em relação à geração dos arquivos, os avisos que não foram dados se referiram a problemas na gerência da fila de mensagens, uma vez que as *cloud activities* que produzem os arquivos **.mafft* são as associadas a atividade de alinhamento e estas são bem curtas, o que fez com que uma grande quantidade de

mensagens se acumulasse na fila rapidamente, gerando assim uma sobrecarga de envio de mensagens e a consequente perda de informações pelo C2DM.

7.12 Análise de Vazão ao Acesso aos Dados e de Impacto de Transferência de Dados

Apesar de não se o foco desta tese a otimização da transferência de dados, devemos analisar seu impacto na execução do *workflow* na nuvem. A fim de compreender a vazão no acesso aos dados e o desempenho de transferência dos mesmos usando o SciCumulus com o *workflow* SciPhy, medimos a taxa de transferência de dados dos componentes do SciCumulus implantado sobre o serviço de armazenamento de dados Amazon S3. Para esta experiência fixamos a quantidade de arquivos multi-fasta de entrada para 200 e o tipo de máquina virtual para *large*.

Cada máquina lê e (principalmente) escreve grande volume de dados de e para o ambiente da Amazon (80 MB e 8,5 GB, respectivamente). A vazão de transferência agregada é apresentada na Figura 45. Neste experimento, para transferir os arquivos de 80 MB comprimidos (arquivo *zip* compactado com compressão máxima), necessitamos de 7 minutos utilizamos uma conexão ADSL comum lenta (2 mbps). Esta latência quando comparada com o tempo de execução total do SciPhy é completamente aceitável. No entanto, esses dados de entrada podem ser colocados *a priori* no ambiente S3, reduzindo o custo geral de transferência.

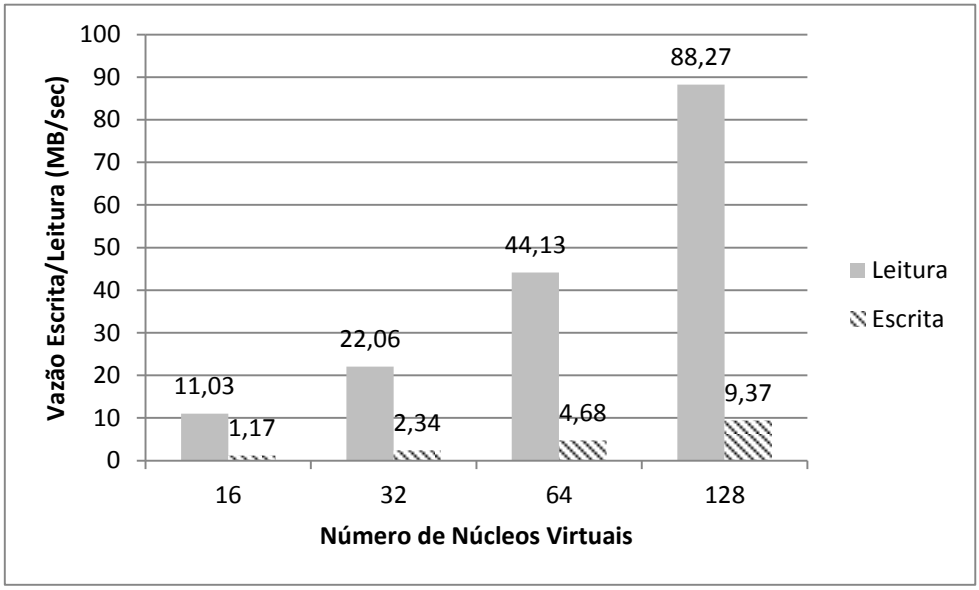


Figura 45 Vazão de escrita e leitura de dados no SciCumulus

Esta avaliação da vazão no acesso aos dados foi realizada de forma simples, pois o intuito era apenas mostrar que o impacto não é grande a ponto de tornar a execução do *workflow* proibitiva. Está fora do escopo desta tese o estudo sobre otimizações na transferência dos dados, apesar de que a incorporação de mecanismos de transferência otimizados como o GridFTP (Kourtellis *et al.* 2008) não está descartada.

7.13 Análise de Consultas de Proveniência no SciCumulus

Esta seção discute os resultados das diversas execuções do SciPhy com o SciCumulus materializados sob a forma de consultas baseadas no repositório de proveniência proposto. As consultas têm como função recuperar os descritores de proveniência que foram previamente coletados pelo SciCumulus ao executar o *workflow* SciPhy na nuvem da Amazon.

As consultas consideradas nesta seção representam apenas uma fração do subconjunto original de consultas previamente definidas pelos pesquisadores que utilizam o *workflow* SciPhy. Todas as consultas foram executadas na base de proveniência instanciada com o PostgreSQL 8.4 na máquina mp4-4.dyndns.info. A base em questão contém 220.548 *cloud activities* geradas e 1.102.740 arquivos produzidos nas mais de 300 execuções do SciPhy. As consultas são realizadas como parte dos testes da cobertura do repositório de proveniência. As consultas foram classificadas de acordo em três grupos distintos que são baseados nos diferentes tipos de proveniência a que estão associadas. As três categorias são:

i. Consultas envolvendo descritores de proveniência prospectiva;
ii. Consultas envolvendo descritores de proveniência retrospectiva;
iii. Consultas envolvendo descritores de proveniência retrospectiva e prospectiva simultaneamente.

A seguir apresentamos uma série de consultas relevantes formuladas em SQL (uma vez que trabalhamos com um banco de dados relacional para representar o repositório de proveniência) para os cientistas e que foram executadas na base de proveniência do SciCumulus. Estas consultas foram elaboradas a partir da necessidade da Dra. Kary Ann del Carmen Soriano Ocaña, especialista em bioinformática que acompanhou a execução dos experimentos apresentados nesta tese.

7.13.1 Consultas de Proveniência Prospectiva

Consulta 1: "Recuperar, por ordem crescente de identificador dos *workflows* as atividades relacionadas a cada *workflow* e as *cloud activities* associadas a cada atividade".

```
SELECT w.tag,
        w.description
        a.tag,
        t.workspace
from hworkflow w, hactivity a, hactivation t
where w.wkfid = a.wkfid
and a.actid = t.actid
order by w.wkfid
```

SciPhy-60003	Phylogeny using RAXML-kalign	raxml	/root/exp/raxml/4/
SciPhy-60003	Phylogeny using RAXML-kalign	raxml	/root/exp/raxml/3/
SciPhy-60003	Phylogeny using RAXML-kalign	raxml	/root/exp/raxml/2/
SciPhy-60003	Phylogeny using RAXML-kalign	raxml	/root/exp/raxml/1/
SciPhy-60003	Phylogeny using RAXML-kalign	readseq	/root/exp/readseq/5/
SciPhy-60003	Phylogeny using RAXML-kalign	readseq	/root/exp/readseq/4/
SciPhy-60003	Phylogeny using RAXML-kalign	readseq	/root/exp/readseq/3/
SciPhy-60003	Phylogeny using RAXML-kalign	readseq	/root/exp/readseq/2/
SciPhy-60003	Phylogeny using RAXML-kalign	readseq	/root/exp/readseq/1/
SciPhy-60003	Phylogeny using RAXML-kalign	mafft	/root/exp/mafft/2/
SciPhy-60003	Phylogeny using RAXML-kalign	mafft	/root/exp/mafft/3/
SciPhy-60003	Phylogeny using RAXML-kalign	mafft	/root/exp/mafft/4/
SciPhy-60003	Phylogeny using RAXML-kalign	mafft	/root/exp/mafft/1/
SciPhy-60003	Phylogeny using RAXML-kalign	mafft	/root/exp/mafft/5/

Figura 46 Resultado da execução da consulta 1

Esta consulta talvez seja a mais básica dentre todas as consultas de proveniência prospectiva, porém é importante, pois descreve toda a estrutura dos *workflows* que foram ou estão sendo executados. Através desta consulta estamos aptos a capturar os dados de um processo caso queiramos gerar um grafo OPM de proveniência.

Consulta 2: "Recuperar todos os *workflows* que contém uma determinada atividade chamada "modelgenerator".

```
select w.wkfid, w.tag, w.exectag from hworkflow w
where exists  (select 1 from hworkflow w2, hactivity a
                where w2.wkfid = w.wkfid
                and w2.wkfid = a.wkfid
                and a.tag = 'modelgenerator')
```

56	MafftAdaptive_3	5	/root/exp/	Phylogeny using RAXML
57	MafftAdaptive_4	5	/root/exp/	Phylogeny using RAXML
58	MafftAdaptive_5	5	/root/exp/	Phylogeny using RAXML
59	MafftAdaptive_6	5	/root/exp/	Phylogeny using RAXML
60	MafftAdaptive_7	5	/root/exp/	Phylogeny using RAXML
61	MafftAdaptive_teste	5	/root/exp/	Phylogeny using RAXML
62	MafftAdaptive_8	5	/root/exp/	Phylogeny using RAXML
63	MafftAdaptive_10	5	/root/exp/	Phylogeny using RAXML
64	MafftAdaptive_16Cores	5	/root/exp/	Phylogeny using RAXML
65	MafftAdaptive_16Cores_2	5	/root/exp/	Phylogeny using RAXML
66	MafftAdaptive_16Cores_3	5	/root/exp/	Phylogeny using RAXML
67	Mafft_Adaptive_16Cores_Ir	5	/root/exp/	Phylogeny using RAXML
68	TestCostModel	5	/root/exp/	Phylogeny using RAXML
69	TesteCostModel	5	/root/exp/	Phylogeny using RAXML
72	CostModel-Confiabilidade	5	/root/exp/	Phylogeny using RAXML
73	CostModel-Custo	5	/root/exp/	Phylogeny using RAXML
74	CostModel-Igual-16cores	5	/root/exp/	Phylogeny using RAXML
70	CostModel-Igual-8cores	5	/root/exp/	Phylogeny using RAXML

Figura 47 Resultado da execução da consulta 2

Esta consulta tem como objetivo a descoberta do conhecimento. Este tipo de consulta auxilia o cientista na descoberta de *workflows* já existentes e executados e que possam ser reaproveitados, ou seja, fomenta o reuso de *workflows* conforme discutido por Mattoso *et al.* (2008, 2009, 2010b).

7.13.2 Consultas sobre Proveniência Retrospectiva

Consulta 3: "Recuperar, por ordem crescente de execuções dos *workflows*, as datas e horas de início e término, *tags* dos *workflows* e suas descrições, bem como o nome de todas as atividades associadas a todas as execuções dos *workflows* que foram executados e que não contenham nenhuma *cloud activity* que executou com erro".

```
SELECT w.tag,
        a.tag,
        t.exitstatus,
        t.processor,
        t.workspace,
        t.status,
        t.endtime,
        t.starttime,
        extract ('epoch' from (t.endtime-t.starttime))||','  as duration
from hworkflow w, hactivity a, hactivation t
where w.wkfid = a.wkfid
and a.actid = t.actid
and not exists (select * from hactivation a2
                where a2.actid = a.actid
                and a2.exitstatus <> 0)
order by w.wkfid
```

MafftAdaptive	mafft5	0	1	/root/exp/mafft5/47/	FINISHED	2011-08-16 14:28:47.54-03	2011-08-16 14:28:42.337-03	5.203,
MafftAdaptive	mafft5	0	1	/root/exp/mafft5/62/	FINISHED	2011-08-16 14:29:27.54-03	2011-08-16 14:29:23.031-03	4.509,
MafftAdaptive	mafft5	0	2	/root/exp/mafft5/77/	FINISHED	2011-08-16 14:30:05.497-03	2011-08-16 14:30:00.82-03	4.677,
MafftAdaptive	mafft5	0	1	/root/exp/mafft5/105/	FINISHED	2011-08-16 14:31:19.211-03	2011-08-16 14:31:13.834-03	5.377,
MafftAdaptive	mafft5	0	2	/root/exp/mafft5/120/	FINISHED	2011-08-16 14:32:02.459-03	2011-08-16 14:31:55.972-03	6.487,
MafftAdaptive	mafft5	0	1	/root/exp/mafft5/136/	FINISHED	2011-08-16 14:32:48.431-03	2011-08-16 14:32:43.829-03	4.602,
MafftAdaptive	mafft5	0	2	/root/exp/mafft5/149/	FINISHED	2011-08-16 14:33:26.907-03	2011-08-16 14:33:21.177-03	5.73,
MafftAdaptive	mafft5	0	2	/root/exp/mafft5/164/	FINISHED	2011-08-16 14:34:10.702-03	2011-08-16 14:34:06.119-03	4.583,
MafftAdaptive	mafft5	0	1	/root/exp/mafft5/186/	FINISHED	2011-08-16 14:35:06.652-03	2011-08-16 14:35:01.58-03	5.072,
MafftAdaptive	mafft5	0	2	/root/exp/mafft5/200/	FINISHED	2011-08-16 14:35:42.759-03	2011-08-16 14:35:38.148-03	4.611,
MafftAdaptive	mafft5	0	1	/root/exp/mafft5/1/	FINISHED	2011-08-16 14:26:21.113-03	2011-08-16 14:26:09.956-03	11.157,
MafftAdaptive	mafft5	0	1	/root/exp/mafft5/3/	FINISHED	2011-08-16 14:26:30.246-03	2011-08-16 14:26:21.317-03	8.929,
MafftAdaptive	mafft5	0	2	/root/exp/mafft5/2/	FINISHED	2011-08-16 14:26:39.512-03	2011-08-16 14:26:09.957-03	29.555,
MafftAdaptive	mafft5	0	1	/root/exp/mafft5/4/	FINISHED	2011-08-16 14:26:40.491-03	2011-08-16 14:26:30.45-03	10.041,

Figura 48 Resultado da execução da consulta 3

Em termos práticos esta consulta é fundamental, pois permite que os pesquisadores monitorem em tempo real as execuções individualizadas de cada *cloud activity* de cada *workflow*. É importante relembrar que a proveniência do SciCumulus é gerada em tempo de execução, ou seja, diferentemente de outras abordagens que somente geram a proveniência ao final da execução do *workflow*, possibilitamos ao cientista fazer uma análise prévia dos dados para avaliar se o experimento está satisfatório ou não.

Consulta 4: "Recuperar os nomes, tamanhos e localização dos arquivos com a extensão .mafft que foram produzidos em todas as execuções do *workflow* SciPhy. Recuperar juntamente qual *workflow* e qual atividade produziu o arquivo".

```
SELECT w.tag,
        a.tag,
        f.fname,
        f.fsize,
        f.fdir
from hworkflow w, hactivity a, hactivation t, hfile f
where w.wkfid = a.wkfid
and a.actid = t.actid
and f.taskid = t.taskid
and w.tag like '%SciPhy%'
and f.fname like '%mafft'
```

SciPhy-60003	readseq	supermatrixCombined3.mafl	641661	/root/exp/readseq/3/
SciPhy-60003	readseq	supermatrixCombined2.mafl	641661	/root/exp/readseq/2/
SciPhy-60003	readseq	supermatrixCombined4.mafl	641661	/root/exp/readseq/4/
SciHmm-SciPhy-30009	mafft	E_Poba_aa.fastaC01.077_f	26700	/root/exp/mafft/5/
SciHmm-SciPhy-30009	mafft	E_Plkn_aa.fastaC01.077_fa	2448	/root/exp/mafft/3/
SciHmm-SciPhy-30009	mafft	E_Phtr_aa.fastaC01.077_fa	3150	/root/exp/mafft/1/
SciHmm-SciPhy-30009	mafft	E_Plfa_aa.fastaC01.077_fa	2955	/root/exp/mafft/2/
SciHmm-SciPhy-30009	mafft	E_Plyo_aa.fastaC01.077_fa	1752	/root/exp/mafft/4/
SciHmm-SciPhy-30009	readseq	E_Plyo_aa.fastaC01.077_fa	1752	/root/exp/readseq/4/
SciHmm-SciPhy-30009	readseq	E_Plfa_aa.fastaC01.077_fa	2955	/root/exp/readseq/2/
SciHmm-SciPhy-30009	readseq	E_Poba_aa.fastaC01.077_f	26700	/root/exp/readseq/5/
SciHmm-SciPhy-30009	readseq	E_Plkn_aa.fastaC01.077_fa	2448	/root/exp/readseq/3/
SciHmm-SciPhy-30009	readseq	E_Phtr_aa.fastaC01.077_fa	3150	/root/exp/readseq/1/
SciPhy-60004	mafft	supermatrixCombined1.mafl	641661	/root/exp/mafft/1/
SciPhy-60004	mafft	supermatrixCombined5.mafl	641661	/root/exp/mafft/5/
SciPhy-60004	mafft	supermatrixCombined4.mafl	641661	/root/exp/mafft/4/

Figura 49 Resultado da execução da consulta 4

Similarmente a Consulta 3, esta consulta é fundamental para o monitoramento em tempo real, pois permite ao cientista descobrir os arquivos que já tenham sido gerados, analisa-los e tomar uma decisão como por exemplo, interromper a execução do *workflow*. Este tipo de consulta abre caminhos para que possa ser desenvolvido no SciCumulus mecanismos de *steering* de *workflows* como os propostos por Dias *et al.* (2011).

7.13.3 Consultas de Proveniência Prospectiva e Retrospectiva

Consulta 5: "Selecionar todas as imagens de máquinas virtuais, suas máquinas virtuais associadas, envolvidos na execução das atividades do *workflow* de identificador 119".

```
SELECT w.tag,
        a.tag,
        t.workspace,
        i.tag,
        av.publicdns
from hworkflow w, hactivity a, hactivation t, himage i, havailablevm av
where w.wkfid = a.wkfid
and a.actid = t.actid
and w.amid = av.amid
and av.amiid = i.amiid
and w.wkfid =119
order by a.actid
```

SciPhy-60003	mafft	/root/exp/mafft/1/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	mafft	/root/exp/mafft/4/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	mafft	/root/exp/mafft/3/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	mafft	/root/exp/mafft/2/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	readseq	/root/exp/readseq/5/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	readseq	/root/exp/readseq/1/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	readseq	/root/exp/readseq/3/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	readseq	/root/exp/readseq/2/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	readseq	/root/exp/readseq/4/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	raxml	/root/exp/raxml/3/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	raxml	/root/exp/raxml/2/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	raxml	/root/exp/raxml/4/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	raxml	/root/exp/raxml/1/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com
SciPhy-60003	raxml	/root/exp/raxml/5/	i-f2676d92	ec2-50-16-13-168.compute-1.amazonaws.com

Figura 50 Resultado da execução da consulta 5

Finalmente a Consulta 5 cataloga todos os artefatos do provedor de nuvem que foram utilizados na execução de um determinado *workflow*. Este tipo de consulta também apoia a questão da reprodutibilidade, pois na imagem em questão já temos o mesmo ambiente configurado, facilitando que os mesmos resultados possam ser alcançados por diferentes pesquisadores.

7.14 Considerações Finais sobre Análise Experimental do SciCumulus

Os experimentos realizados nesta tese foram divididos em três grupos de avaliações: análise de desempenho, análise financeira e análise dos dados de proveniência. O primeiro grupo apresenta uma avaliação do modelo de custo proposto, do escalonamento guloso e das estratégias de escalonamento adaptativas. Estes experimentos mostram que o SciCumulus, mesmo sem dimensionar o tamanho do *cluster* virtual dinamicamente tem um desempenho satisfatório frente a outras abordagens consideradas padrões *de facto* como o Hadoop. A segunda avaliação apresenta uma análise monetária sobre a execução do *workflow*, mostrando que em determinados cenários (dependendo do modelo de cobrança) o SciCumulus consegue oferecer uma economia na execução paralela do *workflow*.

A terceira avaliação analisa a cobertura do repositório de proveniência do SciCumulus. Esta avaliação apresenta um conjunto de consultas sobre descritores de proveniência níveis distintos. Os resultados evidenciam a funcionalidade do repositório de descritores de proveniência coletados pelo SciCumulus. As consultas evidenciam que é viável acessar descritores de proveniência de experimentos executados em complexos ambientes distribuídos como nuvens por intermédio de consultas baseadas em um modelo relacional.

Todos os experimentos permitiram avaliar que o SciCumulus viabilizou a execução paralela de um *workflow* científico em um ambiente de nuvem de modo transparente e foi capaz de apresentar melhorias de desempenho de até 37,9% quando comparado as abordagens tradicionais. Estes resultados foram alcançados com a vantagem adicional de se ter proveniência prospectiva e retrospectiva gerada em tempo real possibilitando consultas à base de dados de proveniência durante a execução dos experimentos, o que permite a geração de gráficos como o do percentual de ociosidade apresentado na Figura 43.

Além do *workflow* de análise filogenética SciPhy, durante o período do doutorado, outros experimentos foram executados com o SciCumulus de forma a ampliar a gama de tipos de experimento durante a análise. Os outros experimentos incluem análises filogenômicas (Oliveira et al. 2012), cristalografia de raios-x (Oliveira et al. 2011a), genômica comparativa (Ocaña et al. 2011a) e análise de relações

evolucionárias (Ocaña et al. 2012). Baseado na experiência do SciPhy e dos outros *workflows*, pudemos concluir que *workflows* que possuam atividades que possam ser paralelizadas como *cloud activities* embaraçosamente paralelas tendem a se beneficiar do paralelismo proporcionado pelo SciCumulus, uma vez que executam independentemente umas das outras o que facilita a distribuição e diminui a sobrecarga de comunicação. Este benefício pode ser ampliado caso as *cloud activities* possuam tempos de processamento muito díspares e uma taxa alta de compartilhamento de dados, fazendo com que o modelo de custo proposto possa distribuir as *cloud activities* de maneira mais eficiente.

Capítulo 8 - Trabalhos Relacionados

Existem algumas abordagens para a execução paralela de *workflows* científicos em nuvens de computadores. Entretanto, nenhuma destas abordagens cobre todas as necessidades explicitadas nesta tese, como por exemplo, a adaptabilidade na execução devido a mudanças no ambiente, a distribuição das atividades levando em conta o desempenho, confiabilidade e custo monetário e a captura de proveniência distribuída. Para avaliar as diferentes abordagens relacionadas à gerência da execução paralela de *workflows* científicos em nuvens, foi realizada uma busca sistemática, datada de 18 de janeiro de 2012, nas bases de dados do Scopus e IEEE Xplore, utilizando a seguinte *string* de busca: "*workflow*" AND ("*cloud*" OR "*cloud computing*") AND ("*parallel execution*" OR "*distributed execution*").

As buscas retornaram 276 artigos, dos quais 58 foram selecionados para leitura completa por apresentarem abordagens de execução paralela de *workflows* (ou pelo menos de programas) em nuvens. Adicionalmente a esta busca sistemática na área de *workflows* científicos foi feita uma pesquisa *ad hoc* visando complementar as bases pesquisadas e incluir a literatura nacional, que muitas vezes não é indexada nestas bases.

A distribuição adaptativa de tarefas para execução paralela na nuvem baseada em um modelo de custo, tal como apresentado nesta tese foi inspirada em estabelecidas técnicas de processamento adaptativo de consultas em bancos de dados (Furtado *et al.* 2008, Lima *et al.* 2010). No processamento de consultas adaptativo, o foco é utilizar o *feedback* dos sistemas de banco de dados em tempo de execução para modificar o processamento de consultas de um modo que forneça um melhor tempo de resposta ou se alcance a utilização mais eficiente da CPU. No contexto desta tese, a consulta é equivalente à execução de uma atividade na nuvem que consome dados e uma combinação de valores de parâmetros.

Baseados nos artigos que foram analisados, estruturamos este capítulo em três seções principais. Na Seção 8.1 discutimos os algoritmos existentes para escalonamento de tarefas em nuvens e grades que se baseiam em modelos de custo multiobjetivo, como o apresentado nesta tese. A Seção 8.2 apresenta os arcabouços existentes para execução

distribuída de *workflows* científicos em nuvens computacionais. E, finalmente, a Seção 8.3 discute sobre abordagens de captura de proveniência distribuída em nuvens.

8.1 Algoritmos de Escalonamento de Tarefas em Nuvens Computacionais

Na literatura, existem diversos trabalhos que se concentram em desenvolver algoritmos de escalonamento multiobjetivo para ambientes distribuídos, sendo a maioria deles algoritmos de escalonamento biobjetivo. Nesta seção apresentamos alguns algoritmos de escalonamento que são utilizados em ambientes distribuídos, sejam eles de nuvem ou não e que de alguma forma tem relação com a abordagem de distribuição apresentada nesta tese. Esta seção foi separada em duas para melhor organização. A Seção 8.1.1 descreve algoritmos de escalonamento não-adaptativos enquanto que a Seção 8.1.2 foca nos algoritmos que se adaptam durante a execução.

8.1.1 Algoritmos Não-Adaptativos

Uma abordagem não adaptativa clássica é a apresentada por Boeres *et al.* (2011) que introduz um algoritmo baseado em uma função de custo chamada MRCD (do inglês *Makespan Reliability Cost Driven*). O algoritmo integra objetivos de confiabilidade e desempenho simultaneamente na mesma fórmula. Este trabalho foi a inspiração para a abordagem proposta nesta tese, principalmente por tratar a questão do modelo de custos de uma forma ponderada para permitir uma sintonia fina dos critérios. No entanto, Boeres *et al.* não levam em conta o custo monetário envolvido na execução e a questão da variação da quantidade de recursos, uma vez que o algoritmo foi projetado para executar em ambientes "estáticos", tais como *clusters*.

Byun *et al.* (2011) apresentam um algoritmo de aproximação chamado BTS (do inglês *Balanced Time Scheduling*), que estima o número mínimo de máquinas virtuais necessárias para executar atividades de um *workflow* em paralelo dentro de um tempo especificado pelo usuário para seu término (*i.e. deadline*). A ideia principal do artigo é dimensionar a quantidade de máquinas virtuais *a priori* para atender a demanda, ou seja, o escalonamento é realizado antes da execução do *workflow*. Esta abordagem apresenta diversos resultados experimentais, com base em uma série de *workflows* sintéticos e reais.

Assayad *et al.* (2004) introduzem uma heurística de escalonamento biobjetivo para a aplicações modeladas como um DAG (que incluem *workflows* científicos) de acordo com dois critérios: primeiro, de minimização do tempo de execução do DAG, e o segundo de maximização da confiabilidade do sistema. Uma vantagem apresentada por esta abordagem é que ele utiliza o conceito de replicação ativa de atividades para melhorar a confiabilidade total da execução. Entretanto, uma vez que esta abordagem é focada em *clusters* de computadores, não leva em conta o custo monetário a ser minimizado, tornando-se assim não indicada para ambientes de nuvens de computadores.

Kloh *et al.* (2010, 2011) apresentam um modelo de escalonamento de *workflows* baseado em um algoritmo de escalonamento híbrido de biobjetivo (tempo total de execução e confiabilidade). O modelo proposto tem como objetivo otimizar os critérios escolhidos pelos usuários, com base em sua ordem de prioridade especificada e um limite de variação especificados pelos mesmos. Todo o plano de escalonamento é realizado antes da execução do *workflow*, ou seja, não há redimensionamento do ambiente e nem a distribuição de acordo com o tipo de recurso.

Outra classe de abordagens para escalonamento em grades e nuvens inclui diferentes tipos de modelos econômicos, que são especialmente populares em nuvens por causa da ênfase no modelo pague-pelo-uso. A tomada de decisões sobre o escalonamento nestes modelos é normalmente direcionada por mecanismos de mercado, tais como leilões. No entanto, estas abordagens são frequentemente criticadas por seu baixo desempenho devido à sobrecarga que impõem que é frequentemente causada quando as negociações falham. Chard et. al. (2010) propõem estratégias de escalonamento que visam reduzir falhas de alocação e com isso aumentar a ocupação e o desempenho usando técnicas como o sobre-reserva (do inglês *overbooking*). Chang *et al.* (2010) e Zhao e Jamali (2011) abordam o problema de baixo desempenho em escalonamento baseado em modelos econômicos sob uma perspectiva diferente: em vez de aumentar a ocupação, ou seja, usando os muitos recursos disponíveis da forma mais eficiente possível, tentam usar poucos recursos para reduzir a sobrecarga de avaliação de custos no momento de despachar as execuções dos programas nos recursos.

8.1.2 Algoritmos Adaptativos

Qin *et al.* (2005) apresentam uma heurística de programação dinâmica paralela em tempo real, para escalonamento de tarefas em *clusters* heterogêneos. Supõe-se que os as tarefas são modeladas como DAGs e que chegam ao sistema seguindo um processo Poisson. O algoritmo leva em consideração a confiabilidade e o tempo de execução como metas a serem minimizadas. Ainda, analisa se o prazo informado pode ou não ser garantido. A adaptabilidade da abordagem se dá na distribuição das tarefas de acordo com a capacidade da máquina virtual que irá executá-la. Embora essa abordagem concentre-se no cumprimento de prazos, não considera o custo monetário ou o orçamento limite informados pelos cientistas.

Lin e Lu (2011) introduzem o algoritmo de escalonamento adaptativo SHEFT para *workflows* executados em ambientes de computação em nuvem. O algoritmo SHEFT é uma versão adaptativa do algoritmo de escalonamento conhecido como HEFT, já aplicado em grades por Mayer *et al.* (2007), que é um dos representante da classe de algoritmos de escalonamento de *workflows* que otimizam o tempo de execução do mesmo (Topcuoglu *et al.* 2002). A diferença do SHEFT é que o mesmo permite que os recursos sejam ampliados (ou reduzidos) elasticamente durante a execução do *workflow*.

O trabalho de Lu e Gu (2011) propõe o escalonamento dinâmico e adaptativo de recursos para a execução de aplicações modeladas como DAG em nuvens de acordo com a mudança de carga requisitada. Para atingir tal objetivo, é utilizado o algoritmo de colônia de formigas (Dorigo e Stützle 2004). A ideia deste algoritmo é simular o caminho de formigas na natureza e os autores o utilizam para explorar o espaço de busca de recursos como se fosse um caminho. O foco principal deste algoritmo de escalonamento é na redução do makespan e do tráfego de rede, uma vez que os autores ressaltam que estes são os gargalos a serem tratados.

Hsu e Chen (2010) apresentam em seu artigo duas técnicas adaptativas de escalonamento de atividades para otimização de tempo de execução geral e minimização de custos monetários do uso de recursos. Nesta abordagem, o usuário tem que optar por escalonar otimizando somente um dos dois critérios existentes. A adaptabilidade neste caso se dá apenas na quantidade de recursos (máquinas virtuais)

que pode ser aumentada ou diminuída a critério do usuário. Todo o escalonamento é realizado *a priori* e quando ocorre uma mudança no ambiente o mesmo é refeito.

Mao *et al.* (2011) apresentam um processo de otimização do escalonador utilizado em implementações do modelo MapReduce em nível de tarefa. Esta abordagem foca na otimização da carga de trabalho e no ajustamento da distribuição de tarefas de acordo com a configuração de *hardware* existente. Essa distribuição é realizada em tempo real nos nós em um *cluster* Hadoop. Esta abordagem foca unicamente na redução do tempo de execução e no aumento de utilização de recursos de *hardware*. Toda a implementação foi baseada em um controlador dinâmico dos slots (DSC), que ajusta a distribuição dos Maps e Reduces adaptativamente.

Smanchat *et al.* (2009) propõem um algoritmo de escalonamento adaptativo para grades e nuvens que visa apoiar múltiplas instâncias de *workflows* e evitar a alocação de tarefas com alta concorrência de recursos para minimizar o atraso devido ao bloqueio das tarefas. A ideia é verificar qual o percentual de utilização de recursos e escalonar tarefas de *workflows* para recursos menos utilizados. Esta abordagem foi implementada no Nimrod/K (Abramson *et al.* 2008) e acoplado ao Kepler. Entretanto, neste artigo, apenas uma avaliação por simulação foi realizada.

Similarmente a abordagem de Smanchat *et al.* (2009), Nie e Xu (2009) propõem uma abordagem de escalonamento adaptativo para grades elásticas. O conceito de grades elásticas pode ser descrito como o oferecimento sob demanda na internet de recursos de grades tradicionais que estejam sendo subutilizados. Nesta abordagem, Nie e Xu podem aumentar ou diminuir a quantidade de recursos utilizados na execução de um *workflow*, porém o foco aqui é somente na redução do makespan. Neste artigo, toda a avaliação experimental foi realizada com a utilização de dados sintéticos e simulações.

Existe uma categoria de algoritmos de escalonamento adaptativos que consideram a questão da otimização do consumo de energia juntamente com a redução do makespan. Muitas destas propostas utilizam métodos de aprendizado de máquina, a fim de tirar proveito das informações do sistema. Berral et al (2010, 2011) e Goiri *et al.* (2010), propõem algoritmos de escalonamento adaptativos baseados e um dimensionador de recursos que liga e desliga as máquinas virtuais de acordo com o nível de consumo de energia. Os algoritmos de aprendizado de máquina são utilizados

para reconhecimento do consumo de energia e para lidar com informações incertas, maximizando assim o desempenho.

8.1.3 Considerações Sobre os Algoritmos de Escalonamento

Existe um grande número de trabalhos observados que, de alguma forma, procuram otimizar o escalonamento de *workflows* científico (ou ao menos aplicações modeladas como DAG) em paralelo em nuvens computacionais. Estes trabalhos podem ser classificados em não-adaptativos e adaptativos. Na categoria não-adaptativos sobressaem as propostas de Boeres *et al.* (2011), Byun *et al.* (2011) e Kloh *et al.* (2010, 2011) por focarem no escalonamento de *workflows* científicos propriamente ditos. Entretanto, nenhum destes trabalhos trata todos os requisitos que consideramos fundamentais em nuvens que são: a redução do tempo de execução, o aumento na confiabilidade e a redução do custo financeiro.

Entretanto, apenas Boeres *et al.* utilizam o modelo de custo ponderado que consideramos importante para o cientista executar uma sintonia fina no escalonamento. Em abordagens como a de Kloh *et al.* o cientista configura o critério principal e o secundário e a otimização é automática, ou seja, uma vez definidos os critérios, o cientista não pode mais tomar nenhuma ação. Ainda nos algoritmos não-adaptativos, as abordagens propostas por Chard et. al. (2010), Chang *et al.* (2010) e Zhao e Jamali (2011) focam em otimizar o consumo de energia com o escalonamento. Apesar de ser uma estratégia promissora e interessante como tema de pesquisa, foge ao escopo desta tese.

Na categoria de algoritmos adaptativos, apenas a abordagem de Hsu e Chen (2010) foca em reduzir o tempo de execução e o custo monetário. Todas as outras abordagens focam em reduzir somente o tempo de execução. A questão da adaptabilidade se dá apenas no aumento e diminuição de custos. Nenhuma das abordagens foca no agrupamento de tarefas ou em distribuir as tarefas levando-se em conta a confiabilidade do ambientem, fato este que é fundamental em nuvens. Além disso, boa parte das abordagens adaptativas propostas foram avaliadas apenas em ambientes de simulação como o *Cloud*Sim. A Tabela 8 apresenta um resumo comparativo dos algoritmos de escalonamento apresentados neste capítulo frente à abordagem proposta nesta tese.

Tabela 8 Quadro comparativo dos algoritmos de escalonamento

Algoritmo	*Adaptativo*	*Método*	*Parâmetros*	*Ambiente*	*Ferramenta*
MRCD (Boeres *et al.* 2011)	Não	Análise de dependência	Makespan, Confiabilidade	Clusters e grades	MPI
BTS (Byun *et al.* 2011)	Não	Análise de dependência	Makespan	Nuvens	Pegasus
Assayad *et al.* (2004)	Não	Múltiplas instâncias	Makespan, Confiabilidade	Clusters	MPI
Kloh (2010, 2011)	Não	Múltiplas instâncias	MakeSpan, Confiabilidade	Nuvens	MPI
Chard et. al. (2010)	Não	Múltiplas instâncias	Desempenho, Confiabilidade, Ocupação	Grades e nuvens	Aneka
Zhao e Jamali (2011)	Não	Múltiplas instâncias	Desempenho, Confiabilidade, Ocupação	Grades e nuvens	Aneka
Chang *et al.* (2010) e	Não	Múltiplas instâncias	Desempenho, Confiabilidade, Ocupação	Grades e nuvens	Aneka
Qin *et al.* (2005)	Sim	Múltiplas instâncias	Makespan	Grades e nuvens	-
SHEFT	Sim	Análise de dependência	Makespan	Nuvens	*Cloud*Sim
Lu e Gu (2011)	Sim	Múltiplas instâncias	Makespan, Tráfego de rede	Nuvens	*Cloud*Sim
Hsu e Chen (2010)	Sim	Múltiplas instâncias	Makespan, Custo financeiro	Nuvens	*Cloud*Sim
Mao *et al.* (2011)	Sim	Múltiplas instâncias	Makespan, Alocação de recursos	Grades e nuvens	*Cloud*Sim
Smanchat *et al.* (2009)	Sim	Análise de dependência	Makespan	Grades	GridSim
Nie e Xu (2009)	Sim	Múltiplas instâncias	Makespan, Consumo de energia	Grades	*Cloud*Sim
Goiri *et al.* (2010)	Sim	Múltiplas instâncias	Makespan, Consumo de energia	Nuvens	*Cloud*Sim
Oliveira *et al.* (2011c)	Sim	Análise de Dependências	Makespan, Confiabilidade, Custo	Nuvens	SciCumulus

8.2 Abordagens Existentes para Execução Paralela de *Workflows* Científicos em Nuvens

Existem diversas abordagens propostas que tem como objetivo fornecer um arcabouço para execução de *workflows* em ambientes de PAD, e mais precisamente em ambientes de nuvens de computadores. Nesta seção apresentamos algumas destas abordagens, que inclusive implementam alguns dos algoritmos apresentados na seção anterior. Esta seção foi separada em três para sua melhor organização. A Seção 8.2.1 introduz as abordagens baseadas no modelo MapReduce. A Seção 8.2.2 apresenta as abordagens

baseadas em varredura de parâmetros enquanto que a Seção 8.2.3 descreve abordagens para execução de *workflows* que se adaptam durante a execução.

8.2.1 Abordagens Baseadas no Modelo MapReduce

O MapReduce recentemente ganhou muita atenção como um modelo de programação paralela para análises científicas que demandam PAD. Existe uma gama de aplicações que podem ser paralelizadas utilizando este modelo. Em geral as aplicações embaraçosamente paralelas (Foster 1995) podem ser beneficiar muito desse modelo, como por exemplo o alinhamento de sequencias no domínio da bioinformática (Matsunaga *et al.* 2008). Entretanto, as implementações do modelo MapReduce como o Hadoop não focam em paralelizar atividades de um *workflow* científico. Desta forma, esses arcabouços devem ser acoplados aos SGWfC existentes.

Existem três SGWfC que focaram em acoplar suas máquinas de execução com o Hadoop. A primeira delas foi o Kepler, gerando assim a abordagem Kepler+Hadoop (Wang *et al.* 2009). Seguindo a mesma ideia, o VisTrails acoplou sua máquina ao Hadoop (Howe *et al.* 2009) para paralelizar visualizações paralelas. Além do Kepler e do VisTrails, o SGWfC View (Cui Lin *et al.* 2009) implementa componentes para funções de Map e Reduce em sua arquitetura (Fei *et al.* 2009). Apesar de ser apresentado como contribuição nos artigos, o uso de nuvem não fica claro nas propostas. Em todas as abordagens os componentes para conexão com a nuvem não são nativos dos SGWfC e tampouco são detalhados. E no caso do Kepler, o teste foi realizado em um *cluster* e utilizando como estudo de caso um *workflow* de contagem de palavras, que não necessariamente traduz os cenários encontrados na pesquisa científica.

Além disso, alguns tipos de *workflows*, no entanto, não são interessantes para serem modelados utilizando o modelo MapReduce. Como as implementações do MapReduce fazem uma distribuição estática, esta distribuição pode retardar o tempo de execução de alguns programas científicos (Vega *et al.* 2010). Além disto, as implementações de MapReduce retardam o disparo de tarefas, pois consomem um período de tempo para gerar as tarefas do tipo Map antes de iniciar a execução. Nas tarefas de curta duração (menos de um minuto), esta situação gera uma sobrecarga significativa na duração do *workflow*.

8.2.2 Abordagens Baseadas em Varredura de Parâmetros

Algumas abordagens que executam *workflows* científicos em paralelo em nuvens se concentram exclusivamente no apoio à varredura de parâmetros. Essa varredura é muito comum em *workflows*. Uma das propostas mais proeminentes é o Nimrod/K (Abramson *et al.* 2009a, 2009b, 2011). O Nimrod/K é um *plug-in* que pode ser acoplado ao Kepler para apoiar a varredura de parâmetros e a distribuição das tarefas de um *workflow* em ambientes de PAD como *clusters* e nuvens. A distribuição é realizada escalonando cada combinação de valores de parâmetros para os nós do ambiente de PAD. Esta abordagem não oferece captura de proveniência da execução distribuída, que fica a cargo do SGWfC.

O Swift (Zhao *et al.* 2007) é um SGWfC que foi desenvolvido para apoiar o paralelismo de *workflows* em ambiente de PAD de qualquer natureza. O Swift é baseado na máquina de *workflow* chamada Karajan (Laszewski *et al.* 2007), que permite escalonamento paralelo e balanceamento de carga em tempo de execução. O Swift pode paralelizar problemas de varredura de parâmetros alocando cada tarefa do *workflow* (cada combinação de valores a ser explorada) em diferentes nós computacionais (no caso da nuvem, diversas máquinas virtuais). O paralelismo é explicitamente representado através de uma linguagem própria de *script* (chamada de *SwiftScript*). A linguagem possui comandos como o *ForEach* que explicitamente representa as primitivas de paralelismo. Além disso, o apoio à proveniência durante a execução é feito exclusivamente por arquivos de *log* que podem ser devidamente exportados após a execução do *workflow* para uma base relacional. Esta exportação dos dados *a posteriori* dificulta o monitoramento da execução do *workflow*.

Similarmente ao Swift, o Pegasus (Deelman *et al.* 2007), tem o foco em executar *workflows* em paralelo em ambiente de PAD, principalmente em *clusters* e grades, e mais recentemente em nuvens computacionais. O Pegasus tem como objetivo distribuir as atividades de acordo com o poder computacional dos nós e a demanda de cada atividade. O Pegasus pode ser instanciado em um ambiente de nuvem onde exista um *cluster* de máquinas virtuais previamente instanciado. O Pegasus é dependente do Condor (Couvares *et al.* 2007) para distribuir as tarefas. Porém diferentemente do SciCumulus, o Pegasus não configura o ambiente e nem dimensiona o tamanho do *cluster* sob demanda. Além disso, toda otimização realizada pelo Pegasus é focada em melhorar o tempo de execução, ignorando a confiabilidade e o custo monetário.

Similarmente ao Swift, o Pegasus também disponibiliza seus dados de proveniência após a execução do *workflow* e somente os dados relativos à execução das atividades, sem conter descritores sobre os ambientes distribuídos utilizados.

A abordagem proposta por Franz *et al.* (2011) introduz um motor que permite a execução de *workflows* em plataformas de nuvem existentes. O motor fornece automaticamente mecanismos para calcular o custo de processamento e monetário de cada tarefa de forma a distribui-las para as melhores máquinas virtuais. Além disso, o motor possui mecanismos de transferência de dados de entrada/saída de/para diferentes plataformas. Entretanto, todos os cálculos e escalonamentos são realizados *a priori*, ou seja, antes do início da execução do *workflow*. Além disso, o motor não se preocupa em capturar a proveniência em nenhum dos níveis (prospectiva ou retrospectiva).

8.2.3 Abordagens Adaptativas

Outras abordagens focam na execução de tarefas em paralelo adaptativamente, se aproveitando dos benefícios das nuvens como a elasticidade de recursos. Um dos exemplos dessas abordagens é o Nephele (Warneke e Kao 2009). O Nephele é uma abordagem para processamento de dados distribuídos que explicitamente explora a alocação dinâmica de recursos oferecidos pelas nuvens, ou seja, o algoritmo de distribuição de tarefas do Nephele considera o aumento e diminuição de recursos durante a execução de tarefas em paralelo similarmente aos algoritmos propostos por Smanchat *et al.* (2009) e Nie e Xu (2009). O Nephele escalona suas tarefas para diferentes tipos de recursos na nuvem e gerencia o curso de execução do *job*. No entanto, o Nephele é desconectado do conceito de *workflows* científicos, a distribuição de tarefas realizada por ele considera apenas o aumento e diminuição na quantidade de recursos, sem considerar a questão de distribuir tarefas mais pesadas para máquinas virtuais mais potentes e vice-versa. Além disso, não oferece mecanismos para explorar a variação dos tipos de recursos e nem para captura de descritores das execuções (proveniência).

O Nuage, proposto por Lee *et al.* (2011) é uma das abordagens para paralelismo de tarefas que usa a teoria dos jogos evolucionários para fornecer uma implementação de aplicações adaptáveis e estáveis em ambientes de nuvem computacional. Diferentemente do SciCumulus, o Nuage está focado na implantação de programas que executam tarefas sequenciais, *i.e.* sem explorar a questão do paralelismo e tenta adaptar

o volume de solicitações de cada programa de acordo com as máquinas virtuais disponíveis para uso. A abordagem proposta pelo Nuage se parece muito com o dimensionamento de recursos oferecido pelo projeto Reservoir (Rochwerger *et al.* 2009) para atender as demandas se sistemas *Web* implantados na nuvem. No entanto, o Nuage é desconectado do conceito de *workflow* cientifico, além de não oferecer apoio algum a questão da captura de dados de proveniência.

A proposta apresentada por Lin *et al.* (Lin *et al.* 2010), chamada de SCPOR tem como objetivo escalar a quantidade de recursos utilizados de acordo com a demanda por parte das tarefas do *workflow*. Entretanto, esta abordagem não está focada na captura de proveniência distribuída e nem no escalonamento das tarefas de modo a minimizar tempo o custo monetário ou aumentar a confiabilidade, mantendo seu foco apenas em reduzir o tempo de execução do *workflow*.

8.2.4 Considerações Sobre as Abordagens para Execução Paralela de *Workflows* em Nuvens Computacionais

Até o momento existem poucos SGWfC ou arcabouços que representam uma solução para a gerência da execução de *workflows* em ambientes distribuídos do tipo nuvens de computadores. A maioria dos sistemas trata a execução em nuvens como se fosse uma execução em um ambiente de *cluster* ou supercomputador, onde os recursos não escalam, *i.e.* onde não podem se beneficiar da característica de elasticidade. As abordagens que oferecem questões adaptativas ainda o fazem apenas dimensionando a quantidade de recursos, mas pecam muitas vezes na captura de proveniência distribuída e na possibilidade de otimizacao do escalonamento para se reduzir tempo de execução ou custos monetários envolvidos na execução do *workflow*. A Tabela 9 apresenta um resumo comparativo das abordagens estudadas neste capítulo.

Tabela 9 Quadro comparativo entre as principais abordagens de execução de *workflows* em paralelo em nuvens de computadores

Abordagem	***Considera elasticidade de recursos***	***Parâmetros de Escalonamento***	***Captura de Proveniência na nuvem***	***Adaptativo***
Kepler+Hadoop	Não	Makespan	Não	Não
VisTrails+Hadoop	Não	Makespan	Não	Não
View+Hadoop	Não	Makespan	Não	Não
Nimrod/K	Não	Makespan	Não	Não
Swift	Não	Makespan,	Sim, mas *a*	Não

		Utilização de recursos	*posteriori*	
Pegasus	Não	Makespan, Utilização de recursos	Sim, mas *a posteriori*	Não
Nephele	Sim	Makespan	Não	Sim
Nuage	Sim	Makespan	Não	Sim
SCPOR	Sim	Makespan	Não	Sim

8.3 Abordagens Existentes para Captura de Proveniência em Nuvens Computacionais

Como a captura de proveniência no SciCumulus é um dos diferenciais da abordagem, foi considerada importante a comparação do mecanismo de captura de proveniência do SciCumulus frente as abordagens no estado da arte. Para avaliar as diferentes abordagens relacionadas à captura de proveniência de *workflows* científicos em nuvens, foi realizada uma busca sistemática, datada de 22 de janeiro de 2012, nas bases de dados do Scopus, ACM Digital Library e IEEE Xplore, utilizando a seguinte *string* de busca: "*workflow*" AND ("*cloud*" OR "*cloud computing*") AND ("*provenance*" OR "*lineage*").

As buscas retornaram 32 artigos, dos quais 15 foram selecionados para leitura completa por apresentarem abordagens de captura de proveniência de *workflows* em nuvens. Adicionalmente a esta busca sistemática na área de *workflows* científicos foi feita uma pesquisa *ad hoc* visando complementar as bases pesquisadas e incluir a literatura nacional, que muitas vezes não é indexada nestas bases. As abordagens para captura de proveniência, distribuída ou não, seguem duas direções distintas. Em geral, existem dois tipos de mecanismos para a coleta destes descritores: os dependentes de SGWfC e os independentes de SGWfC. Essa seção é separada nestas duas categorias.

8.3.1 Abordagens Acopladas a SGWfC

A maioria dos SGWfC que oferecem recursos de proveniência possuem mecanismos diretamente codificadas e dependentes do motor de execução do SGWfC para capturar os descritores de proveniência durante a composição e execução do *workflow*. O VisTrails (Callahan *et al.* 2006), por exemplo, possui mecanismos de proveniência prospectiva e retrospectiva e são armazenados no próprio arquivo do VisTrails (.vt) que é um arquivo compactado com a estrutura do *workflow* em XML e o log de

proveniência. Este arquivo de log do VisTrails é consultado através da interface e utilizando uma linguagem própria (VTQL). Uma das vantagens oferecidas pelo VisTrails é o controle de versões oferecido de forma nativa. Por outro lado o VisTrails não trata a proveniência distribuída uma vez que este não é o foco do SGWfC.

Similarmente ao VisTrails, o Taverna (Hull *et al.* 2006a) e o PASOA (Groth *et al.* 2004) também possuem recursos para captura de proveniência da execução, sendo que o primeiro com a vantagem de oferecer proveniência associada a questão ontológica, para prover consultas com mais semântica. O Taverna grava os dados de proveniência em formato RDF, possibilitando assim consultas de mais alto nível para serem realizadas em SPARQL (Segaran *et al.* 2009). Como o PASOA armazena os dados de proveniência no formato XML, as consultas de proveniência são realizadas utilizando XQuery (Chamberlin 2003). O Kepler oferece um componente (chamado de diretor) especializado na captura de proveniência (Altintas *et al.* 2006, Bowers *et al.* 2008). Apesar de oferecer mecanismos para execução distribuída em grades, o Kepler só captura a proveniência da máquina centralizada que iniciou a execução, não levando em consideração a questão da proveniência obtida no ambiente distribuído. Apesar de armazenar os dados de proveniência em formato relacional, as consultas de proveniência são realizadas utilizando ProLog (Wikipedia 2011).

Existem ainda os SGWfC que focam em distribuição e por isso oferecem mecanismos de captura de proveniência distribuída, como o Swift (Zhao *et al.* 2007) e o Pegasus (Deelman *et al.* 2007). Em especial o Pegasus possui mecanismos nativos para captura de proveniência de *workflows* executados em nuvens (Vöckler *et al.* 2011), entretanto os dados capturados se limitam a informações da máquina virtual utilizada e tempos de execução das tarefas paralelas. Além disso, todas as abordagens supracitadas apenas disponibilizam a proveniência para os usuários ao final da execução do *workflow*, fazendo com que mecanismos de *steering* e de monitoramento sejam mais complexos de serem desenvolvidos por terceiros e acoplados ao sistema. Tanto o Pegasus quanto o Swift armazenam os dados de proveniência na forma relacional, entretanto diferem na abordagem de consulta. O Pegasus possibilita ao usuário realizar consultas utilizando SPARQL (DuCharme 2011) enquanto que o Swift utiliza a linguagem SQL.

8.3.2 Abordagens Independentes de SGWfC

Diferentemente das abordagens anteriores, existem propostas de sistemas de captura de proveniência que atuam independente de SGWfC. Um exemplo destes sistemas é o ProvManager (Marinho *et al.* 2009, 2010a, 2010b, 2011). O ProvManager utiliza uma abordagem de instrumentação do *workflow* para incluir componentes de captura de proveniência. Entretanto a proveniência capturada não leva em consideração os dados de proveniência da execução distribuída. Uma das vantagens da utilização do ProvManager é que o mesmo trabalha com a questão do conceito de experimento, podendo associar os dados de proveniência de vários *workflows* independentes. O ProvManager trabalha com *workflows* gerados no Kepler, Taverna e VisTrails e armazena os dados de proveniência como fatos em ProLog, fazendo com que a consulta a estes dados seja realizada em ProLog. Outro sistema que captura a proveniência de forma semelhante é o Zoom (Kim et al. 2008). Entretanto, nenhuma destas abordagens está preparada para capturar dados de proveniência de *workflows* executados em nuvens de computadores. Diferentemente do ProvManager, o Zoom armazena os dados de forma relacional e a consulta é realizada utilizando SQL.

Existem abordagens que já oferecem mecanismos para a captura de proveniência em nuvens. Uma delas é o PASS (Simmhan *et al.* 2010), que é um sistema para coleta e armazenamento de proveniência distribuída. O PASS propõe a utilização de estruturas de armazenamento nativas da nuvem, como o *Simple Storage Service* (S3), o SimpleDB e o *Simple Queueing Service* (SQS) da Amazon EC2. Entretanto essa proposta se mostra deficiente em alguns pontos, como por exemplo, no caso do S3 que se mostrou ineficiente na consulta aos dados armazenados e o SimpleDB não garante a atomicidade dos dados. Além disso, toda a proposta depende das estruturas nativas da nuvem da Amazon EC2, com isso, caso o ambiente de execução não seja a nuvem da Amazon EC2 o sistema terá que ser totalmente adaptado. O PASS armazena os dados no formato XML e a consulta só pode ser realizada através da API fornecida.

Outra abordagem capaz de capturar proveniência em nuvens computacionais é a Matriohska (Cruz *et al.* 2008a) que inicialmente não focava em nuvens, mas foi estendida para alcançar tal objetivo (Paulino *et al.* 2009, 2010, 2011). Este trabalho de extensão se deu em conjunto com o autor desta tese e parte do modelo de proveniência da Matriohska foi herdada do modelo de proveniência do SciCumulus. Entretanto o foco da Matriohska é na captura de proveniência de mais alto nível como *workflows*

abstratos associados a conceitos da ontologia OvO, proposta por Cruz *et al.* (2011). Além disso, a Matriohska não captura automaticamente os dados do ambiente de nuvem, dependendo do cientista informa-los através de arquivos de manifesto, o que é contraproducente. A Matriohska armazena os dados de forma relacional e a consulta é realizada utilizando a linguagem SQL.

8.3.3 Considerações Sobre as Abordagens Existentes para Captura de Proveniência em Nuvens Computacionais

Os sistemas PASOA, Zoom, PASS, ProvManager e Matriohska são sistemas de captura de proveniência distribuída independentes de SGWfC. Sendo que os dois primeiros focam em proveniência centralizada e os três últimos sistemas são construídos focados no modelo SaaS, *i.e.* serviços que podem ser acoplados aos SGWfC existentes. As outras abordagens avaliadas são atreladas aos SGWfC existentes, oferecendo assim um alto grau de dependência com os mesmos. Dentre os sistemas avaliados, observa-se que grande parte utiliza a linguagem SQL para consultar os descritores de proveniência que foram armazenados, Apesar de algumas consultas serem complicadas de desenvolver com SQL, como por exemplo, determinar um fecho transitivo, esta linguagem oferece o apoio necessário para grande parte das consultas importantes para os cientistas. Um resumo das abordagens estudadas se encontra na Tabela 10.

Tabela 10 Quadro comparativo entre as principais abordagens de captura proveniência

Abordagem	***Escalabilidade***	***Apoio à Nuvem***	***Mecanismo***	***Persistência***	***Consulta***
VisTrails	Centralizado	Não	Interno	Relacional e XML	VTQL e SQL
Kepler	Centralizado	Não	Interno	Relacional	ProLog
Taverna	Centralizado	Não	Interno	Relacional e RDF	TriQL
Swift	Distribuído	Sim	Interno	Relacional	SQL
Pegasus	Distribuído	Sim	Interno	Relacional	SPARQL
PASS	Distribuído	Sim	Externo	XML	API
PASOA	Centralizado	Não	Externo	XML	XQuery
Zoom	Centralizado	Não	Externo	Relacional	SQL
ProvManager	Distribuído	Não	Externo	ProLog	ProLog
Matriohska	Distribuído	Sim	Externo	Relacional	SQL
SciCumulus	Distribuído	Sim	Externo	Relacional	SQL

Capítulo 9 - Conclusões

Este capítulo apresenta as conclusões gerais desta tese bem como suas principais contribuições e limitações. Finalizamos discutindo sobre as principais perspectivas de trabalhos futuros que podem ser desenvolvidos como consequência direta desta tese. A abordagem aqui apresentada se baseou na experiência anterior, obtida juntamente com o grupo de pesquisa da Profª Marta Lima de Queirós Mattoso, com a execução de *workflows* científicos em ambientes de PAD como *clusters*, grades e redes P2P (Barbosa et al. 2009, Coutinho et al. 2010, 2011, Cunha et al. 2009, Dias et al. 2010c, 2010b, Guerra et al. 2009, 2012, Mattoso et al. 2010a, Ogasawara et al. 2010, 2009a, Silva et al. 2010, 2011b). Desta forma, baseados nestas experiências anteriores, fomos capazes de propor uma abordagem que tratasse o problema da gerência de *workflows* científicos de larga escala executados em ambientes de nuvens computacionais.

A abordagem proposta para a gerência de *workflows* científicos de larga escala na nuvem é inspirada nas técnicas adaptativas de bancos de dados (Furtado *et al.* 2008, Lima *et al.* 2010), e apresenta uma série de serviços que englobam a distribuição, controle, monitoramento e captura de proveniência de *workflows* executados na nuvem. A abordagem, portanto, possibilita um tratamento sistemático para as tarefas e as atividades dos *workflows*, fazendo com que o cientista não tenha que se preocupar com a questão do paralelismo e da captura de proveniência distribuída e foque na especificação conceitual do experimento.

Um *workflow* científico a ser executado em paralelo usualmente demanda o processamento de grandes volumes de dados. Uma estratégia comum para prover o paralelismo é dividir as atividades do *workflow* em grãos menores, ou tarefas, que possam ser executados em paralelo. Nesta tese, cada tarefa do *workflow* a ser executada em paralelo recebe o nome de *cloud activity* e contém o programa a ser executado, os valores de parâmetros e dados a serem consumidos encapsulados em uma única unidade de trabalho. Este modelo é similar ao conceito de ativação de atividade em banco de dados (Bouganim *et al.* 1996), que foi também adaptado por Ogasawara *et al.* (2011) para o contexto de *workflows*. O conceito de *cloud activity* é fundamental para que possamos realizar uma distribuição das tarefas em ambientes de PAD de forma transparente e sem a intervenção do cientista.

Desta forma, dado um conjunto de *cloud activities* disponíveis para execução, devemos distribuí-las no ambiente de nuvem, *i.e.* nas diversas máquinas virtuais que processam dados em paralelo. Entretanto, distribuir, controlar e coletar proveniência destas *cloud activities* em ambientes de nuvem não é uma tarefa trivial de ser executada. Esta tarefa se torna complexa principalmente devido a dois fatores. O primeiro diz respeito à complexidade de se gerenciar centenas ou milhares de *cloud activities* que são executadas concorrentemente e a complexidade de se capturar, armazenar e consultar os dados de proveniência distribuída, ou seja, os dados de proveniência relativos a cada *cloud activity* que foi executada em diferentes máquinas virtuais. O segundo fator diz respeito às mudanças constantes que o ambiente de nuvem sofre e que o cientista não tem controle. Essas mudanças podem gerar alterações de desempenho durante o curso de execução do *workflow*, exigindo soluções adaptativas de escalonamento e dimensionamento de recursos. Rotinas de manutenção, desligamento ou movimentação de máquinas virtuais ficam a cargo do provedor de nuvem e o cientista não pode tomar nenhuma ação preventiva quanto a isso. Desta forma, é fundamental que qualquer solução que fosse proposta para a gerência de *workflows* em nuvens levasse em conta esses desafios inerentes a esta nova tecnologia.

Neste contexto, a abordagem apresentada nesta tese possibilita a gerência do *workflow* executado em nuvens computacionais por meio de algoritmos adaptativos de escalonamento e dimensionamento de recursos aliados a mecanismos de captura da proveniência distribuída que é baseada em um modelo que engloba tanto os descritores das características do ambiente de nuvem quanto os dados de proveniência prospectiva e retrospectiva do *workflow*. Nesta tese foi especificado um modelo de custo ponderado que foi acoplado ao algoritmo de escalonamento de forma que o cientista possa atribuir pesos aos critérios de acordo com a sua necessidade, a saber: tempo de execução, confiabilidade e custo monetário. Essa ponderação oferece maior flexibilidade ao cientista para realizar um ajuste fino nestes parâmetros.

A abordagem proposta também contempla um conjunto de serviços adicionais na gerência de *workflows* executados em nuvem. Embora consigamos executar um *workflow* com os componentes básicos propostos, esses serviços adicionais oferecem mecanismos de monitoramento, re-execução e previsão de demanda de recursos que são fundamentais no processo de experimentação científica assistida por computador.

Para que todos os experimentos fossem elaborados e avaliados, desenvolveu-se o SciCumulus, que é um mediador de execução paralela de *workflows* em ambientes de nuvens de computadores. A ideia chave do SciCumulus é servir como uma ponte entre o SGWfC instalado na máquina local do cientista e o ambiente de nuvem, distribuindo as *cloud activities* para a execução paralela. O SciCumulus foi utilizado em diversos experimentos envolvendo um *workflow* real na área de filogenia (Ocaña *et al.* 2011b), mostrando-se um mediador de execução capaz de apoiar a execução paralela de experimentos científicos em nuvens.

Neste sentido, a proveniência tem papel fundamental dentro do SciCumulus, tanto para auxiliar na reprodutibilidade dos experimentos quanto para propiciar o escalonamento adaptativo de uma forma eficiente. A proveniência no SciCumulus é armazenada no grão fino (*i.e. cloud activities*) em um banco de dados relacional e em tempo de execução. Assim, é possível monitorar o estado do *workflow* e avaliar os resultados disponíveis durante a execução. Este recurso é um diferencial, uma vez que as abordagens para gerência de *workflows* (paralelas ou não) disponibilizam estes dados para o cientista apenas ao final da execução.

9.1 Contribuições

A importância do apoio ao cientista na gerência dos experimentos científicos em larga escala tem sido o foco de discussões tanto em âmbito nacional quanto internacional. Especificamente no Brasil, o documento dos "Grandes Desafios" da SBC ressaltou a importância desta gerência na pesquisa brasileira nos próximos anos. Esta tese apresentou os resultados de uma pesquisa desenvolvida ao longo dos quatro últimos anos e suas principais contribuições são:

i. A definição de uma abordagem adaptativa para a execução de experimentos em nuvem que ofereça ao cientista melhor desempenho ao considerar (e se adaptar) as características e mudanças inerentes a este tipo de ambiente;
ii. Um modelo de custo ponderado para distribuição de *cloud activities* em nuvens computacionais, o que possibilita uma sintonia fina dos critérios de distribuição por parte do cientista;
iii. Um modelo de proveniência que considera tanto os descritores dos dados relativos ao ambiente de nuvem quanto aos dados relativos à estrutura e execução dos *workflows* (proveniência prospectiva e retrospectiva)

iv. A especificação e o desenvolvimento de uma arquitetura modular (SciCumulus) e de baixo acoplamento, sob a forma de um conjunto de serviços, que permite a gerência da execução paralela de *workflows* científicos em nuvens de computadores;
v. A avaliação desta arquitetura, por meio da execução de um *workflow* real de filogenia, que mostrou a capacidade do SciCumulus em gerenciar a execução paralela com o adicional da captura de proveniência distribuída. Os experimentos também evidenciaram a capacidade do SciCumulus em responder uma série de questões de proveniência definidas pelos pesquisadores.

Os resultados indicaram que o uso da abordagem proposta nesta tese propiciou uma gerência eficiente do *workflow* executado em paralelo juntamente com uma análise de proveniência (prospectiva e retrospectiva) mais completa do que apenas o uso dos mecanismos padrão oferecidos pelos atuais SGWfC (quando oferecidos).

Esta tese é a fusão de diversas publicações associadas à gerência de experimentos científicos executados em nuvens de computadores. As contribuições mencionadas podem ser sumarizadas na seleção das seguintes publicações realizadas durante o doutorado (ordenadas cronologicamente):

i. OLIVEIRA, D., Baião, F., Mattoso, M., (2010a), "Towards a Taxonomy for *Cloud* Computing from an e-Science Perspective", *Cloud* Computing: Principles, Systems and Applications, Nick Antonopoulos and Lee Gillam ed Heidelberg: Springer-Verlag
ii. OLIVEIRA, D., Ogasawara, E., Baião, F., Mattoso, M., (2010c), "SciCumulus: A Lightweight *Cloud* Middleware to Explore Many Task Computing Paradigm in Scientific *Workflows*". In: 3rd International Conference on *Cloud* Computing, p. 378–385, Washington, DC, USA.
iii. OLIVEIRA, D., Ogasawara, E., Baiao, F., Mattoso, M., (2010b), "An Adaptive Approach for *Workflow Activity* Execution in *Cloud*s". In: International Workshop on Challenges in e-Science - SBAC, p. 9-16, Petrópolis, RJ - Brazil.
iv. OLIVEIRA, D., Ocana, K., Ogasawara, E., Dias, J., Baiao, F., Mattoso, M., (2011a), "A Performance Evaluation of X-Ray Crystallography Scientific *Workflow* Using SciCumulus". In: IEEE International Conference on *Cloud* Computing (*CLOUD*), p. 708-715, Washington, D.C., USA.

v. OLIVEIRA, D., Ogasawara, E., Ocana, K., Baiao, F., Mattoso, M., (2011b), "An Adaptive Parallel Execution Strategy for *Cloud*-based Scientific *Workflows*", Concurrency and Computation: Practice and Experience, v. (online)

vi. Ocaña, K. A. C. S., OLIVEIRA, D., Ogasawara, E., Dávila, A. M. R., Lima, A. A. B., Mattoso, M., (2011b), "SciPhy: A *Cloud*-Based *Workflow* for Phylogenetic Analysis of Drug Targets in Protozoan Genomes", In: Norberto de Souza, O., Telles, G. P., Palakal, M. [orgs.] (eds), Advances in Bioinformatics and Computational Biology, , chapter 6832, Berlin, Heidelberg: Springer Berlin Heidelberg, p. 66-70.

vii. Viana, V., de OLIVEIRA, D., Mattoso, M., (2011a), "Towards a Cost Model for Scheduling Scientific *Workflows Activities* in *Cloud* Environments". In: 2011 IEEE World Congress on Services (SERVICES) 2011 IEEE World Congress on Services (SERVICES), p. 216-219

viii. Viana, V., OLIVEIRA, D., Dias, J., Ogasawara, E., Mattoso, M., (2011b), "SciCumulus-ECM: Um Serviço de Custos para a Execução de *Workflows* Científicos em Nuvens". In: Anais do XXVI SBBD SBBD, Florianópolis, SC, Brasil.

ix. Costa, F., OLIVEIRA, D., Mattoso M., (2011), "Heurísticas para Controle de Execução de Atividades de *Workflows* Científicos na Nuvem". In: Anais do Workshop de Teses e Dissertações em bancos de Dados - SBBD 2011 Simpósio Brasileiro de Banco de Dados 2011, Florianópolis, SC, Brasil.

x. Ocaña, K. A. C. S., OLIVEIRA, D., Dias, J., Ogasawara, E., Mattoso, M., (2011a), "Optimizing Phylogenetic Analysis Using SciHmm *Cloud*-based Scientific *Workflow*". In: 2011 IEEE Seventh International Conference on e-Science (e-Science), p. 190-197, Stockholm, Sweden.

xi. Ocaña, K. A. C. S., OLIVEIRA, D. de, Horta, F., Dias, J., Ogasawara, E., Mattoso, M., (2012), "Exploring Molecular Evolution Reconstruction Using a Parallel *Cloud*-based Scientific *Workflow*". In: Proceedings of the 2012 Brazilian Symposium on Bioinformatics (BSB 2012)2012 Brazilian Symposium on Bioinformatics (BSB 2012), Campo Grande, MT.

xii. OLIVEIRA, D., Ocaña, K. A. C. S., Ogasawara, E., Dias, J., Goncalves, J., Mattoso, M., (2012), "*Cloud*-based Phylogenomic Inference of Evolutionary Relationships: A Performance Study". In: Proceedings of the 2nd International Workshop on *Cloud* Computing and Scientific Applications (CCSA) 2nd

International Workshop on *Cloud* Computing and Scientific Applications (CCSA), Ottawa, Canadá.

xiii. OLIVEIRA, D., Ocaña, K. A. C. S. Baião, F.,, Mattoso, M., (2012), "A Provenance-based Adaptive Scheduling Heuristic for Parallel Scientific *Workflows* in *Cloud*s", Journal of Grid Computing, Submetido em 30 de Setembro de 2011.

Além das publicações diretamente ligadas ao contexto desta tese, foram realizadas pesquisas em outras fases do ciclo de vida do experimento científico. Desta forma, a pesquisa de doutoramento apresentou contribuições no ciclo de vida do experimento científico nas suas diferentes fases (Mattoso *et al.* 2008, 2010b): composição, execução e análise.

Na etapa de composição o foco foi em disponibilizar ao cientista uma forma de representar os *workflows* em maior nível de abstração, fazendo assim com que o cientista pudesse especificar o experimento em nível abstrato e esse fosse automaticamente mapeado para uma versão concreta e executável. Neste sentido, estas propostas foram baseadas na utilização de ontologias (Oliveira *et al.* 2007, 2009b, 2011b) ou em técnicas de engenharia de *software* como as linhas de produto (Oliveira *et al.* 2010d). Ainda na etapa de composição foram realizados também estudos sobre o reuso de *workflows*, tanto baseados na sua estrutura (Silva *et al.* 2010, 2011a) quanto nas anotações textuais associadas ao mesmo (Costa *et al.* 2010).

Na etapa de execução, que foi o foco principal desta tese, foram realizados diversos estudos de execução paralela de tarefas em ambientes PAD como *clusters*, grades e redes P2P (Barbosa et al. 2009, Coutinho et al. 2010, 2011, Cunha et al. 2009, Dias et al. 2010c, 2010b, Guerra et al. 2009, 2012, Mattoso et al. 2010a, Ogasawara et al. 2010, 2009a, Silva et al. 2010, 2011b). Esses estudos viabilizaram a proposta adaptativa aqui apresentada. E, finalmente, na etapa de análise foram feitos estudos acerca da coleta de dados de proveniência em ambientes heterogêneos (Paulino *et al.* 2009, 2010, 2011).

9.2 Limitações

Toda pesquisa científica, seja ela de doutoramento ou não, possui determinadas limitações. Desta maneira, algumas limitações foram identificadas, principalmente no

que tange o desenvolvimento do SciCumulus, a partir de uma análise criteriosa da prova de conceito implementada, dos *workflows* executados e dos dados de proveniência:

i. Por questões de simplificação na implementação, diversos componentes do SciCumulus estão hoje com a implementação dependente da API da Amazon EC2. Idealmente, os componentes devem ser independentes de ambiente. Entretanto, o ambiente da Amazon EC2 foi mais estável (senão o único disponível de fato) para que pudéssemos realizar os experimentos.
ii. A versão atual do SciCumulus é baseada em uma arquitetura de disco compartilhado. Infelizmente este não é o cenário arquitetural ideal, pois nem sempre poderemos contar com um disco compartilhado ao qual todas as máquinas virtuais tem acesso. Desta forma, uma limitação existente é que o SciCumulus atualmente não se encontra preparado para execuções em ambientes de arquitetura de disco não-compartilhado.
iii. Como não era o foco da tese a pesquisa sobre técnicas de transferência de dados otimizadas, o SciCumulus utiliza mecanismos de transferência de arquivos usuais como o SCP. Idealmente deveríamos utilizar protocolos mais elaborados como GridFTP (Kourtellis *et al.* 2008) de forma a diminuir o tempo de transferência de dados, que pode ser demasiadamente grande.
iv. Por limitações de tempo e de recursos, os experimentos apresentados nesta tese não exploraram experimentos com volumes de tarefas de centenas de milhares ou até milhões. Este tipo de estudo é necessário, para garantirmos que atendemos a qualquer tipo de experimento em larga escala.

Apesar das limitações apresentadas, a abordagem apresentada nesta tese dá um passo importante em direção à construção de um arcabouço definitivo para a gerência de experimentos em larga escala executados em nuvens de computadores.

9.3 Trabalhos Futuros

Gerenciar experimentos científicos executados em paralelo em ambientes distribuídos não é uma tarefa trivial, pelo contrário, é uma tarefa complexa e que demanda tempo, principalmente se o ambiente distribuído for um ambiente novo, ainda incipiente e caixa-preta como as nuvens de computadores. As abordagens discutidas nesta tese representam apenas os passos iniciais para o desenvolvimento de uma abordagem definitiva para a gerência de experimentos científicos em nuvens de computadores.

Com isso, a presente tese abriu uma linha de novas oportunidades de pesquisa. Considerando o estágio atual do desenvolvimento e as necessidades dos cientistas, algumas das perspectivas de trabalhos futuros são apresentadas a seguir. Estes trabalhos futuros são aderentes aos desafios relacionados aos experimentos científicos assistidos por *workflows* científicos (Gil *et al.* 2007). Os trabalhos futuros foram agrupados de acordo com a etapa do ciclo de vida ao qual estão associados.

Na etapa de composição, um dos desafios existentes é a questão da derivação de um *workflow* abstrato diretamente para a execução no SciCumulus. Por mais que o SciCumulus tenha sido uma evolução, ainda pode ser complicado para um cientista especificar um *workflow* diretamente em sua forma executável (XML). O ideal é que o cientista elaborasse uma especificação de mais alto nível, onde informasse restrições de execução e a partir desta especificação pudéssemos derivar *workflows* concretos no SciCumulus. Neste contexto, os conceitos de Linhas de Experimento (Ogasawara *et al.* 2009b) e ferramentas como a GExpLine (Oliveira *et al.* 2010d) se mostram fundamentais.

Na fase de execução, que detém a maior quantidade de trabalhos futuros, podemos citar as evoluções do SciCumulus para que seja independente da arquitetura do ambiente de nuvem, como por exemplo a questão dos discos não-compartilhados. Outra questão importante que se mostra como trabalhos futuros é a questão dos *workflows* dinâmicos. É interessante no ambiente de nuvem que possamos modificar a estrutura do *workflow* em tempo de execução de forma a atender restrições impostas pelos cientistas, como por exemplo critérios de desempenho ou qualidade dos dados. Por exemplo, imagine que um determinado *workflow* de filogenia utilize o programa MAFFT para alinhamento genético. Entretanto, durante a execução do *workflow*, podemos avaliar que o MAFFT não está gerando dados com qualidade de acordo com critérios explicitados pelos cientistas. Desta forma, é interessante que se possa explorar alternativas de programas na mesma execução do *workflow*. Essa questão da dinamicidade dos *workflows* é um problema de pesquisa relevante e ressaltado por diversos autores influentes da área (Deelman *et al.* 2009, Gil *et al.* 2007).

Outra questão que pode ser explorada na fase de execução do *workflow* em nuvens é a execução autonômica de *workflows* científicos. O desenvolvimento de um arcabouço autonômico onde todos os possíveis eventos fossem mapeados possibilitaria que o mesmo se adaptasse sem a intervenção do cientista aos mais variados problemas,

aumentando assim o nível de adaptabilidade da solução proposta. O arcabouço poderia se adaptar tanto as flutuações do meio quanto a erros e problemas na execução que pudessem ocorrer Ainda na fase de execução, outro ponto que merece atenção é a análise sobre a influência do grau de dependência dos dados consumidos e gerados no escalonamento adaptativo. No experimento executado nesta tese, as *cloud activities* possuíam pouca dependência de dados, o que fazia com que o paralelismo fosse relativamente simples. Entretanto, este não é o cenário de muitos experimentos como os de engenharia. Análises mais profundas nesse sentido ainda se fazem necessárias.

Outra questão importante inerente à fase de execução é a questão da federação de nuvens de computadores (do inglês *cloud federation*). O conceito de federação de nuvens se refere à integração de serviços de nuvem disponibilizados por diversos provedores. Desta forma, poderíamos executar um mesmo experimento utilizando máquinas virtuais instanciadas em diversos ambientes de nuvem com características diferenciadas. Entretanto, essa execução apresenta desafios de desempenho e de recuperação de desastres principalmente devido a distribuição geográfica. Esse cenário se torna bem provável, uma vez que centros de pesquisa estão investindo em nuvens privadas e certamente ocorrerá a colaboração entre centros de pesquisa tanto no que tange o conhecimento científico quanto no que se refere aos recursos computacionais.

A gerência dos dados produzidos na nuvem também é uma questão por si só demasiadamente complexa. Nesta tese, simplificamos a questão do armazenamento dos dados criando *workspaces* individuais onde os arquivos são armazenados e referenciados no banco de proveniência. Entretanto, esta abordagem possui restrições sérias, uma vez que os dados podem ser apagados e sua referência continua existindo no banco de dados de proveniência. São necessárias novas abordagens para o armazenamento de dados em nuvens que garantam algo similar as propriedades ACID (Elmasri e Navathe 2006) de sistemas de gerência de banco de dados. Abordagens como o Yahoo PNUTS (Cooper *et al.* 2008) proveem armazenamento em bancos de dados replicados, porém não garantem consistência dos dados, o que pode invalidar os descritores de proveniência.

Finalmente, na fase de análise, podem-se realizar diferentes pesquisas relacionadas à proveniência, principalmente no que tange a captura de dados específicos do domínio. Idealmente deve existir uma forma de analisar os produtos de dados das diversas *cloud activities* do *workflow* e extrair informações importantes dos arquivos

referentes a conhecimento específico do domínio, tudo isto com o agravante da distribuição de execução em ambientes de PAD. Informações do domínio podem ser de fundamental importância para o desenvolvimento de ferramentas de *steering* e para garantir a qualidade dos dados, Além disso, outro ponto importante é a própria fragmentação do banco de proveniência. Uma vez que fique caracterizado o problema da distribuição geográfica entre equipes, podemos aplicar técnicas clássicas de fragmentação de banco de dados (Özsu e Valduriez 2011). Outro ponto fundamental se refere a questão da pesquisa de métodos de integração do repositório de proveniência do SciCumulus com os repositórios de proveniência produzidos por SGWfC centralizados tais como VisTrails, Kepler e Taverna.

Referências Bibliográficas

Aalst, W. van der, Hee, K. van, (2002), *Workflow Management: Models, Methods, and Systems*. The MIT Press.

Abramson, D., Bethwaite, B., Enticott, C., Garic, S., Peachey, T., (2009a), "Parameter Space Exploration Using Scientific *Workflows*", *Computational Science – ICCS 2009*, , chapter 5544, Springer Berlin / Heidelberg, p. 104-113.

Abramson, D., Bethwaite, B., Enticott, C., Garic, S., Peachey, T., (2011), "Parameter Exploration in Science and Engineering Using Many-Task Computing", *IEEE Trans. Parallel Distrib. Syst.*, v. 22, n. 6 (jun.), p. 960–973.

Abramson, D., Bethwaite, B., Enticott, C., Garic, S., Peachey, T., Michailova, A., Amirriazi, S., Chitters, R., (2009b), "Robust *workflows* for science and engineering". In: *2nd Workshop on Many-Task Computing on Grids and Supercomputers*, p. 1-9, Portland, Oregon, USA.

Abramson, D., Enticott, C., Altinas, I., (2008), "Nimrod/K: towards massively parallel dynamic grid *workflows*". In: *Proc. of International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 1-11, Austin, Texas, USA.

Al-Azzoni, I., Down, D. G., (2008), "Dynamic scheduling for heterogeneous Desktop Grids". In: *Grid Computing, 2008 9th IEEE/ACM International Conference onGrid Computing, 2008 9th IEEE/ACM International Conference on*, p. 136-143

de Almeida-Neto, C., Liu, J., Wright, D. J., Mendrone-Junior, A., Takecian, P. L., Sun, Y., Ferreira, J. E., de Alencar Fischer Chamone, D., Busch, M. P., et al., (2011), "Demographic characteristics and prevalence of serologic markers among blood donors who use confidential unit exclusion (CUE) in São Paulo, Brazil: implications for modification of CUE polices in Brazil", *Transfusion*, v. 51, n. 1, p. 191–197.

Alonso, G., Casati, F., Kuno, H., Machiraju, V., (2010), *Web Services: Concepts, Architectures and Applications*. Softcover edition of hardcover 1st ed. 2004 edition ed. Springer.

Altintas, I., Barney, O., Jaeger-Frank, E., (2006), "Provenance Collection Support in the Kepler Scientific *Workflow* System", *Provenance and Annotation of Data*, , chapter 4145, Springer Berlin, p. 118-132.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S., (2004a), "Kepler: an extensible system for design and execution of scientific *workflows*". In: *Scientific and Statistical Database Management*, p. 423-424, Greece.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S., (2004b), "Kepler: an extensible system for design and execution of scientific *workflows*". In: *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, p. 423 - 424

Amazon EC2, (2010), *Amazon Elastic Compute Cloud (Amazon EC2)*, http://aws.amazon.com/ec2/.

Anderson, A. C., (2003), "The process of structure-based drug design", *Chemistry & Biology*, v. 10, n. 9 (set.), p. 787-797.

Anderson, D., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D., (2002),

"SETI@home: an experiment in public-resource computing", *Commun. ACM*, v. 45, n. 11 (nov.), p. 61, 56.

Anderson, E. W., Freire, J., Koop, D., Santos, E., Silva, C. T., (2007), *Provenance Challenge - Vistrails*. Disponível em: http://twiki.ipaw.info/bin/view/Challenge/VisTrails,

Anglano, C., Canonico, M., (2008), "Scheduling algorithms for multiple Bag-of-Task applications on Desktop Grids: A knowledge-free approach". In: *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium onParallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, p. 1-8

Anglano, C., Canonico, M., Guazzone, M., Botta, M., Rabellino, S., Arena, S., Girardi, G., (2008), "Peer-to-Peer Desktop Grids in the Real World: The ShareGrid Project". In: *8th IEEE International Symposium on Cluster Computing and the Grid, 2008. CCGRID '08*, p. 609-614

Antonopoulos, N., Gillam, L., (2010), *Cloud Computing: Principles, Systems and Applications*. 1st Edition. ed. Springer.

Applegate, D. L., Bixby, R. E., Chvatal, V., Cook, W. J., (2007), *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., et al., (2010), "A view of *cloud* computing", *Commun. ACM*, v. 53, n. 4, p. 50-58.

Asosheh, A., Danesh, M. H., (2008), "Comparison of OS level and hypervisor server virtualization". In: *Proceedings of the 8th conference on Systems theory and scientific computation*, p. 241-246, Rhodes, Greece.

Assayad, I., Girault, A., Kalla, H., (2004), "A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints". In: *2004 International Conference on Dependable Systems and Networks2004 International Conference on Dependable Systems and Networks*, p. 347- 356

Atkins, D. E., Droegemeier, K. K., Feldman, S. I., Garcia-molina, H., Klein, M. L., Messerschmitt, D. G., Messina, P., Ostriker, J. P., Wright, M. H., (2003), "Revolutionizing Science and Engineering Through Cyberinfrastructure : Report of the National Science Foundation Blue-Ribbon Advisory Panel on", *Science*, v. 81, n. 8, p. 1562–1567.

Atom, (2010), *RFC 4287 - The Atom Syndication Format*, http://tools.ietf.org/html/rfc4287.

Bäck, T., (1996), *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford, UK, Oxford University Press.

Baeza-Yates, R., Ribeiro-Neto, B., (2011), *Modern Information Retrieval: The Concepts and Technology behind Search (2nd Edition)*. 2 ed. Addison-Wesley Professional.

Barbosa, C. E., Ogasawara, E., Oliveira, D. de, Mattoso, M., (2009), "Paralelização de Tarefas de Mineração de Dados Utilizando *Workflows* Científicos". In: *V Workshop em Algoritmos e Aplicações de Mineração de Dados*, p. 91-96, Fortaleza, Ceara, Brazil.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R.,

Pratt, I., Warfield, A., (2003), "Xen and the art of virtualization". In: *Proceedings of the nineteenth ACM symposium on Operating systems principles*, p. 164-177, Bolton Landing, NY, USA.

Barker, A., van Hemert, J., (2008), "Scientific *Workflow*: A Survey and Research Directions", *Parallel Processing and Applied Mathematics*, , p. 746-753.

Bayucan, A., Henderson, R. L., Jones, J. P., (2000), *Portable Batch System Administration Guide*. Mountain View, CA, USA, Veridian Systems.

Beberg, A. L., Ensign, D. L., Jayachandran, G., Khaliq, S., Pande, V. S., (2009), "Folding@home: Lessons from eight years of volunteer distributed computing". In: *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, p. 1–8, Washington, DC, USA.

Bergsten, H., (2003), *JavaServer pages*. O'Reilly Media, Inc.

Berral, J. L., Gavalda, R., Torres, J., (2011), "Adaptive Scheduling on Power-Aware Managed Data-Centers Using Machine Learning". In: *2011 12th IEEE/ACM International Conference on Grid Computing (GRID)2011 12th IEEE/ACM International Conference on Grid Computing (GRID)*, p. 66-73

Berral, J. L., Goiri, Í., Nou, R., Julià, F., Guitart, J., Gavaldà, R., Torres, J., (2010), "Towards energy-aware scheduling in data centers using machine learning". In: *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, p. 215–224, New York, NY, USA.

Beveridge, W. I. B., (2004), *The Art of Scientific Investigation*. Blackburn Press.

Bezerra, E., (2007), *Princípios de análise e projeto de sistemas com UML*. 2. ed. rev. e atual. ed. Rio de Janeiro, Elsevier;Campus.

Boeres, C., Sardiña, I., Drummond, L., (2011), "An efficient weighted bi-objective scheduling algorithm for heterogeneous systems", *Parallel Computing*, v. 37, n. 8 (ago.), p. 349-364.

Bouganim, L., Florescu, D., Valduriez, P., (1996), "Dynamic load balancing in hierarchical parallel database systems". In: *Proceedings of the 22nd International Conference on Very Large DatabasesVLDB*, p. 436-447

Bowers, S., Mcphillips, T., Riddle, S., Anand, M. K., Ludäscher, B., (2008), "Kepler/pPOD: Scientific *Workflow* and Provenance Support for Assembling the Tree of Life". In: *Second International Provenance and Annotation Workshop*, p. 70-77, Salt Lake City, UT, USA.

Buarque de Holanda, A., (2004), *Aurélio - O Dicionário da Língua Portuguesa - C/ CD-ROM*. 3 ed. Positivo Editora.

Buneman, P., Khanna, S., Tan, W., (2001), "Why and Where: A Characterization of Data Provenance", *International Conference on Database Theory*, p. 316-330.

Buneman, P., Tan, W.-C., (2007), "Provenance in databases". In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, p. 1171–1173, New York, NY, USA.

Busset, J., Cabau, C., Meslin, C., Pascal, G., (2011), "PhyleasProg: a user-oriented web server for wide evolutionary analyses", *Nucleic Acids Research*, v. 39, n. Web Server (abr.), p. W479-W485.

Butler, M., (2011), "Android: Changing the Mobile Landscape", *IEEE Pervasive Computing*, v. 10, n. 1 (mar.), p. 4-7.

Buyya, R., Broberg, J., Goscinski, A. M., (2011), *Cloud Computing: Principles and Paradigms*. 1 ed. Wiley.

Buyya, R., Ranjan, R., Calheiros, R., (2009), "Modeling and Simulation of Scalable *Cloud* Computing Environments and the *Cloud*Sim Toolkit: Challenges". In: *Proc. of HPCS 2009HPCS 2009*, Leipzig, Germany.

Byun, E.-K., Kee, Y.-S., Kim, J.-S., Deelman, E., Maeng, S., (2011), "BTS: Resource capacity estimate for time-targeted science *workflows*", *Journal of Parallel and Distributed Computing*, v. 71, n. 6 (jun.), p. 848-862.

Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., Vo, H. T., (2006), "VisTrails: visualization meets data management". In: *SIGMOD International Conference on Management of Data*, p. 745-747, Chicago, Illinois, USA.

Carpenter, B., Getov, V., Judd, G., Skjellum, A., Fox, G., (2000), "MPJ: MPI-like message passing for Java", *Concurrency: Practice and Experience*, v. 12, n. 11, p. 1019-1038.

Carvalho, L. O. M., (2009), *Application of Scientific Workflows in the Design of Offshore Systems for Oil Production (in Portuguese)*. M.Sc. Dissertation, COPPE - Federal University of Rio de Janeiro , Civil Engineering Department

Cavalcanti, M. C., Targino, R., Baião, F., Rössle, S. C., Bisch, P. M., Pires, P. F., Campos, M. L. M., Mattoso, M., (2005), "Managing structural genomic *workflows* using web services", *Data & Knowledge Engineering*, v. 53, n. 1, p. 45-74.

Chamberlin, D., (2003), "XQuery: a query language for XML". In: *Proceedings of the 2003 ACM SIGMOD international conference on Management of dataACM SIGMOD international conference on Management of data*, San Diego, California, USA.

Chang, F., Ren, J., Viswanathan, R., (2010), "Optimal Resource Allocation in *Cloud*s". In: *2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*, p. 418-425

Chard, K., Bubendorfer, K., Komisarczuk, P., (2010), "High occupancy resource allocation for grid and *cloud* systems, a study with DRIVE". In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, p. 73–84, New York, NY, USA.

Chen, S.-H., Su, S.-Y., Lo, C.-Z., Chen, K.-H., Huang, T.-J., Kuo, B.-H., Lin, C.-Y., (2009), "PALM: a paralleled and integrated framework for phylogenetic inference with automatic likelihood model selectors", *PloS One*, v. 4, n. 12, p. e8116.

Chen, Y., Sun, X.-H., Wu, M., (2008), "Algorithm-system scalability of heterogeneous computing", *J. Parallel Distrib. Comput.*, v. 68, n. 11, p. 1403-1412.

Ching-Hsien Hsu, Tai-Lung Chen, (2010), "Adaptive Scheduling Based on Quality of Service in Heterogeneous Environments". In: *2010 4th International Conference on Multimedia and Ubiquitous Engineering (MUE)2010 4th International Conference on Multimedia and Ubiquitous Engineering (MUE)*, p. 1-6

Cirne, W., Brasileiro, F., Andrade, N., Costa, L., Andrade, A., Novaes, R., Mowbray, M., (2006), "Labs of the World, Unite!!!", *Journal of Grid Computing*, v. 4, n. 3, p. 225-246.

Cooper, B. F., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H.-A., Puz, N., Weaver, D., Yerneni, R., (2008), "PNUTS: Yahoo!'s hosted data serving platform", *Proc. VLDB Endow.*, v. 1, n. 2 (ago.), p. 1277–1288.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., (2009), *Introduction to Algorithms*. third edition ed. The MIT Press.

Costa, F., Oliveira, D., Mattoso M., (2011), "Heurísticas para Controle de Execução de Atividades de *Workflows* Científicos na Nuvem". In: *Anais do Workshop de Teses e Dissertações em bancos de Dados - SBBD 2011Simpósio Brasileiro de Banco de Dados 2011*, Florianópolis, SC, Brasil.

Costa, F., Oliveira, D., Ogasawara, E., Lima, A. A. B., Mattoso, M., (2010), "Athena: Text Mining Based Discovery of Scientific *Workflows* in Disperse Repositories". In: *Third International Workshop on REsource Discovery (RED)*, Paris, France.

Coutinho, F., Ogasawara, E., Oliveira, D., Braganholo, V., Lima, A. A. B., Dávila, A. M. R., Mattoso, M., (2011), "Many task computing for orthologous genes identification in protozoan genomes using Hydra", *Concurrency and Computation: Practice and Experience*, v. 23, n. 17 (dez.), p. 2326-2337.

Coutinho, F., Ogasawara, E., de Oliveira, D., Braganholo, V., Lima, A. A. B., Dávila, A. M. R., Mattoso, M., (2010), "Data parallelism in bioinformatics *workflows* using Hydra". In: *19th ACM International Symposium on High Performance Distributed Computing*, p. 507–515, New York, NY, USA.

Couvares, P., Kosar, T., Roy, A., Weber, J., Wenger, K., (2007), "*Workflow* Management in Condor", *Workflows for e-Science*, Springer, p. 357-375.

Cruz, S. M. S., (2011), *Uma estratégia de apoio à gerência de dados de proveniência em experimentos científicos*. Tese de Doutorado, Universidade Federal do Rio de Janeiro

Cruz, S. M. S. da, Barros, P. M., Bisch, P. M., Campos, M. L. M., Mattoso, M., (2008a), "Provenance Services for Distributed *Workflows*". In: *Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid*, p. 526-533

Cruz, S. M. S. da, Batista, V., Dávila, A. M. R., Silva, E., Tosta, F., Vilela, C., Campos, M. L. M., Cuadrat, R., Tschoeke, D., et al., (2008b), "OrthoSearch: a scientific *workflow* approach to detect distant homologies on protozoans". In: *Proc. of the ACM SAC*, p. 1282-1286, Fortaleza, Ceara, Brazil.

Cruz, S. M. S. da, Campos, M., Mattoso, M., (2009), "Towards a Taxonomy of Provenance in Scientific *Workflow* Management Systems". In: *IEEE International Workshop on Scientific Workflows*, Los Angeles, California, United States.

Cui Lin, Shiyong Lu, (2011), "Scheduling Scientific *Workflows* Elastically for *Cloud* Computing". In: *2011 IEEE International Conference on Cloud Computing (CLOUD)2011 IEEE International Conference on Cloud Computing (CLOUD)*, p. 746-747

Cui Lin, Shiyong Lu, Xubo Fei, Chebotko, A., Darshan Pai, Zhaoqiang Lai, Fotouhi, F., Jing Hua, (2009), "A Reference Architecture for Scientific *Workflow* Management Systems and the VIEW SOA Solution", *IEEE Transactions on Services Computing*, v. 2, n. 1 (mar.), p. 79-92.

Cunha, L., Oliveira, D., Ogasawara, E., Andrade, S., Zimbrão, G., Santos, M. de L., (2009), "On the use of scientific *workflows* for digital soil mapping". In: *Pedometrics*,

Beijing, China.

Dantas, M., (2005a), "Clusters Computacionais", *Computação Distribuída de Alto Desempenho: Redes, Clusters e Grids Computacionais*, 1 edRio de Janeiro: Axcel Books, p. 145-180.

Dantas, M., (2005b), "Grids Computacionais", *Computação Distribuída de Alto Desempenho: Redes, Clusters e Grids Computacionais*, 1 edRio de Janeiro: Axcel Books, p. 201-238.

Darwin, C., (2003), *The Origin Of Species: 150th Anniversary Edition*. Rep Anv ed. Signet Classics.

Daubin, V., Moran, N. A., Ochman, H., (2003), "Phylogenetics and the cohesion of bacterial genomes", *Science (New York, N.Y.)*, v. 301, n. 5634 (ago.), p. 829-832.

Davidson, S. B., Freire, J., (2008), "Provenance and scientific *workflows*: challenges and opportunities". In: *ACM SIGMOD international conference on Management of data*, p. 1345-1350, Vancouver, Canada.

Dávila, A. M. R., Mendes, P. N., Wagner, G., Tschoeke, D. A., Cuadrat, R. R. C., Liberman, F., Matos, L., Satake, T., Ocaña, K. A. C. S., et al., (2008), "ProtozoaDB: dynamic visualization and exploration of protozoan genomes", *Nucleic Acids Research*, v. 36, n. Database issue, p. D547-D552.

Dávila, Kary A. C. S. Ocaña, (2011), "Phylogenomics-Based Reconstruction of Protozoan Species Tree", *Evolutionary Bioinformatics* (jul.), p. 107.

Dean, J., Ghemawat, S., (2010), "MapReduce: a flexible data processing tool", *Commun. ACM*, v. 53 (jan.), p. 72–77.

Deelman, E., Gannon, D., Shields, M., Taylor, I., (2009), "*Workflows* and e-Science: An overview of *workflow* system features and capabilities", *Future Generation Computer Systems*, v. 25, n. 5, p. 528-540.

Deelman, E., Mehta, G., Singh, G., Su, M.-H., Vahi, K., (2007), "Pegasus: Mapping Large-Scale *Workflows* to Distributed Resources", *Workflows for e-Science*, Springer, p. 376-394.

Deitel, P., Deitel, H., (2010), *Java como programar*. 6a ed. São Paulo, Prentice Hall / Pearson.

Delange, Y., (2002), *Jean-Baptiste Lamarck*. Actes Sud.

Dereeper, A., Audic, S., Claverie, J.-M., Blanc, G., (2010), "BLAST-EXPLORER helps you building datasets for phylogenetic analysis", *BMC Evolutionary Biology*, v. 10, n. 1, p. 8.

Dereeper, A., Guignon, V., Blanc, G., Audic, S., Buffet, S., Chevenet, F., Dufayard, J.-F., Guindon, S., Lefort, V., et al., (2008), "Phylogeny.fr: robust phylogenetic analysis for the non-specialist", *Nucleic Acids Research*, v. 36, n. Web Server issue (jul.), p. W465-469.

Dias, J., Ogasawara, E., Mattoso, M., (2010a), "Paralelismo de dados científicos em *workflows* usando técnicas P2P". In: *IX Workshop de Teses e Dissertações em Banco de Dados*, p. 85-91, Belo Horizonte, Minas Gerais, Brazil.

Dias, J., Ogasawara, E., de Oliveira, D., Pacitti, E., Mattoso, M., (2010b), "Improving Many-Task computing in scientific *workflows* using P2P techniques". In: *Proceedings*

of the 3rd IEEE Workshop on Many-Task Computing on Grids and Supercomputers, p. 1-10, New Orleans, Louisiana, USA.

Dias, J., Ogasawara, E., Oliveira, D., Porto, F., Coutinho, A., Mattoso, M., (2011), "Supporting Dynamic Parameter Sweep in Adaptive and User-Steered *Workflow*". In: *6th Workshop on Workflows in Support of Large-Scale ScienceWORKS '11*, p. 31-36, Seattle, WA, USA.

Dias, J., Rodrigues, C., Ogasawara, E., Oliveira, D., Braganholo, V., Pacitti, E., Mattoso, M., (2010c), "SciMulator: Um Ambiente de Simulação de *Workflows* Científicos em Redes P2P". In: *Workshop P2P 2010*, p. 45-56, Gramado, Rio Grande do Sul - Brazil.

Do, C. B., Mahabhashyam, M. S. P., Brudno, M., Batzoglou, S., (2005), "ProbCons: Probabilistic consistency-based multiple sequence alignment", *Genome Research*, v. 15, n. 2 (fev.), p. 330 -340.

Dorigo, M., Stützle, T., (2004), *Ant Colony Optimization*. A Bradford Book.

DuCharme, B., (2011), *Learning SPARQL*. O'Reilly Media.

Eddy, S. R., (2009), "A new generation of homology search tools based on probabilistic inference", *Genome Informatics. International Conference on Genome Informatics*, v. 23, n. 1 (out.), p. 205-211.

Edgar, R. C., (2004), "MUSCLE: multiple sequence alignment with high accuracy and high throughput", *Nucleic Acids Research*, v. 32, n. 5 (mar.), p. 1792 -1797.

El-Khamra, Y., Kim, H., Jha, S., Parashar, M., (2010), "Exploring the Performance Fluctuations of HPC Workloads on *Clouds*". In: *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, p. 383–387, Washington, DC, USA.

Elmasri, R., Navathe, S. B., (2006), *Fundamentals of Database Systems*. 5 ed. Addison Wesley.

Evsukoff, A., Lima, B., Ebecken, N., (2011), "Long-Term Runoff Modeling Using Rainfall Forecasts with Application to the Iguaçu River Basin", *Water Resources Management*, v. 25, n. 3, p. 963-985.

Fahringer, T., Prodan, R., Rubing Duan, Nerieri, F., Podlipnig, S., Jun Qin, Siddiqui, M., Hong-Linh Truong, Villazon, A., et al., (2005), "ASKALON: a Grid application development and computing environment". In: *6th IEEE/ACM International Workshop on Grid Computing*, p. 122-131, Seattle, Washington, USA.

Fallsite, D., Walmsley, P., (2004). XML Schema Part 0: Primer Second Edition. Disponível em: http://www.w3.org/TR/xmlschema-0/. Acesso em: 3 nov 2011.

Fei, X., Lu, S., Lin, C., (2009), "A MapReduce-Enabled Scientific *Workflow* Composition Framework". In: *ICWS*, p. 663-670, Los Alamitos, CA, USA.

Felsenstein J, (1989), "PHYLIP - Phylogeny Inference Package (Version 3.2)", *Cladistics*, v. 5, p. 164-166.

Ferreira, J. E., Wu, Q., Malkowski, S., Pu, C., (2010), "Towards Flexible Event-Handling in *Workflows* Through Data States". In: *Proc. of the 2010 IEEE 6th World Congress on Services2010 IEEE 6th World Congress on Services*, p. 344-351, Miami, FL.

Fileto, R., Liu, L., Pu, C., Assad, E. D., Medeiros, C. B., (2003), "POESIA: An ontological *workflow* approach for composing Web services in agriculture", *The VLDB Journal*, v. 12, n. 4 (nov.), p. 352–367.

Foster, I., (1995), *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison Wesley.

Foster, I., Kesselman, C., (1996), "Globus: A Metacomputing Infrastructure Toolkit", *INTERNATIONAL JOURNAL OF SUPERCOMPUTER APPLICATIONS*, v. 11, p. 115-128.

Foster, I., Kesselman, C., (2004), *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann.

Fowler, M., (2003), *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3 ed. Addison-Wesley Professional.

Franz, D., Tao, J., Marten, H., Streit, A., (2011), "A *Workflow* Engine for Computing *Clouds*". *CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, p. 1-6

Freire, J., Koop, D., Santos, E., Silva, C. T., (2008a), "Provenance for Computational Tasks: A Survey", *Computing in Science and Engineering, v.10*, n. 3, p. 11-21.

Freire, J., Koop, D., Santos, E., Silva, C. T., (2008b), "Provenance for Computational Tasks: A Survey", *Computing in Science and Engineering*, v. 10 (maio.), p. 11–21.

Fuller, U., Johnson, C. G., Ahoniemi, T., Cukierman, D., Hernán-Losada, I., Jackova, J., Lahtinen, E., Lewis, T. L., Thompson, D. M., et al., (2007), "Developing a computer science-specific learning taxonomy", *SIGCSE Bull.*, v. 39, n. 4 (dez.), p. 152–170.

Furtado, C., Lima, A., Pacitti, E., Valduriez, P., Mattoso, M., (2008), "Adaptive hybrid partitioning for OLAP query processing in a database cluster", *International Journal of High Performance Computing and Networking*, v. 5, n. 4, p. 251-262.

Futuyma, D. J., Futuyma, (1997), *Evolutionary Biology*. 3 Sub ed. Sinauer Associates.

Gadelha, L. M. R., Mattoso, M., (2008), "Kairos: An Architecture for Securing Authorship and Temporal Information of Provenance Data in Grid-Enabled *Workflow* Management Systems". In: *International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES 2008)*, p. 597-602

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., et al., (2007), "Examining the Challenges of Scientific *Workflows*", *Computer*, v. 40, n. 12, p. 24-32.

Gilbert, D., (2003), "Sequence file format conversion with command-line readseq", *Current Protocols in Bioinformatics / Editoral Board, Andreas D. Baxevanis ... [et Al*, v. Appendix 1 (fev.), p. Appendix 1E.

Glatard, T., Sipos, G., Montagnat, J., Farkas, Z., Kacsuk, P., (2007), "*Workflow*-Level Parametric Study Support by MOTEUR and the P-GRADE Portal", *Workflows for e-Science*, Springer, p. 279-299.

Goble, C., Wroe, C., Stevens, R., (2003), "The myGrid project: services, architecture and demonstrator". In: *Proc. of the UK e-Science All Hands Meeting*, p. 595-602, Nottingham, UK.

Goiri, Í., Julià, F., Nou, R., Berral, J. L., Guitart, J., Torres, J., (2010), "Energy-Aware

Scheduling in Virtualized Datacenters". In: *Cluster Computing, IEEE International Conference on*, p. 58-67, Los Alamitos, CA, USA.

Goldberg, D. E., (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*. 1 ed. Addison-Wesley Professional.

Goncalez, T. T., Sabino, E. C., Capuani, L., Liu, J., Wright, D. J., Walsh, J. H., Ferreira, J. E., Chamone, D. A., Busch, M. P., et al., (2011), "Blood transfusion utilization and recipient survival at Hospital das Clinicas in São Paulo, Brazil", *Transfusion*, p. no–no.

Gorder, P. F., (2008), "Coming Soon: Research in a *Cloud*", *Computing in Science and Engg.*, v. 10, n. 6, p. 6-10.

Gorton, I., Greenfield, P., Szalay, A., Williams, R., (2008), "Data-Intensive Computing in the 21st Century", *Computer*, v. 41, n. 4, p. 30-32.

Governato, F., Brook, C., Mayer, L., Brooks, A., Rhee, G., Wadsley, J., Jonsson, P., Willman, B., Stinson, G., et al., (2010), "Bulgeless dwarf galaxies and dark matter cores from supernova-driven outflows", *Nature*, v. 463, n. 7278 (jan.), p. 203-206.

Gray, J., Liu, D. T., Nieto-Santisteban, M., Szalay, A., DeWitt, D. J., Heber, G., (2005), "Scientific data management in the coming decade", *SIGMOD Record*, v. 34, n. 4, p. 34-41.

Groth, P., Luck, M., Moreau, L., (2004), "A protocol for recording provenance in service-oriented grids", *IN PROCEEDINGS OF THE 8TH INTERNATIONAL CONFERENCE ON PRINCIPLES OF DISTRIBUTED SYSTEMS (OPODIS'04*, v. 3544, p. 124-139.

Guerra, G. M., Rochinha, F. A., (2009a), "Uncertainty quantification in fluid-structure interaction via sparse grid stochastic collocation method". In: *30th Iberian-Latin-American Congress on Computational Methods in Engineering, 2009, Buzios*

Guerra, G. M., Rochinha, F. A., (2009b), "A Sparse Grid Method Applied to Stochastic Fluid-Structure Interaction". In: *COBEM, 2009, Gramado, Brazil*

Guerra, G. M., Rochinha, F. A., (2010), "Stochastic modeling of Flow-Structure Interaction using a Sparse Grid Stochastic Collocation Method". In: *IV European Congress on Computational Mechanics, 2010, Paris*

Guerra, G., Rochinha, F., Elias, R., Coutinho, A., Braganholo, V., Oliveira, D. de, Ogasawara, E., Chirigati, F., Mattoso, M., (2009), "Scientific *Workflow* Management System Applied to Uncertainty Quantification in Large Eddy Simulation". In: *Congresso Ibero Americano de Métodos Computacionais em Engenharia*, p. 1-13, Búzios, Rio de Janeiro, Brazil.

Guerra, G., Rochinha, F., Elias, R., Oliveira, D., Ogasawara, E., Dias, J., Mattoso, M., Coutinho, A. L. G. A., (2012), "Uncertainty Quantification in Computational Predictive Models for Fluid Dynamics Using *Workflow* Management Engine", *International Journal for Uncertainty Quantification*, v. 2, n. 1, p. 53-71.

Guindon, S., Lethiec, F., Duroux, P., Gascuel, O., (2005), "PHYML Online--a web server for fast maximum likelihood-based phylogenetic inference", *Nucleic Acids Research*, v. 33, n. Web Server issue (jul.), p. W557-559.

Hadoop, (2012), *Apache Hadoop Web page*, http://hadoop.apache.org/.

Hartman, A. L., Riddle, S., McPhillips, T., Ludäscher, B., Eisen, J. A., (2010), "Introducing W.A.T.E.R.S.: a *Workflow* for the Alignment, Taxonomy, and Ecology of

Ribosomal Sequences", *BMC Bioinformatics*, v. 11, n. 1, p. 317.

He, Q., Zhou, S., Kobler, B., Duffy, D., McGlynn, T., (2010), "Case study for running HPC applications in public *clouds*". In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, p. 395–401, New York, NY, USA.

Hey, T., Tansley, S., Tolle, K., (2009), *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research.

Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B., Good, J., (2008), "On the use of *cloud* computing for scientific *workflows*". In: *IEEE Fourth International Conference on eScience (eScience 2008), Indianapolis, USA*, p. 7–12

Howe, B., Vo, H., Silva, C., Freire, J., (2009), "Query-driven visualization in the *cloud* with mapreduce". In: *Proceedings of the Fourth Annual Workshop on Ultrascale VisualizationFourth Annual Workshop on Ultrascale Visualization*, Portland, Oregon, USA.

HTTP, (2010), *HTTP - Hypertext Transfer Protocol Overview*, http://www.w3.org/Protocols/.

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M. R., Li, P., Oinn, T., (2006a), "Taverna: a tool for building and running *workflows* of services", *Nucleic Acids Research*, v. 34, n. 2, p. 729-732.

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M. R., Li, P., Oinn, T., (2006b), "Taverna: a tool for building and running *workflows* of services", *Nucleic Acids Research*, v. 34, n. 2, p. 729-732.

Iosup, A., Jan, M., Sonmez, O., Epema, D., (2007), "The Characteristics and Performance of Groups of Jobs in Grids", In: Kermarrec, A.-M., Bougé, L., Priol, T. [orgs.] (eds), *Euro-Par 2007 Parallel Processing*, , chapter 4641, Berlin, Heidelberg: Springer Berlin Heidelberg, p. 382-393.

Jackson, K. R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H. J., Wright, N. J., (2010), "Performance Analysis of High Performance Computing Applications on the Amazon Web Services *Cloud*". In: *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, p. 159–168, Washington, DC, USA.

Jacob, J. C., Katz, D. S., Berriman, G. B., Good, J. C., Laity, A. C., Deelman, E., Kesselman, C., Singh, G., Su, M.-H., et al., (2009), "Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking", *Int. J. Comput. Sci. Eng.*, v. 4, n. 2, p. 73-87.

Jarrard, R. D., (2001), *Scientific Methods*. Online book, Url.: http://emotionalcompetency.com/sci/booktoc.html.

Jensen, M., Schwenk, J., Gruschka, N., Iacono, L. L., (2009), "On Technical Security Issues in *Cloud* Computing". In: *Proceedings of the 2009 IEEE International Conference on Cloud Computing*, p. 109-116

Jones, M. O., Koutsovoulos, G. D., Blaxter, M. L., (2011), "iPhy: an integrated phylogenetic workbench for supermatrix analyses", *BMC Bioinformatics*, v. 12, n. 1, p. 30.

Jr, E. B. W., (1991), *An Introduction to Scientific Research*. Rev Sub ed. Dover

Publications.

Jr, H. G. G., (2002), *Scientific Method in Practice*. 1st ed. Cambridge University Press.

JSON, (2010), *JSON*, http://www.json.org/.

Juristo, N., Moreno, A. M., (2001), *Basics of Software Engineering Experimentation*. 1 ed. Springer.

Katoh, K., Toh, H., (2008), "Recent developments in the MAFFT multiple sequence alignment program", *Briefings in Bioinformatics*, v. 9, n. 4 (jul.), p. 286-298.

Keane, T. M., Creevey, C. J., Pentony, M. M., Naughton, T. J., Mclnerney, J. O., (2006), "Assessment of methods for amino acid matrix selection and their use on empirical data shows that ad hoc assumptions for choice of matrix are not justified", *BMC Evolutionary Biology*, v. 6, p. 29.

Keane, T. M., Naughton, T. J., McInerney, J. O., (2007), "MultiPhyl: a high-throughput phylogenomics webserver using distributed computing", *Nucleic Acids Research*, v. 35, n. Web Server issue (jul.), p. W33-37.

Kertész, A., Sipos, G., Kacsuk, P., (2007), "Brokering Multi-grid *Workflows* in the P-GRADE Portal", In: Lehner, W., Meyer, N., Streit, A., Stewart, C. [orgs.] (eds), *Euro-Par 2006: Parallel Processing*, , chapter 4375, Berlin, Heidelberg: Springer Berlin Heidelberg, p. 138-149.

Kim, J., Deelman, E., Gil, Y., Mehta, G., Ratnakar, V., (2008), "Provenance trails in the Wings-Pegasus system", *Concurrency and Computation: Practice & Experience*, v. 20 (abr.), p. 587–597.

Kim, W., Kim, S. D., Lee, E., Lee, S., (2009), "Adoption issues for *cloud* computing". In: *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, p. 3-6, Kuala Lumpur, Malaysia.

Kllapi, H., Sitaridi, E., Tsangaris, M. M., Ioannidis, Y., (2011), "Schedule optimization for data processing flows on the *cloud*". , p. 289

Kloh, H., Schulze, B., Mury, A., Pinto, R. C. G., (2010), "A scheduling model for *workflows* on grids and *clouds*". In: *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, p. 3:1–3:6, New York, NY, USA.

Kloh, H., Schulze, B., Pinto, R., Mury, A., (2011), "A bi-criteria scheduling process with CoS support on grids and *clouds*", *Concurrency and Computation: Practice and Experience*

Kourtellis, N., Prieto, L., Iamnitchi, A., Zarrate, G., Fraser, D., (2008), "Data transfers in the grid: workload analysis of globus GridFTP". In: *Proceedings of the 2008 international workshop on Data-aware distributed computing*, p. 29–38, New York, NY, USA.

Lander, E. S., (2001), "Initial sequencing and analysis of the human genome", *Nature*, v. 409, n. 6822 (fev.), p. 860-921.

Larsen, R. J., Marx, M. L., (2011), *Introduction to Mathematical Statistics and Its Applications, An*. 5 ed. Prentice Hall.

Lassmann, T., Sonnhammer, E. L. L., (2005), "Kalign--an accurate and fast multiple sequence alignment algorithm", *BMC Bioinformatics*, v. 6, p. 298.

Laszewski, G., Hategan, M., Kodeboyina, D., (2007), "Java CoG Kit *Workflow*",

Workflows for e-Science, Springer, p. 340-356.

Lee, K., Paton, N. W., Sakellariou, R., Fernandes, A. A. A., (2011), "Utility functions for adaptively executing concurrent *workflows*", *Concurrency and Computation: Practice and Experience*, v. 23, n. 6 (abr.), p. 646-666.

Lemos, M., Casanova, M. A., Seibel, L. F. B., Macedo, J. A. F., Miranda, A. B., (2004), "Ontology-Driven *Workflow* Management for Biosequence Processing Systems", In: Galindo, F., Takizawa, M., Traunmüller, R. [orgs.] (eds), *Database and Expert Systems Applications*, Berlin, Heidelberg: Springer Berlin Heidelberg, p. 781-790.

Leusse, P. de, Periorellis, P., Watson, P., Maierhofer, A., (2008), "Secure & Rapid Composition of Infrastructure Services in the *Cloud*". In: *Proceedings of the 2008 Second International Conference on Sensor Technologies and Applications*, p. 770-775

Li, A., Yang, X., Kandula, S., Zhang, M., (2010), "*Cloud*Cmp: comparing public *cloud* providers". In: *Proceedings of the 10th annual conference on Internet measurement*, p. 1–14, New York, NY, USA.

Lima, A., Mattoso, M., Valduriez, P., (2010), "Adaptive Virtual Partitioning for OLAP Query Processing in a Database Cluster", *Journal of Information and Data Management*, v. 1, n. 1, p. 75-88.

Lin, C., Barga, R., Guo, D., Jackson, J., (2010), "*CloudWorkflow* Service: Automatically Scaling Scientific *Workflows* on Demand". In: *Proceedings of the Microsoft eScience WorkshopMicrosoft eScience Workshop*, p. 105-107, Pittsburgh, Pennsylvania.

Lin, C.-Y., Lin, F.-K., Lin, C. H., Lai, L.-W., Hsu, H.-J., Chen, S.-H., Hsiung, C. A., (2005), "POWER: PhylOgenetic WEb Repeater--an integrated and user-optimized framework for biomolecular phylogenetic analysis", *Nucleic Acids Research*, v. 33, n. Web Server issue (jul.), p. W553-556.

Lins, E. F., Elias, R. N., Guerra, G. M., Rochinha, F. A., Coutinho, A. L. G. A., (2009), "Edge-based finite element implementation of the residual-based variational multiscale method", *International Journal for Numerical Methods in Fluids*, v. 61, n. 1, p. 1–22.

Ludascher, B., Bowers, S., McPhillips, T., Podhorszki, N., (2006), "Scientific *Workflows*: More e-Science Mileage from Cyberinfrastructure". In: *Second IEEE International Conference on e-Science and Grid Computing*, p. 145–153, Washington, DC, USA.

Mao, H., Hu, S., Zhang, Z., Xiao, L., Ruan, L., (2011), "A load-driven task scheduler with adaptive DSC for MapReduce". In: *Proceedings - 2011 IEEE/ACM International Conference on Green Computing and Communications, GreenCom 2011*, p. 28-33

Marinho, A., Murta, L., Werner, C., Braganholo, V., Cruz, S. M. S. da, Mattoso, M., (2009), "A Strategy for Provenance Gathering in Distributed Scientific *Workflows*". In: *IEEE International Workshop on Scientific Workflows*, p. 344-347, Los Angeles, California, United States.

Marinho, A., Murta, L., Werner, C., Braganholo, V., Cruz, S. M. S. da, Ogasawara, E., Mattoso, M., (2011), "ProvManager: a provenance management system for scientific *workflows*", *Concurrency and Computation: Practice and Experience*, v. (online)

Marinho, A., Murta, L., Werner, C., Braganholo, V., Cruz, S. M. S., Ogasawara, E., Mattoso, M., (2010a), "Managing Provenance in Scientific *Workflows* with ProvManager". In: *International Workshop on Challenges in e-Science - SBAC*, p. 17-

24, Petrópolis, RJ - Brazil.

Marinho, A., Murta, L., Werner, C., Braganholo, V., Ogasawara, E., Cruz, S. M. S., Mattoso, M., (2010b), "Integrating Provenance Data from Distributed *Workflow* Systems with ProvManager", In: McGuinness, D. L., Michaelis, J. R., Moreau, L. [orgs.] (eds), *Provenance and Annotation of Data and Processes*, Berlin, Heidelberg: Springer Berlin Heidelberg, p. 286-288.

Marinos, A., Briscoe, G., (2009), "Community *Cloud* Computing". In: *Proceedings of the 1st International Conference on Cloud Computing*, p. 472-484, Beijing, China.

Martinho, W., Ogasawara, E., Oliveira, D., Chirigati, F., Santos, I., Travassos, G. H. T., Mattoso, M., (2009), "A Conception Process for Abstract *Workflows*: An Example on Deep Water Oil Exploitation Domain". In: *5th IEEE International Conference on e-Science*, Oxford, UK.

Matsunaga, A., Tsugawa, M., Fortes, J., (2008), "*Cloud*BLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications", *IEEE eScience 2008*, p. 229, 222.

Mattos, A., (2008), *Gerência de Dados Genômicos em Workflows de Bioinformática*. Dissertação de Mestrado, Universidade Federal do Rio de Janeiro

Mattos, A., Silva, F., Ruberg, N., Cruz, M., (2008), "Gerência de *Workflows* Científicos: Uma Análise Crítica no Contexto da Bioinformática", *COPPE/UFRJ*, n. Relatório técnico

Mattoso, M., Coutinho, A., Elias, R., Oliveira, D., Ogasawara, E., (2010a), "Exploring Parallel Parameter Sweep in Scientific *Workflows*". In: *WCCM - World Congress on Computational Mechanics*, Australia.

Mattoso, M., Werner, C., Travassos, G., Braganholo, V., Murta, L., (2008), "Gerenciando Experimentos Científicos em Larga Escala". In: *SEMISH - CSBC*, p. 121-135, Belém, Pará - Brasil.

Mattoso, M., Werner, C., Travassos, G., Braganholo, V., Murta, L., Ogasawara, E., Oliveira, F., Martinho, W., (2009), "Desafios no Apoio à Composição de Experimentos Científicos em Larga Escala". In: *SEMISH - CSBC*, p. 307-321, Bento Gonçalves, Rio Grande do Sul, Brazil.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Murta, L., Ogasawara, E., Oliveira, D., Cruz, S. M. S. da, Martinho, W., (2010b), "Towards Supporting the Life Cycle of Large-scale Scientific Experiments", *International Journal of Business Process Integration and Management*, v. 5, n. 1, p. 79–92.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Ogasawara, E., Oliveira, D., Cruz, S. M., Martinho, W., Murta, L., (2010c), "Towards supporting the life cycle of large scale scientific experiments", *International Journal of Business Process Integration and Management*, v. 5, n. 1, p. 79.

Mc Evoy, G., Schulze, B., (2011), "Understanding scheduling implications for scientific applications in *cloud*s". In: *Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science*, p. 3:1–3:6, New York, NY, USA.

McLaughlin, B., (2001), *Java & XML*. 2nd ed. ed. Beijing ;;Cambridge [Mass.], O'Reilly.

Medeiros, C. B., Perez-Alcazar, J., Digiampietri, L., G. Z. Pastorello, J., Santanche, A.,

Torres, R. S., Madeira, E., Bacarin, E., (2005), "WOODSS and the Web: annotating and reusing scientific *workflows*", *SIGMOD Record*, v. 34, n. 3, p. 18-23.

Mednieks, Z., Dornin, L., Meike, G. B., Nakamura, M., (2011), *Programming Android*. 1 ed. O'Reilly Media.

Meyer, L., Scheftner, D., Vöckler, J., Mattoso, M., Wilde, M., Foster, I., (2007), "An Opportunistic Algorithm for Scheduling *Workflows* on Grids", *VECPAR 2006*, 1 ed, p. 1-12.

Moreau, L., Freire, J., Futrelle, J., McGrath, R., Myers, J., Paulson, P., (2008a), "The Open Provenance Model: An Overview", *Provenance and Annotation of Data and Processes*, , p. 323-326.

Moreau, L., Ludäscher, B., Altintas, I., Barga, R. S., Bowers, S., Callahan, S., George Chin, J., Clifford, B., Cohen, S., et al., (2008b), "Special Issue: The First Provenance Challenge", *Concurrency and Computation: Practice and Experience*, v. 20, n. 5, p. 409-418.

Moreau, L., Missier, P., (2011). The PROV Data Model and Abstract Syntax Notation. *W3C Working Draft. (Work in progress.)*. Disponível em: http://www.w3.org/TR/2011/WD-prov-dm-20111018/.

Moreau, L., Missier, P., Belhajjame, K., Cresswell, S., Golden, R., Groth, P., Miles, S., Sahoo, S., (2011). The PROV Data Model and Abstract Syntax Notation. Disponível em: http://www.w3.org/TR/2011/WD-prov-dm-20111018/. Acesso em: 14 dez 2011.

Muehlenbein, M. P., (2010), *Human Evolutionary Biology*. 1 ed. Cambridge University Press.

Napper, J., Bientinesi, P., (2009), "Can *cloud* computing reach the top500?". In: *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, p. 17-20, Ischia, Italy.

Nie, L., Xu, Z., (2009), "An adaptive scheduling mechanism for elastic grid computing". In: *SKG 2009 - 5th International Conference on Semantics, Knowledge, and Grid*, p. 184-191

NIST, (2010), *NIST definition of Cloud Computing*, http://csrc.nist.gov/groups/SNS/*cloud*-computing/.

O'Reilly, T., Milstein, S., (2011), *The Twitter Book*. 2 ed. O'Reilly Media.

OAuth, (2010), *OAuth Community Site*, http://oauth.net/.

Ocaña, K. A. C. S., Oliveira, D. de, Horta, F., Dias, J., Ogasawara, E., Mattoso, M., (2012), "Exploring Molecular Evolution Reconstruction Using a Parallel *Cloud*-based Scientific *Workflow*". In: *Proceedings of the 2012 Brazilian Symposium on Bioinformatics (BSB 2012)2012 Brazilian Symposium on Bioinformatics (BSB 2012)*, Campo Grande, MT.

Ocaña, K. A. C. S., Oliveira, D., Dias, J., Ogasawara, E., Mattoso, M., (2011a), "Optimizing Phylogenetic Analysis Using SciHmm *Cloud*-based Scientific *Workflow*". In: *2011 IEEE Seventh International Conference on e-Science (e-Science)IEEE e-Science 2011*, p. 190-197, Stockholm, Sweden.

Ocaña, K. A. C. S., Oliveira, D., Ogasawara, E., Dávila, A. M. R., Lima, A. A. B., Mattoso, M., (2011b), "SciPhy: A *Cloud*-Based *Workflow* for Phylogenetic Analysis of Drug Targets in Protozoan Genomes", In: Norberto de Souza, O., Telles, G. P., Palakal,

M. [orgs.] (eds), *Advances in Bioinformatics and Computational Biology*, , chapter 6832, Berlin, Heidelberg: Springer Berlin Heidelberg, p. 66-70.

Ochi, L. S., (1998), *Algoritmos genéticos: origem e evolução*. 1 ed. Sao Paulo, Sociedade Brasileira de Matemática Aplicada e Computacional.

Ogasawara, E., Dias, J., Oliveira, D., Porto, F., Valduriez, P., Mattoso, M., (2011), "An Algebraic Approach for Data-Centric Scientific *Workflows*", *Proc. of VLDB Endowment*, v. 4, n. 12, p. 1328-1339.

Ogasawara, E., Dias, J., Oliveira, D., Rodrigues, C., Pivotto, C., Antas, R., Braganholo, V., Valduriez, P., Mattoso, M., (2010), "A P2P approach to many tasks computing for scientific *workflows*". In: *VECPAR'10*, p. 327–339, Berlin, Heidelberg.

Ogasawara, E., Oliveira, D., Chirigati, F., Barbosa, C. E., Elias, R., Braganholo, V., Coutinho, A., Mattoso, M., (2009a), "Exploring many task computing in scientific *workflows*". In: *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and SupercomputersMTAGS'09*, p. 1-10, Portland, Oregon, USA.

Ogasawara, E., Paulino, C., Murta, L., Werner, C., Mattoso, M., (2009b), "Experiment Line: Software Reuse in Scientific *Workflows*". In: *Scientific and Statistical Database ManagementSSDBM 2009*, p. 264-272, New Orleans, Louisiana, USA.

Oliveira, D., Baião, F., Mattoso, M., (2007), "MiningFlow: Adding Semantics to Text Mining *Workflows*". In: *First Poster Session of the Brazilian Symposium on Databases*, p. 15-18, João Pessoa, PB - Brazil.

Oliveira, D., Baião, F., Mattoso, M., (2010a), "Towards a Taxonomy for *Cloud* Computing from an e-Science Perspective", *Cloud Computing: Principles, Systems and Applications (to be published)*, Nick Antonopoulos and Lee Gillam edHeidelberg: Springer-Verlag

Oliveira, D., Cunha, L., Tomaz, L., Pereira, V., Mattoso, M., (2009a), "Using Ontologies to Support Deep Water Oil Exploration Scientific *Workflows*". In: *IEEE International Workshop on Scientific Workflows*, p. 364-367, Los Angeles, California, United States.

Oliveira, D., Ocaña, K. A. C. S., Ogasawara, E., Dias, J., Goncalves, J., Mattoso, M., (2012), "*Cloud*-based Phylogenomic Inference of Evolutionary Relationships: A Performance Study". In: *Proceedings of the 2nd International Workshop on Cloud Computing and Scientific Applications (CCSA)2nd International Workshop on Cloud Computing and Scientific Applications (CCSA)*, Ottawa, Canadá.

Oliveira, D., Ocana, K., Ogasawara, E., Dias, J., Baiao, F., Mattoso, M., (2011a), "A Performance Evaluation of X-Ray Crystallography Scientific *Workflow* Using SciCumulus". In: *IEEE International Conference on Cloud Computing (CLOUD)IEEE Cloud*, p. 708-715, Washington, D.C., USA.

Oliveira, D., Ogasawara, E., Baiao, F., Mattoso, M., (2010b), "An Adaptive Approach for *Workflow Activity* Execution in *Cloud*s". In: *International Workshop on Challenges in e-Science - SBAC*, p. 9-16, Petrópolis, RJ - Brazil.

Oliveira, D., Ogasawara, E., Baiao, F., Mattoso, M., (2011b), "Adding Ontologies to Scientific *Workflow* Composition". In: *XXVI SBBD*, p. 1-8, Florianópolis, SC, Brazil.

Oliveira, D., Ogasawara, E., Baião, F., Mattoso, M., (2010c), "SciCumulus: A Lightweight *Cloud* Middleware to Explore Many Task Computing Paradigm in Scientific *Workflows*". In: *3rd International Conference on Cloud Computing*, p. 378–

385, Washington, DC, USA.

Oliveira, D., Ogasawara, E., Chirigati, F., Silva, V., Murta, L., Werner, C., Mattoso, M., (2009b), "Uma Abordagem Semântica para Linhas de Experimentos Científicos Usando Ontologias". In: *III e-Science*, p. 17-24, Fortaleza, Ceara, Brazil.

Oliveira, D., Ogasawara, E., Ocana, K., Baiao, F., Mattoso, M., (2011c), "An Adaptive Parallel Execution Strategy for *Cloud*-based Scientific *Workflows*", *Concurrency and Computation: Practice and Experience*, v. (online)

Oliveira, D., Ogasawara, E., Seabra, F., Silva, V., Murta, L., Mattoso, M., (2010d), "GExpLine: A Tool for Supporting Experiment Composition", *Provenance and Annotation of Data and Processes*, Springer Berlin / Heidelberg, p. 251-259.

Oliveira, R. da S. de, Carissimi, A., Toscani, S., (2010e), *Sistemas Operacionais (Vol. 11)*. 4 ed. Bookman.

OpenID, (2010), *OpenID Foundation website*, http://openid.net/.

OVF, (2010), *DMTF OVF Specification V1.1.0 published 2010-01-12*, http://www.dmtf.org/standards/published_documents/DSP0243_1.1.0.pdf.

Özsu, M. T., Valduriez, P., (2011), *Principles of Distributed Database Systems*. 3 ed. New York, Springer.

Patavino, G. M., de Almeida-Neto, C., Liu, J., Wright, D. J., Mendrone-Junior, A., Ferreira, M. I. L., de Freitas Carneiro, A. B., Custer, B., Ferreira, J. E., et al., (2012), "Number of recent sexual partners among blood donors in Brazil: associations with donor demographics, donation characteristics, and infectious disease markers", *Transfusion*, v. 52, n. 1, p. 151–159.

Paulino, C. E., Cruz, S. M. S., Oliveira, D., Campos, M. L. M., Mattoso, M., (2011), "Capturing Distributed Provenance Metadata from *Cloud*-Based Scientific *Workflows*", *Journal of Information and Data Management*, v. 2, n. 1, p. 43-50.

Paulino, C., Oliveira, D., Cruz, S. M. S., Campos, M. L. M., Mattoso, M., (2010), "Captura de Metadados de Proveniência para *Workflows* Científicos em Nuvens Computacionais". In: *Anais do XXV Simpósio Brasileiro de Banco de DadosXXV Simpósio Brasileiro de Banco de Dados*, Belo Horizonte, Minas Gerais, Brazil.

Paulino, C., Oliveira, D., Cruz, S. M. S., Mattoso, M., (2009), "Captura de Proveniência de *Workflows* Científicos Executados em Nuvem". In: *Proc. III Sessão de Pôsteres do XXIV SBBDIII Sessão de Pôsteres do XXIV SBBD*, Fortaleza, Ceara, Brazil.

Pereira, G. C., Ebecken, N. F. F., (2011), "Combining in situ flow cytometry and artificial neural networks for aquatic systems monitoring", *Expert Systems with Applications*, v. 38, n. 8, p. 9626 - 9632.

Pigliucci, M., Kaplan, J., (2006), *Making Sense of Evolution: The Conceptual Foundations of Evolutionary Biology*. University Of Chicago Press.

Porto, F., Moura, A. M., Silva, F. C., Bassini, A., Palazzi, D. C., Poltosi, M., Castro, L. E. V., Cameron, L. C., (2011), "A metaphoric trajectory data warehouse for Olympic athlete follow-up", *Concurrency and Computation: Practice and Experience*

Pressman, R., Roger, P., (2009), *Software Engineering: A Practitioner's Approach*. McGraw-Hill Higher Education.

ProvChallenge, (2009), *Third Provenance Challenge*,

http://twiki.ipaw.info/bin/view/Challenge/ThirdProvenanceChallenge.

ProvChallenge, (2010), *Provenance Challenge Wiki*, http://twiki.ipaw.info/bin/view/Challenge/WebHome.

Pruitt, K. D., Tatusova, T., Klimke, W., Maglott, D. R., (2009), "NCBI Reference Sequences: current status, policy and new initiatives", *Nucleic Acids Research*, v. 37, n. Database issue (jan.), p. D32-D36.

Python, (2010), *Python Programming Language – Official Website*, http://www.python.org/.

Qin, X., Hong, J., (2005), "A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters", *Journal of Parallel and Distributed Computing*, v. 65 (ago.), p. 885-900.

Raicu, I., Foster, I. T., Yong Zhao, (2008), "Many-task computing for grids and supercomputers". In: *Proceedings of the Workshop on Many-Task Computing on Grids and Supercomputers*, p. 1-11, Austin, Texas, USA.

Rochwerger, B., Galis, A., Levy, E., Caceres, J. A., Breitgand, D., Wolfsthal, Y., Llorente, I. M., Wusthoff, M., Montero, R. S., et al., (2009), "RESERVOIR: Management technologies and requirements for next generation Service Oriented Infrastructures". In: *IFIP/IEEE International Symposium on Integrated Network Management, 2009. IM '09IFIP/IEEE International Symposium on Integrated Network Management, 2009. IM '09*, p. 307-310

Russell, M. A., (2011), *21 Recipes for Mining Twitter*. 1 ed. O'Reilly Media.

Sabino, E. C., Gonçalez, T. T., Carneiro-Proietti, A. B., Sarr, M., Ferreira, J. E., Sampaio, D. A., Salles, N. A., Wright, D. J., Custer, B., et al., (2011), "Human immunodeficiency virus prevalence, incidence, and residual risk of transmission by transfusions at Retrovirus Epidemiology Donor Study-II blood centers in Brazil", *Transfusion*, p. no–no.

Samples, M. E., Daida, J. M., Byom, M., Pizzimenti, M., (2005), "Parameter sweeps for exploring GP parameters". In: *2005 workshops on Genetic and evolutionary computation*, p. 212-219, Washington, D.C., USA.

Sánchez, R., Serra, F., Tárraga, J., Medina, I., Carbonell, J., Pulido, L., de María, A., Capella-Gutíerrez, S., Huerta-Cepas, J., et al., (2011), "Phylemon 2.0: a suite of web-tools for molecular evolution, phylogenetics, phylogenomics and hypotheses testing", *Nucleic Acids Research*, v. 39 Suppl 2 (jul.), p. W470-474.

Sardiña, I., Boeres, C., Drummond, L., (2009), "Escalonamento Bi-objetivo de Aplicações Paralelas em Recursos Heterogêneos". In: *Anais do XXVII Simpósio Brasileiro de Redes de ComputadoresXXVII Simpósio Brasileiro de Redes de Computadores*, p. 25-29, Recife/PE.

Segaran, T., Evans, C., Taylor, J., (2009), *Programming the Semantic Web*. 1 ed. O'Reilly Media.

Shafi, A., Carpenter, B., Baker, M., (2009), "Nested parallelism for multi-core HPC systems using Java", *Journal of Parallel and Distributed Computing*, v. 69, n. 6 (jun.), p. 532-545.

Shafi, A., Carpenter, B., Baker, M., Taboada, G. L., (2010). MPJExpress: A Implementation of MPI in Java. Disponível em: http://mpj-

express.org/docs/guides/windowsguide.pdf. Acesso em: 13 dez 2010.

Silva, V., Chirigati, F., Maia, K., Ogasawara, E., Oliveira, D., Braganholo, V., Murta, L., Mattoso, M., (2010), "SimiFlow: Uma Arquitetura para Agrupamento de *Workflows* por Similaridade". In: *IV e-Science*, p. 1-8, Belo Horizonte, Minas Gerais, Brazil.

Silva, V., Chirigati, F., Maia, K., Ogasawara, E., Oliveira, D., Braganholo, V., Murta, L., Mattoso, M., (2011a), "Similarity-based *Workflow* Clustering", *Journal of Computational Interdisciplinary Science*, v. 2, n. 1, p. 23-35.

Silva, V., Chirigati, F., Ogasawara, E., Dias, J., Oliveira, D., Porto, F., Valduriez, P., Mattoso, M., (2011b), "Uma avaliação da Distribuição de Atividades Estática e Dinâmica em Ambientes Paralelos usando o Hydra". In: *V e-Science*, p. 1-8, Natal, Rio Grande do Norte, Brazil.

Simmhan, Y., Ingen, C., Subramanian, G., Li, J., (2010), "Bridging the Gap between Desktop and the *Cloud* for eScience Applications". In: *3rd IEEE Conference of Cloud Computing3rd IEEE Conference of Cloud Computing*, p. 474-481, Miami.

Simpson, B., Toussi, F., (2002). Hsqldb User Guide. Disponível em: http://www-lia.deis.unibo.it/Courses/TecnologieWeb0708/materiale/laboratorio/guide/hsqldb/guide.html. Acesso em: 8 set 2011.

Smanchat, S., Indrawan, M., Ling, S., Enticott, C., Abramson, D., (2009), "Scheduling multiple parameter sweep *workflow* instances on the grid". In: *e-Science 2009 - 5th IEEE International Conference on e-Science*, p. 300-306

Soanes, C., Stevenson, A., (2003), *Oxford Dictionary of English*. 2nd Revised edition ed. Oxford University Press.

Sonmez, O., Yigitbasi, N., Abrishami, S., Iosup, A., Epema, D., (2010), "Performance Analysis of Dynamic *Workflow* Scheduling in Multicluster Grids". In: *10th IEEE International Symposium on High Performance Distributed Computing*, p. 49-60, Chicago, Illinois, USA.

Sorde, S. W., Aggarwal, S. K., Song, J., Koh, M., See, S., (2007), "Modeling and Verifying Non-DAG *Workflows* for Computational Grids". In: *Services, IEEE Congress on*, p. 237-243, Los Alamitos, CA, USA.

Stamatakis, A., (2006), "RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models", *Bioinformatics (Oxford, England)*, v. 22, n. 21 (nov.), p. 2688-2690.

Staples, G., (2006), "TORQUE resource manager". In: *ACM/IEEE conference on Supercomputing*, p. 8, Tampa, Florida, USA.

Sub*Cloud*, (2011), *Shared Enterprise File System for Amazon S3 Cloud Storage | SubCloud*, http://www.sub*cloud*.com/.

Sullivan, F., (2009), "Guest Editor's Introduction: *Cloud* Computing for the Sciences", *Computing in Science and Engineering*, v. 11, n. 4, p. 10-11.

Sun, (2010), *Java*, http://java.sun.com/.

Taylor, I. J., Deelman, E., Gannon, D. B., Shields, M., (2007a), *Workflows for e-Science: Scientific Workflows for Grids*. 1 ed. Springer.

Taylor, I., Shields, M., Wang, I., Harrison, A., (2007b), "The Triana *Workflow* Environment: Architecture and Applications", *Workflows for e-Science*, Springer, p.

320-339.

Taylor, I., Shields, M., Wang, I., Harrison, A., (2007c), "The Triana *Workflow* Environment: Architecture and Applications", In: Taylor, I. J., Deelman, E., Gannon, D. B., Shields, M. [orgs.] (eds), *Workflows for e-Science*, London: Springer, p. 320-339.

Thain, D., Tannenbaum, T., Livny, M., (2002), "Condor and the Grid", *Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons Inc*

Thompson, J. D., Higgins, D. G., Gibson, T. J., (1994), "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice", *Nucleic Acids Research*, v. 22, n. 22 (nov.), p. 4673-4680.

Topcuoglu, H., Hariri, S., Min-You Wu, (2002), "Performance-effective and low-complexity task scheduling for heterogeneous computing", *IEEE Transactions on Parallel and Distributed Systems*, v. 13, n. 3 (mar.), p. 260-274.

Travassos, G. H., Barros, M. O., (2003), "Contributions of In Virtuo and In Silico Experiments for the Future of Empirical Studies in Software Engineering". In: *2nd Workshop on Empirical Software Engineering the Future of Empirical Studies in Software Engineering*, p. 117-130, Rome, Italy.

Twitter4j, (2011). Twitter4J - A Java library for the Twitter API. Disponível em: http://twitter4j.org/en/index.html. Acesso em: 22 set 2011.

UK eScience, (2011), *National e-Science Centre*, UK programme for eScience.

Valduriez, P., Pacitti, E., (2005), "Data Management in Large-Scale P2P Systems", *High Performance Computing for Computational Science - VECPAR 2004*, , p. 104-118.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., Lindner, M., (2009), "A break in the *cloud*s: towards a *cloud* definition", *SIGCOMM Comput. Commun. Rev.*, v. 39, n. 1, p. 50-55.

Vega, D., Oliveira, D., Ogasawara, E., Lima, A. A. B., Mattoso, M., (2010), "Modeling Parallel Bioinformatics *Workflows* Using MapReduce". In: *International Workshop on Challenges in e-Science - SBAC*, p. 71-72, Petrópolis, RJ - Brazil.

Viana, V., de Oliveira, D., Mattoso, M., (2011a), "Towards a Cost Model for Scheduling Scientific *Workflows Activities* in *Cloud* Environments". In: *2011 IEEE World Congress on Services (SERVICES)2011 IEEE World Congress on Services (SERVICES)*, p. 216-219

Viana, V., Oliveira, D., Dias, J., Ogasawara, E., Mattoso, M., (2011b), "SciCumulus-ECM: Um Serviço de Custos para a Execução de *Workflows* Científicos em Nuvens". In: *XXVI SBBDSBBD*, Florianópolis, SC, Brasil.

Vöckler, J.-S., Juve, G., Deelman, E., Rynge, M., Berriman, B., (2011), "Experiences using *cloud* computing for a scientific *workflow* application". In: *Proceedings of the 2nd international workshop on Scientific cloud computing*, p. 15–24, New York, NY, USA.

Walker, E., Guiang, C., (2007), "Challenges in executing large parameter sweep studies across widely distributed computing environments". In: *Workshop on Challenges of large applications in distributed environments*, p. 11-18, Monterey, California, USA.

Wall, L., Christiansen, T., Orwant, J., (2000), *Programming Perl*. 3rd ed. O'Reilly Media.

Wallis, J. C., Mayernik, M. S., Borgman, C. L., Pepe, A., (2010), "Digital libraries for scientific data discovery and reuse: from vision to practical reality". In: *Proceedings of the 10th annual joint conference on Digital libraries*, p. 333–340, New York, NY, USA.

Wang, J., Crawl, D., Altintas, I., (2009), "Kepler + Hadoop: a general architecture facilitating data-intensive applications in scientific *workflow* systems". In: *4th Workshop on Workflows in Support of Large-Scale Science*, p. 1-8, Portland, Oregon.

Wang, L., Tao, J., Kunze, M., Castellanos, A. C., Kramer, D., Karl, W., (2008), "Scientific *Cloud* Computing: Early Definition and Experience". In: *10th IEEE HPCC*, p. 825-830, Los Alamitos, CA, USA.

Warneke, D., Kao, O., (2009), "Nephele: efficient parallel data processing in the *cloud*". In: *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, p. 8:1–8:10, New York, NY, USA.

WfMC, I., (2009), *Binding, WfMC Standards*, WFMC-TC-1023, http://www. wfmc. org, 2000.

Wieczorek, M., Prodan, R., Fahringer, T., (2005), "Scheduling of scientific *workflows* in the ASKALON grid environment", *SIGMOD Rec.*, v. 34, n. 3, p. 56-62.

Wikipedia, F., (2011), *Linguagens de programação: Projeto Harbour, Prolog, Sistema especialista, COBOL, Clipper, Ruby, História das Linguagens de Programação*. Books LLC, Wiki Series.

Xin Lu, Zilong Gu, (2011), "A load-adapative *cloud* resource scheduling model based on ant colony algorithm". In: *2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, p. 296-300

XMPP, (2010), *XMPP*, http://xmpp.org/.

Yang, Z., (1994), "Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods", *Journal of Molecular Evolution*, v. 39, n. 3 (set.), p. 306-314.

Youseff, L., Butrico, M., Da Silva, D., (2008), "Toward a Unified Ontology of *Cloud* Computing". In: *Grid Computing Environments Workshop, 2008. GCE '08*, p. 10, 1

Yu, H., Vahdat, A., (2006), "The costs and limits of availability for replicated services", *ACM Trans. Comput. Syst.*, v. 24, n. 1, p. 70-113.

Yu, J., Buyya, R., (2006), "Scheduling scientific *workflow* applications with deadline and budget constraints using genetic algorithms", *Sci. Program.*, v. 14, n. 3,4, p. 217-230.

Yu, J., Buyya, R., Tham, C. K., (2005), "Cost-Based Scheduling of Scientific *Workflow* Application on Utility Grids". In: *International Conference on e-Science and Grid Computing*, p. 140-147, Melbourne, Victoria, Australia.

Zhang, H., Jiang, G., Yoshihira, K., Chen, H., Saxena, A., (2009), "Intelligent Workload Factoring for a Hybrid *Cloud* Computing Model". In: *Proceedings of the 2009 Congress on Services - I*, p. 701-708

Zhao, X., Jamali, N., (2011), "Supporting Deadline Constrained Distributed Computations on Grids". In: *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, p. 165–172, Washington, DC, USA.

Zhao, Y., Hategan, M., Clifford, B., Foster, I., von Laszewski, G., Nefedova, V., Raicu, I., Stef-Praun, T., Wilde, M., (2007), "Swift: Fast, Reliable, Loosely Coupled Parallel Computation". In: *3rd IEEE World Congress on Services*, p. 206, 199, Salt Lake City, USA.

Zhu, J., Wang, W., (2008), "New-Knowledge-View Based Ontology *Cloud* Model". In: *Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 05*, p. 1140-1143

Zvelebil, M., Baum, J., (2007), *Understanding Bioinformatics*. 1 ed. Garland Science.