HEURÍSTICAS PARA CONTROLE DE RE-EXECUÇÃO PARALELA DE *WORKFLOWS* CIENTÍFICOS EM NUVENS DE COMPUTADORES

Flavio da Silva Costa

Dissertação de Mestrado 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 Mestre em Engenharia de Sistemas e Computação.

Orientadora: Marta Lima de Queirós Mattoso

Rio de Janeiro

Setembro de 2012

HEURÍSTICAS PARA CONTROLE DE RE-EXECUÇÃO PARALELA DE *WORKFLOWS* CIENTÍFICOS EM NUVENS DE COMPUTADORES

Flavio da Silva Costa

DISSERTAÇÃO 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 MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Profª. Marta Lima de Queirós Mattoso, D.Sc.

Profª. Flávia Coimbra Delicato, D.Sc.

Prof. Geraldo Bonorino Xexéo, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

SETEMBRO DE 2012

Costa, Flavio da Silva

Heurísticas para Controle de Re-execução Paralela de *Workflows* Científicos em Nuvens de Computadores / Flavio da Silva Costa. – Rio de Janeiro: UFRJ/COPPE, 2012.

XII, 77 p.: il.; 29,7 cm.

Orientadora: Marta Lima de Queirós Mattoso

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2012.

Referências Bibliográficas: p. 58-65.

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.

Aos meus pais pelo apoio incondicional, ao grande amigo Daniel Oliveira por ser a bússola desta empreitada e à minha esposa Camille, por nunca desistir de mim

Agradecimentos

Quando chegamos perto do fim de um ciclo, paramos para refletir o que tornou possível tal realização e uma das coisas que invariavelmente concluímos é que não somos ninguém sozinhos. Nesse momento, o mínimo que podemos fazer é agradecer a todos aqueles que de alguma forma contribuíram para a concretização de mais um sonho. No entanto, pesa-nos a certeza que seremos injustos, pois de certo, muitos nomes vão faltar. Sendo assim, deixo aqui meu obrigado aos que por ventura sejam omitidos em meus agradecimentos. No entanto, o erro do esquecimento seria ainda maior se não agradecesse, em especial, a algumas pessoas sem as quais nada disso seria possível. Sendo assim agradeço:

À minha esposa Camille, minha melhor amiga de todas as horas. Sempre pronta pra me dar força nos momentos de cansaço e até escrever algumas linhas de código para me ajudar. Nos últimos anos você foi a pessoa mais importante na minha vida e será assim pra sempre.

Aos meus pais Marco Antonio e Gilvanete que abriram mão de muito para tudo me oferecer. Ter nascido seu filho já fez de mim um vencedor, todo o resto foi lucro. Vocês são um exemplo na minha vida e vivem sempre comigo em meu coração. Nunca conseguirei retribuir à altura

À minha família pela torcida e companheirismo: Minha irmã Erica, minha sobrinha Clarisse, minhas afilhadas Fernanda e Maryana, meus sogros Terezinha e Luiz Antonio e aos irmãos de coração: Felipe Assis, Giselle Furtado, Jeane Quintiliano, Roberto Aldilei e Fernando Pereira. Aos meus padrinhos Inacira e Euller, por estarem sempre comigo em todas as minhas conquistas.

Aos amigos Luiz Antonio e Sergio Júnior pela grande ajuda nessa caminhada.

Aos meus avós: João, Maria e Odette (que de onde estiver ficará muito feliz por minha conquista).

À professora Marta Lima de Queirós Mattoso, pela oportunidade de fazer parte de sua equipe e pela orientação precisa, eficiente e presente, contribuindo em muito para a excelência desse trabalho. Seguirei sendo um grande admirador seu.

Ao professor Geraldo Bonorino Xexéo, a quem serei eternamente grato pela oportunidade de aqui estar. Hoje o agradeço defendendo minha dissertação.

Ao Eduardo Ogasawara por todo o apoio e amizade de hoje e de outros tempos. À Kary Ocaña por toda ajuda na questão dos experimentos dessa dissertação e por sempre se mostrar disposta e paciente para esclarecer as questões da bioinformática. Em especial ao Daniel Oliveira que não mediu esforços para me ajudar em todas as questões envolvidas neste projeto de mais de 2 anos.

À professora Flavia Delicato por ter aceito fazer parte desta banca.

Aos professores que em algum momento contribuíram com o meu trabalho: Alexandre Assis, Guilherme Horta Travassos e Jano Moreira de Souza.

À Patrícia Leal, Mara Prata, 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;

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

Por fim agradeço a Deus por mais uma importante vitória.

Agradeço.

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

HEURÍSTICAS PARA CONTROLE DE RE-EXECUÇÃO PARALELA DE *WORKFLOWS* CIENTÍFICOS EM NUVENS DE COMPUTADORES

Flavio da Silva Costa

Setembro/2012

Orientadora: Marta Lima de Queirós Mattoso

Programa: Engenharia de Sistemas e Computação

Gerenciar *workflows* científicos quando os mesmos estão associados a aplicações de alta complexidade é um desafio. A ciência atual se tornou fortemente baseada dos avanços de *hardware* e de novas soluções computacionais. Muitos destes *workflows* são formados por milhares de atividades e que em apenas uma execução do experimento, suas atividades podem ser processadas milhares de vezes, chegando a casos onde temos mais de 100.000 atividades sendo executadas, para que o resultado seja gerado. Neste contexto, a computação de alto desempenho é uma alternativa importante e em especial as nuvens computacionais. Nuvens oferecem alto poder de processamento a um custo flexível, o que o torna viável, e ainda possuem baixa complexidade na sua utilização. Quando os cientistas utilizam os recursos de uma nuvem de computadores, não estão mais limitados aos recursos de uma máquina específica. Passam a poder optar por uma configuração específica para cada *workflow* a ser executado. Devido ao seu caráter distribuído, a utilização de recursos computacionais na nuvem traz consigo algumas questões como erros causados por flutuações de desempenho de *hardware*, problemas de comunicação e concorrência. Além da gerência da execução, outro problema que vem à tona é como podemos garantir que, ao término da execução de um *workflow*, todas as atividades de fato tenham sido executadas sem erro. Esta dissertação apresenta soluções que visam a maximizar a probabilidade de sucesso da execução de um *workflow*, diminuindo assim o retrabalho do cientista. O foco foi maior em questões como custo e desempenho.

Abstract of Dissertion presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

HEURISTICS TO SUPPORT PARALLEL RE-EXECUTION OF SCIENTIFIC *WORKFLOWS* ON COMPUTING CLOUDS

Flavio da Silva Costa

September/2012

Advisor: Marta Lima de Queirós Mattoso

Department: Systems and Computer Engineering

Managing scientific *workflows* when associated with applications of high complexity, is a challenge. Nowadays science has become highly based on advances in computer *software* and *hardware* solutions. Many of these *workflows* are composed by thousands of activities. When executing an experiment, these *workflows* can be executed thousands of times, and there can be more than 100,000 activities being executed to generate the result. In this context, high-performance computing is an important alternative in particular cloud computing. Clouds provide high processing power at a flexible cost, which can match the scientist budget and present low complexity involved in its use. When scientists choose to use resources of a cloud environment, they are no longer limited to the resources of a particular machine. They become able to choose a specific configuration for each workflow to be executed. Due to its distributed nature, the use of computational resources in the cloud brings some issues such as errors caused by *hardware* performance fluctuation, problems of communication and resource competition. Another important problem to be treated is how can we ensure that by the end of a *workflow* execution, all the activities that should have been executed have really completed. This dissertation presents to solutions aimed at maximizing the probability of *workflow* completion, reducing scientists rework. We focus in some issues such as cost and performance.

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1 - Introdução

Workflows científicos podem ser definidos como uma abstração para modelar o fluxo de atividades e de dados em um experimento e ao longo dos últimos anos os mesmos se tornaram um padrão para a modelagem de experimentos científicos que são baseados em simulação computacional (Mattoso et al. 2010). Muitos experimentos em diferentes domínios de conhecimento são dependentes de alto poder de processamento. A utilização de *workflows* é uma realidade na comunidade científica (Taylor et al. 2007a) que obtém por meio dos mesmos o ferramental necessário para o desenvolvimento de pesquisas em diversas áreas. Muitos desses *workflows* são compostos por milhares de atividades que por sua vez podem ser decompostas em tarefas. Como podem ser executadas milhares de vezes, para que o cientista finalize seu experimento, podemos criar um cenário onde mais de 100.000 atividades tenham que ser executadas. Neste contexto, os *workflows* científicos podem se beneficiar de ambientes de processamento de alto desempenho (PAD). Além dos já consolidados supercomputadores, outras soluções vêm sendo usadas para suprir a crescente demanda por soluções computacionais mais potentes e escaláveis. Recentemente, as nuvens computacionais (Vaquero et al. 2009) surgiram como um ambiente alternativo e viável para muitos tipos de aplicações (Hey et al. 2009) à medida que oferecem *hardware* e *software* de forma virtualizada e sob demanda para os usuários. Nuvens podem ser consideradas uma nova abordagem para PAD (Jackson et al. 2010). Utilizando-se nuvens computacionais, os cientistas não precisam, necessariamente, adquirir ambientes de PAD caros, como *clusters*, já que podem instanciar o poder de processamento necessário criando um ambiente multiprocessado virtual (por exemplo, um *cluster* virtual), que é composto por várias máquinas virtuais (MV) que possibilitam a execução de seus experimento em paralelo. Além disso, cientistas gastam um tempo considerável analisando os dados gerados pelos experimentos em suas máquinas pessoais durante o qual o ambiente de PAD muitas vezes se torna ocioso. Assim, os cientistas podem se beneficiar muito do modelo de computação em nuvem por serem cobrados pela utilização dos recursos de PAD apenas pelo que foi efetivamente utilizado (modelo de pagamento pelo uso (Vaquero et al. 2009)).

Podemos destacar, a título de exemplo, os *workflows* dos experimentos de filogenética. Estes podem ser compostos por milhares de atividades e por características

que ficarão mais claras no decorrer desta dissertação, serão altamente beneficiados por ambientes de execução paralela.

Além da execução paralela, os cientistas também precisam se preocupar com a gerência dos dados de proveniência (Freire et al. 2008) produzidos pela execução do *workflow*. Como estamos abordando experimentos científicos que precisam ter dentre outras questões, sua reprodutibilidade garantida, a utilização da proveniência é questão fundamental. Por proveniência podemos entender todos os dados que são armazenados com o objetivo de descrever determinado experimento. Todos os dados de entrada, a configuração do ambiente, o tempo de execução de cada atividade, etc. Essa gerência dos dados de proveniência se torna ainda mais complexa quando os *workflows* precisam ser executados em ambientes de PAD paralelos e distribuídos, como nuvens de computadores. É fundamental que saibamos qual atividade gerou que dado e onde esse dado está sendo armazenado (por exemplo, em qual MV foi armazenado).

Existem algumas abordagens que focam no gerenciamento da execução paralela das atividades científicas em nuvens computacionais com suporte a coleta de proveniência (Cui Lin e Shiyong Lu 2011, Juve e Deelman 2010, Oliveira et al. 2011b). No entanto, baseado em nosso conhecimento, apenas o Chiron (Ogasawara et al. 2011) e o SciCumulus (Oliveira et al. 2011b) permitem a consulta aos dados de proveniência durante a execução do *workflow*. Este tipo de dado de proveniência é gerado durante a execução paralela do *workflow* e pode ajudar o cientista à medida que permite ao mesmo tomar ações avaliando os dados gerados pela aplicação, em tempo real. Por meio da análise dos dados de proveniência podemos re-executar uma atividade com problema tão logo verifiquemos a existência do mesmo, evitando assim que uma série de erros ocorram na cadeia de programas que fará uso do dado que deveria ter sido gerado. Sendo assim, o SciCumulus é a única máquina (do inglês *engine)* para a execução de *workflows* na nuvem que permite, em tempo de execução, a consulta estruturada à base de dados de proveniência.

Existem muitos benefícios na utilização dos dados de proveniência gerados em tempo de execução, tais como, possibilidade de escalonamento adaptativo de *workflows* (Oliveira et al. 2011b), possíveis estratégias de supervisão (do inglês *steering)* de *workflows* (Dias et al. 2011) e gerenciamento das falhas das atividades com disparo de re-execuções. Em nosso trabalho, focamos esforços na utilização dos dados de proveniência para a gerência da re-execução das atividades que apresentam falhas

durante a execução com a possibilidade de tratar diferentes tipos falhas, para assim, aumentar a probabilidade de sucesso do experimento. Experimentos científicos já realizados por meio de workflows (Oliveira et al. 2011a)(Ocaña et al. 2011a)(Ocaña et al. 2011b) mostram que esta estratégia é importante, especialmente em ambientes de nuvens computacionais onde o ambiente é volátil e apresenta-se em constante transformação. Nas nuvens, falhas em máquinas virtuais (MVs) não são uma exceção. Os provedores de serviços de nuvens constantemente aplicam correções e atualizações sem o envolvimento das pessoas que utilizam o serviço. Apesar de as empresas que fornecem o serviço garantirem que no caso de falhas de MVs estas são substituídas, estas não garantem a consistência das atividades que vinham sendo executadas nestas instâncias que apresentaram falhas. Juve e Deelman (2010) e Gil *et al.* (2007), em seus trabalhos sobre *workflows* e nuvens, colocam que é de alt*a priori*dade prover mecanismos de confiança para que os cientistas não precisem se preocupar com o grande número de falhas em potencial que podem ocorrer.

Para ilustrar a necessidade de se garantir a alta probabilidade de sucesso de *workflows* científicos executados em paralelo, vamos tomar como motivação um *workflow* científico da área de bioinformática. 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. 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. Experimentos de filogenética são exemplos de *workflows* que são dependentes de PAD e que são compostos por milhares de atividades a serem executadas em paralelo. O SciPhy (Ocaña et al. 2011b) é um exemplo de *workflow* de análise filogenética. Este workflow tem como objetivo processar um grande número de arquivos do formato multi-fasta (cada um contendo várias sequências biológicas de diferentes organismos) para obter árvores filogenéticas, ajudando especialistas a explorar a filogenia para determinar a vida evolucionária de genes ou genomas. O SciPhy foi concebido para ser executado em paralelo onde cada arquivo multi-fasta é processado independentemente. Tipicamente, uma execução do SciPhy, em paralelo, em nuvens, consumindo 250 arquivos multi-fasta, gera 1250 atividades paralelas e demanda aproximadamente 4 dias usando 16 núcleos virtuais. Principalmente por conta de flutuações no ambiente de nuvem, uma execução típica do SciPhy apresenta entre 2% e 9% de falha nas atividades, que aproximadamente demandam mais de 10 horas adicionais de processamento (Ocaña et al. 2011b).

Utilizando-se dados de proveniência em tempo de execução é possível identificar atividades que falharam tão logo as falhas aconteçam. Desta maneira, é possível disparar novas execuções de atividades de modo que todas as atividades do *workflow* (e suas tarefas paralelas associadas) sejam executadas. Com essa estratégia aumentamos a probabilidade de sucesso da execução do *workflow* sem deixar de levar em consideração o seu desempenho. Ao re-executarmos atividades que falham, tão logo o erro seja identificado, evitamos ter que identificar e re-executar todos os pontos de falha somente ao fim da execução paralela (tal como ocorre na maioria das abordagens utilizadas, até por que, os dados de proveniência, de um modo geral, só estão disponíveis ao término da execução do *workflow*), reduzindo assim o tempo que seria gasto com a alternativa mais utilizada. As abordagens que buscam maneiras de tornar a execução do *workflow* mais confiável quando processado em ambientes compartilhados ou com alta taxa de falha, não são baseados na leitura dos dados de proveniência em tempo real; tampouco exploram as nuvens computacionais e suas características próprias, tais como a elasticidade e a volatilidade. Considerando que a execução em nuvens computacionais requer a carga de imagens e diversas configurações, ao deixar para tratar do problema somente no fim da execução do *workflow*, o ambiente que fora configurado pode ter sido extinto e novas configurações podem se fazer necessário.

Nesta dissertação apresentamos o SciMultaneous, um sistema com arquitetura baseada em serviços que implementa um conjunto de heurísticas baseadas nos dados de proveniência gerados em tempo de execução e nas características próprias das nuvens, inspiradas por mecanismos de processamento de transação já consolidados em Sistemas de Gerência de Banco de Dados. O SciMultaneous tem por objetivo monitorar a execução em paralelo de atividades de um *workflow* científico e a tomar ações para contornar possíveis falhas no ambiente ou indisponibilidades.

1.1 Caracterização do Problema: Como Garantir a Confiabilidade da Execução em Paralelo de *Workflows* Científicos em Nuvens Computacionais

O ambiente de nuvem é um ambiente virtual onde recursos de *software* e *hardware* são compartilhados pelos usuários sob demanda, utilizando a *Web*. No entanto, esta estrutura virtual, para ser vista como uma nuvem deve fornecer poder de processamento escalável, alta disponibilidade e grande capacidade de armazenamento. Diferentemente dos *clusters* e supercomputadores (Dantas 2005a, 2005b), não existe uma estrutura virtual residente em uma nuvem pronta para ser explorada no contexto de *workflows* científicos. O que existem são MVs poderosas, que podem ser criadas à medida que sejam necessárias e que podem ser utilizadas para executar atividades dos *workflows*. Entretanto, uma MV apenas, pode não ser capaz de oferecer o poder de processamento necessário a muitos experimentos. Sendo assim, um ambiente com múltiplas MVs poderia executar essas atividades em paralelo (Iosup et al. 2011).

Entretanto, a tecnologia de nuvem ainda é incipiente no que tange à execução de *workflows* científicos (Antonopoulos e Gillam 2010), destacando o fraco controle de falhas na execução de suas atividades. Com o ambiente de nuvem, novas variáveis foram introduzidas, como problemas de armazenamento, transferência e segurança. Não é trivial saber se as abordagens de controle de execução existentes poderiam ser aplicadas/adaptadas para o ambiente de nuvem de forma transparente. Na nuvem podem ocorrer diversas falhas (de virtualização, de execução de programas, acesso ao provedor de nuvem) e estas falhas são inevitáveis, mas suas consequências devem ser minimizadas. É justamente este problema que essa dissertação se propõe a resolver: "Como aumentar a confiabilidade da execução do *workflow* por meio da re-execução

das atividades dos *workflows* sem impactar o desempenho do mesmo ou impactar minimamente?" Desta forma, essa dissertação tem como objetivo o estudo e o desenvolvimento de heurísticas para controle de falhas em execuções paralelas de *workflows* científicos em nuvens, aumentando assim sua probabilidade de sucesso. Mais do que isso, ao adotarmos a estratégia de monitoramento da execução do experimento por meio de seus dados de proveniência, não impactamos, ou impactamos minimamente a execução do *workflow* como um todo. Analisando-se dados de proveniência gerados ao longo da execução do *workflow*, somos capazes de tomar ações durante o experimento e chegar ao fim deste com maior probabilidade de sucesso, tendo em vista que alguns problemas apresentados ao longo de sua execução já podem ter sido resolvidos por nosso sistema de monitoramento e ação.

1.2 Hipótese: Heurísticas e Serviços para Re-execução Distribuída de *Workflows* Científicos em Nuvens

Tentar mapear todos os possíveis cenários de falha na execução de um *workflow* científico pode ser uma tarefa complexa e, em alguns casos, inviável. Diferentemente de *workflows* assíncronos, um sistema complexo de atividades encadeadas, ainda representa um grande desafio para a área de controle de falhas. Podemos levantar o maior número de pontos de atenção e trabalhar de forma exaustiva para que não ocorram falhas na hora da execução, ou caso ocorram, já se tenha previsto tal cenário e já exista uma ação a ser tomada (Ferreira et al. 2010). Esta abordagem é cara, pois envolve muito esforço humano e muitas vezes seu resultado não é o esperado, pois podem existir muitos cenários possíveis e únicos, o que torna muito difícil tal previsão. O que buscamos é oferecer mecanismos que auxiliem na execução de atividades do *workflow*, baseando-se em heurísticas bem definidas, e que sejam capazes de resolver determinadas categorias de problemas (falha na conexão com o banco de dados, falha na obtenção dos arquivos de entrada, ou ausência do arquivo de saída esperado, ou ainda execuções que param sem um motivo aparente) que aconteçam ao longo de sua execução. Essas heurísticas são implementadas no SciMultaneous, uma arquitetura de serviço que ao ser acoplada a um sistema de gerência de *workflow* científico (SGWfC), trabalha no sentido de aumentar a taxa de sucesso das execuções de um *workflow* científico por ele gerenciadas. SciMultaneous se baseia na consulta a dados de proveniência gerados ao longo da execução de um *workflow* científico. Em nosso

trabalho utilizamos a base de proveniência do SciCumulus, mas é possível utilizar o SciMultaneous acoplado a outros SGWfC, desde que o SGWfC permita consulta aos dados de proveniência em tempo de execução.

Baseando-se na análise dos dados de proveniência gerados ao longo da execução do *workflow*, o SciMultaneous adota duas heurísticas básicas para aumentar a taxa de sucesso do experimento. A primeira, H1, trata-se da execução com redundância de tarefas consideradas fundamentais, estando assim mais preparado para uma recuperação no caso de falha. Os dados de proveniência são utilizados para a classificação das tarefas, conforme veremos adiante. A segunda heurística, H2, trata do monitoramento contínuo das tarefas. No caso de falha, podemos direcioná-las para outra MV do mesmo provedor de nuvem ou até mesmo para outra nuvem. Esse redirecionamento não é trivial, pois temos que buscar uma MV que já possua os programas que a tarefa irá invocar, previamente instalados.

1.3 Organização da Dissertação

Nesta dissertação analisamos as questões envolvidas na execução de *workflows* científicos na nuvem. Especificamente tratamos o problema de falhas na execução de um ciclo completo de experimento em virtude de falhas pontuais decorrentes de flutuações de ambiente dentre outros problemas abordados em detalhe ao longo deste trabalho. Com a utilização das estratégias implementadas obtivemos ganhos expressivos, da ordem de 10% em termos de ganho de desempenho. Levando-se em conta que estamos falando de experimentos que rodam por dias, 10% pode representar ganhos de mais de um dia de processamento. No contexto desse trabalho, além de melhorar o desempenho das aplicações, ganhamos também em confiabilidade. O texto foi organizado da seguinte forma: No Capítulo 2 discorremos sobre a fundamentação teórica de experimentos científicos em larga escala. Já no Capítulo 3, apresentamos a ferramenta SciCumulus que foi utilizada como base para alguns testes da ferramenta aqui apresentada. Nos Capítulo 4 e Capítulo 5 discorremos sobre o SciMultaneous, uma ferramenta que teve seu desenvolvimento baseado num conjunto de heurísticas que visam a aumentar a probabilidade de sucesso dos experimentos científicos que dependem de grande poder de processamento. Apresentamos ainda alguns cenários baseados em experimentos científicos reais onde a ferramenta mostrou que contribui para o aumento das chances de sucesso na execução de um *workflow* científico. No

Capítulo 6 concluímos nossa dissertação aproveitando para apresentar sugestões de pesquisas futuras.

Capítulo 2 - Fundamentação Teórica de Experimentos Científicos em Larga Escala

Nas próximas seções abordamos as questões envolvidas em experimentos científicos computacionalmente dependentes de PAD. Segmentamos o texto, dividindo-o da seguinte maneira: A Seção 2.1 apresenta o conceito de experimento científico baseado em simulação, que é o objeto de estudo desta dissertação, enquanto que a Seção 2.2 define o ciclo de vida de um experimento científico. 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 finalmente a Seção **Erro! Fonte de referência não ncontrada.** aborda as questões dos ambientes de processamento de alto desempenho que podem ser utilizados para processamento distribuído de *workflows* científicos.

2.1 Experimentos Científicos Baseados em Simulação

Nos últimos anos, a utilização de sistemas de *workflows* para a execução de experimentos científicos baseados em simulações computacionais vem aumentando consideravelmente (Mattoso et al. 2010) e os experimentos em larga escala requerem 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, o máximo possível, os erros ocasionais durante uma observação planejada (Juristo e Moreno 2001). Sendo assim, podemos concluir que um experimento científico tenta seguir um conjunto de ações controladas, previamente planejadas. Estas 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. 2010).

Nesta dissertação estamos interessados na categoria de experimentos científicos baseados em simulação (Travassos e Barros 2003), ou simplificadamente, experimentos científicos. 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. 2009), 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.

Para que seja viável o desenvolvimento destes tipos de experimentos, todos considerados de larga escala, um ambiente de PAD é necessário, pois para sua resolução muitas horas de processamento serão necessárias, em ambientes 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 devido 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 muitos casos, os experimentos científicos são formados por um conjunto de programas que devem ser executados de forma encadeada, produzindo e consumindo

uma grande quantidade de dados. Um complicador é adicionado a esse contexto que consiste no fato de os ambientes computacionais de processamento poderem ser 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).

Workflows científicos apresentam uma estratégia mais interessante para controlar o encadeamento 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. 2004), o Taverna (Hull et al. 2006), 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 uma com suas próprias características, vantagens e desvantagens.

Nessa dissertação, 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, o *workflow* pode ser definido 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 apenas 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

A seguir apresentamos um dos modelos existentes para representar o ciclo de vida de um experimento científico (Mattoso et al. 2010). A Figura 1 apresenta o ciclo de vida de

um experimento cientifico em larga escala. Como podemos observar na imagem, múltiplas etapas são percorridas pelo cientista várias vezes no curso do experimento, de acordo com as seguintes fases: a execução, a composição e a análise. Cada fase possui um subciclo independente, que é percorrido em momentos distintos do curso do experimento e manipula diferentes tipos de descritores de proveniência.

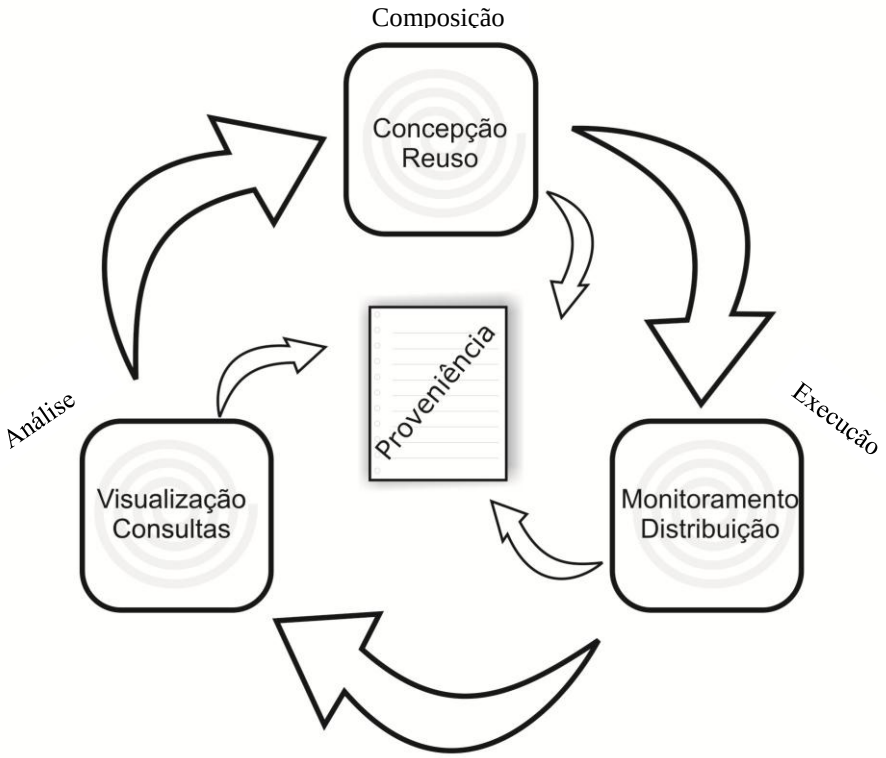

Figura 1 Ciclo de vida do experimento científico adaptado de Mattoso *et al.* (2010)

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. 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.

No entanto, é na fase de execução que concentramos nossos esforços, pois é a fase com maior dependência computacional. Esta é a etapa responsável por tornar concreta e executável uma especificação de *workflow* de forma que possa ser executada por um SGWfC. 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: 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 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. 2008) que foram gerados nas fases anteriores (a definição de proveniência será mais bem explicada a seguir nesta dissertação). 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) o resultado é provável que seja correto ou (ii) baseado nos resultados, a hipótese é refutada.

Em ambos os casos os cientistas provavelmente terão que executar novamente o *workflow* para efetivamente validar a hipótese ou criar uma nova. 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

Antes de definirmos *workflows* científicos vejamos 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 dissertação 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).

Com o avanço das técnicas temos visto a utilização do *workflow* em outros ramos da ciência, como a biologia e a astronomia inicialmente. 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).

2.4 Proveniência de Dados

Quando pensamos no termo proveniência, em geral associamos a palavra procedência, origem. De todas as definições existentes, a que nos interessa no contexto desta dissertação é realmente 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 permitem a citação e atribuem responsabilidades em caso de dados errados e; (iv) informacional: permitem realizar consultas baseadas nos descritores de origem para a descoberta de dados, além de prover o contexto necessário para interpretar os mesmos.

Historicamente podemos dividir o registro da proveniência em dois grupos, a prospectiva e a retrospectiva (Freire et al. 2008). A proveniência prospectiva está interessada em capturar e armazenar os dados relativos à estrutura do processo que levou a 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) criaram uma classificação interessante a do "grão fino" (do inglês *fine grain*) e do "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 dissertação, abordaremos os dois tipos de proveniência, armazenando informações de arquivos produzidos (conjunto de dados) bem como sobre os dados (itens) extraídos destes arquivos.

Para termos dados de proveniência estruturados segundo modelos já bem discutidos e internacionalmente difundidos devemos seguir, 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 propõe uma representação genérica de proveniência. O OPM e 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. Este ponto é fundamental para tornar nossa proposta, o SciMultaneous, uma ferramenta desacoplada da solução de

SGWfC. Por esse motivo, principalmente o OPM já vem sendo utilizado por diversos SGWfC (Davidson e Freire 2008, Freire et al. 2008) 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 Nuvem Computacional

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. A nuvem passou a apresentar uma nova possibilidade em relação aos já consolidados ambientes *desktop*. Os usuários são capazes de acessar programas, documentos e dados de qualquer dispositivo capaz de se conectar a internet. Mesmo aqueles de hardware reduzido como celulares. Conceitos de elasticidade, disponibilidade e segurança estão diretamente relacionados a este ambiente e novos desafios surgem desse novo cenário. A computação em nuvem apresenta um grande potencial para apoiar experimentos científicos que necessitem de ambientes de PAD. A natureza das necessidades da computação científica se encaixa bem com a flexibilidade e a elasticidade, sob demanda, oferecida pelas nuvens, onde 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*.

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, apresentaremos o ciclo de vida do *workflow* científico quando executado em nuvens computacionais.

2.6 Ciclo de Vida do *Workflow* Científico em Nuvem

A expressão ciclo de vida de um *workflow* científico na nuvem nos remete a uma sequência de passos realizados durante uma simulação e segue as mesmas etapas do ciclo de vida de um experimento de caráter geral, 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 2.

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*.

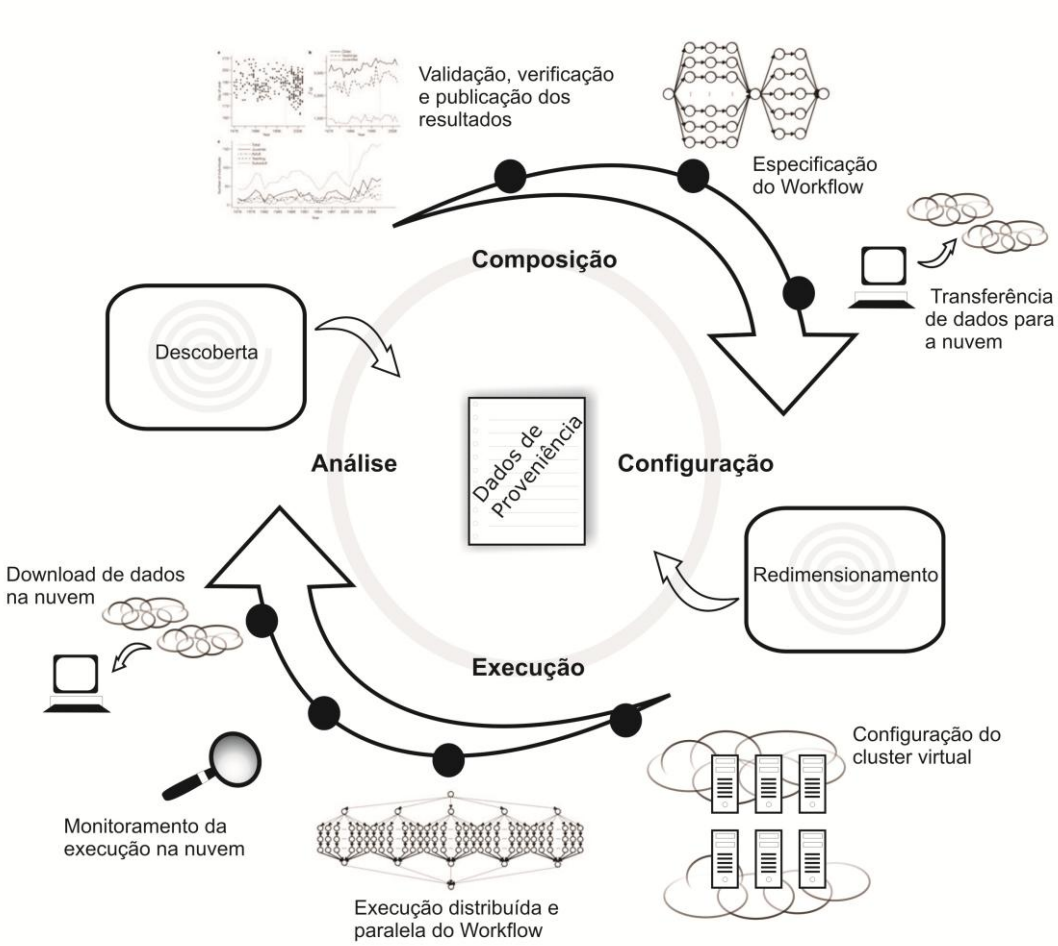

Figura 2 Ciclo de vida de um *workflow* científico executado em nuvens de computadores adaptado de Oliveira(2012)

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 dissertação, apontando em que etapa do ciclo as abordagens propostas se encaixam.

2.7 Trabalhos Relacionados

A preocupação em evitar o retrabalho devido a falhas de processamento em *workflows* científicos é comum, seja em ambientes de supercomputadores, grids ou nuvens. Sendo assim é possível traçar um paralelo entre o que propomos para nuvem e o que vem sendo discutido nestas outras abordagens.

O trabalho de Ferreira et al. (2010) busca uma representação capaz de apresentar a qualquer momento o estado de um *workflow*. Para isso utiliza o conceito de tuplas formadas por um estado atual, o estado anterior ao atual, as N condições que tornaram possível tal avanço e as ações que foram tomadas para efetivar essa transição. Considerando que a todo tempo podemos saber o estado de um *workflow*, o autor compara uma passagem de uma tupla *t* para uma tupla *t'* com uma transação de banco de dados. Dessa forma, sempre que algum problema ocorrer na passagem de *t* para *t'*, será possível restabelecer o último estado consistente do *workflow*, ou seja, restaurar *t*. No entanto, para *workflows* muito complexos, pode ser uma tarefa difícil, e até mesmo inviável, controlar a criação dessas tuplas. Outra questão interessante levantada no trabalho de Sindrilaru et al. (2010) é a maneira como determinamos se uma dada atividade do *workflow* foi executada com sucesso ou não. Em determinadas situações é possível estabelecer heurísticas que nos indiquem que uma tarefa pode ser vista como concluída, mesmo tendo apresentado algum tipo de falha. Outra estratégia encontrada na literatura é aquela baseada nos facilitadores já existentes em banco de dados. Para experimentos fortemente baseados em banco de dados é possível manter uma réplica de tudo o que está sendo alterado. A qualquer tempo, é possível restaurar o banco de dados, ou por meio da execução de *logs* ou mesmo por meio de uma cópia mantida *on-line*. De maneira mais geral, a restauração de um *workflow* que tenha sofrido uma falha será mais simples se ao longo de sua execução temos os dados utilizados em disco. Podemos encontrar esta abordagem no trabalho de Gustavo Alonso et al. (2000). No trabalho de Lee e Zomaya (2010) existe uma estratégia de re-agendamento na nuvem para se garantir a conclusão de um conjunto de atividades dentro de um prazo estimado. Antes do início da execução das atividades em uma grade computacional, é feita uma previsão do tempo que cada uma consumirá para ser finalizada. Se durante a execução das atividades constata-se que está se gastando mais tempo que o esperado (em virtude, por exemplo, da flutuação dos recursos das máquinas) alguma ou algumas atividades podem ser re-agendadas para a nuvem no intuito de garantir que todo o conjunto de

tarefas a ser processado termine dentro do tempo esperado. O trabalho deles foca na nuvem como um auxílio à solução principal baseada em grade computacional. Em alguns sistemas de gerência de execução de *workflows* já existem alternativas nativas que demandam a escrita de scripts (não triviais) pra configurar ações para alguns tipos de falhas como as apontadas por Zhao et al. (2007). Para atacar os problemas mais comuns existentes, alguns autores como Yang Zhang et al. (2009) e Gopi Kandaswamy et al. (2008) descrevem a existência de dois tipos de estratégias básicas, isto é, a que vai replicando a execução de determinadas atividades em mais de um nó e a que vai verificando até que ponto o *workflow* foi executado com sucesso. Para cada uma dessas abordagens, determinadas análise e decisões são feitas em caso de falhas na execução.

No trabalho de (Crawl e Altintas 2008) é proposto o *Scientific Workflow Doctor* (SWD), um componente para o Kepler (Altintas et al. 2004) que utiliza os dados de proveniência prospectiva com o intuito de garantir tolerância a falhas. SWDs são controladores independentes (chamados de atores) postos em determinados pontos da estrutura do *workflow* e estão associados a condições pré-definidas pelo usuário (critério de falha). Um ator é um sub-*workflow* que contém um fluxo de atividades primário e um suplente. Quando este ator falhar (ou seja, um determinado valor não está de acordo com uma condição pré-definida), SWD invoca o *sub-workflow* alternativo, que contém atividades alternativas e recursos para serem consumidos pelos programas. No entanto, SWD está focado na execução de *workflows* sequenciais de forma diferente da abordagem proposta nesta dissertação, que é capaz de tratar vários cenários, ao mesmo tempo, aumentando claramente a complexidade. Além disso, os dados de proveniência utilizados pelo SWD não estão disponíveis para consulta até que *workflow* finalize sua execução.

Como as nuvens estão mudando constantemente, não é trivial que as abordagens apresentadas nesta seção possam ser aplicadas / adaptadas para a execução paralela em ambiente de nuvem. Na nuvem, vários tipos de flutuações podem ocorrer e falhas são inevitáveis, mas suas consequências devem ser minimizadas. É por isso que focamos, neste trabalho, no desenvolvimento de heurísticas baseadas em nuvem para controlar as falhas e re-execuções em paralelo no próprio ambiente de nuvens.

Por meio de nossa pesquisa, não localizamos soluções que abordassem o problema da maneira que estamos propondo. As soluções apresentadas não são capazes de atuar de forma automática durante a execução do *workflow* científico, para o

aumento da confiabilidade de sua execução. Não utilizam características próprias da nuvem pra contornar as falhas de execução, tentando corrigi-las tão logo ocorram diminuindo a chance da propagação de erros.

Como a execução de *workflows* na nuvem com coleta de dados de proveniência já é uma realidade (Oliveira et al. 2011b) nossa aplicação vem suprir uma lacuna para aplicações baseadas neste ambiente. Explorando características próprias da nuvem, buscamos nessa dissertação prover mecanismos que melhorem a confiabilidade em sua execução, preocupando-se também, como já foi dito, com questões importantes como o custo, o desempenho, formas mais eficientes de armazenar e recuperar informação e o grau de qualidade da pesquisa acadêmica.

Capítulo 3 - SciCumulus

3.1 Arquitetura

O SciCumulus foi projetado para proporcionar aos SGWfC a gerência da execução paralela do *workflow* utilizando a infraestrutura de nuvem. A ideia principal do SciCumulus é, para cada atividade do *workflow*, gerar uma série de tarefas que podem ser executadas em paralelo (uma tarefa executando em cada MV). Soluções como o SciCumulus devem ser capazes de identificar e controlar falhas apoiando execuções confiáveis na presença de concorrência e erros. O SciCumulus orquestra a execução das *tarefas* do *workflow* científico em um conjunto distribuído de máquinas virtuais (*cluster* virtual), oferecendo dimensionamento dos recursos durante o curso de execução do *workflow*. O SciCumulus (Oliveira et al. 2010a, 2010b, 2011b, 2011a) foi projetado para seguir uma arquitetura de quatro camadas sendo a primeira delas a Camada Cliente. Seus componentes são instalados nas máquinas dos cientistas. Os componentes desta camada 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). A segunda camada é a de distribuição. Esta cria e gerencia a execução paralela de *tarefas* em uma ou mais máquinas virtuais instanciadas em um ambiente de nuvem. Temos também a camada de execução: seus componentes são instalados nas várias máquinas virtuais envolvidas na execução paralela das *tarefas*. É responsável pela execução de programas encapsulados em tarefas e por coletar e armazenar dados de proveniência. Finalmente temos a quarta camada, a camada de dados, 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 que monitora o experimento.

Espera-se com esta aplicação isolar o cientista das complexidades envolvidas no gerenciamento de execuções de atividades em paralelo. Sua configuração é trivial, e uma vez feita, o cientista terá em suas mãos um poderoso ambiente para executar seus experimentos. De maneira simples é possível ainda ter acesso à proveniência dos dados de execução, muito importante para o trabalho do cientista. SciCumulus dá suporte a

dois tipos de paralelismo mas no contexto deste trabalho exploramos apenas um, o paralelismo de varredura de parâmetros.

O controle desse tipo de paralelismo é difícil quando a gerência é feita de forma ad hoc em ambientes de nuvem devido à grande quantidade de tarefas 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. A arquitetura do SciCumulus é simples e 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 3 apresenta a arquitetura conceitual do SciCumulus em seus quatro níveis principais.

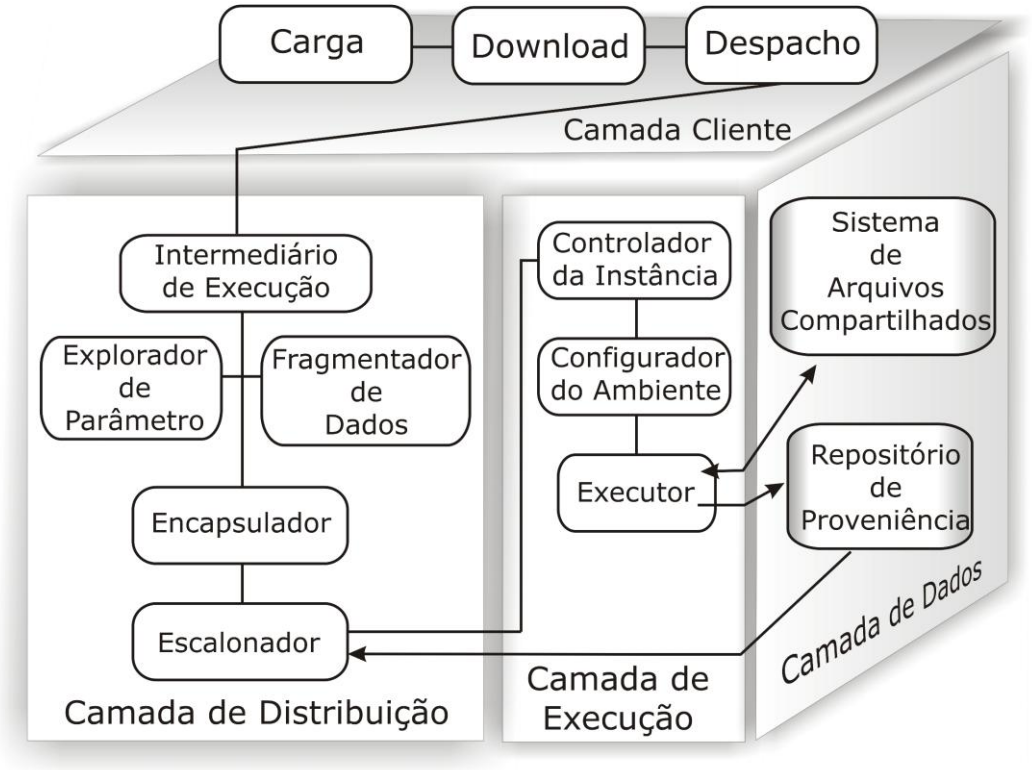

Figura 3 Arquitetura Conceitual do SciCumulus adaptada de **(**Oliveira **2012)**

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 ou o Taverna. Os

componentes desta camada se situam 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 a 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 é responsável por mover os dados para a nuvem, enquanto que o componente de *download* fica responsável por recolher das máquinas virtuais os dados gerados e colocá-los na máquina do cientista, ou seja, esses componentes são responsáveis pela preparação de dados de entrada e sua carga e pela transferência dos dados de saída. Aqui cabe uma reflexão em relação ao consumo de banda de transferência de dados. Os componentes de carga e de *download* são componentes que devem ser preferencialmente invocados quando o experimento científico lida com dados de entrada de tamanho de pequeno a médio e produz uma saída de tamanho de pequeno a médio para ser transferida também. Isto se dá uma vez que a transferência de dados para o ambiente de nuvem ocorre sobre uma conexão comum com a internet. Se os cientistas pretendem executar uma varredura de parâmetros usando um arquivo de 100 MB como entrada para os programas do workflow, 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 100 GB de dados de entrada, por exemplo. Desta forma, os cientistas podem transferir dados de suas estações de trabalho para e da nuvem *a priori*. Na verdade, 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. O componente de despacho cria um arquivo de configuração (que pode ser representado como um arquivo XML) que armazena a configuração para a execução distribuída (*i.e.* ele informa os parâmetros que devem ser explorados, por exemplo) e inicia o processo de execução das *tarefas* na nuvem se comunicando diretamente com os componentes da camada de distribuição. Toda a informação presente no XML em conjunto com os dados de proveniência gerados, nos fornecem informações importantes sobre o experimento.

Na camada de distribuição ocorre o controle da execução de atividades paralelas em nuvens por meio da criação e gerenciamento das *tarefas* 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 seis componentes: o intermediário de execução, o explorador de parâmetros, o fragmentador de dados, o encapsulador e o escalonador. Estes componentes se situam na camada de execução do ciclo de vida do *workflow* executado em nuvens. O intermediário de execução faz a ligação entre a camada cliente e a camada de distribuição. Ele inicia a execução paralela do *workflow* por meio da análise dos metadados fornecidos pelo componente de despacho na camada cliente. De acordo com a 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 tarefa seja composta de mais de uma atividade). Este componente produz as combinações dos valores dos parâmetros a serem consumidos por diversas *tarefas*.

O fragmentador de dados atua quando os dados de entrada devem ser fragmentados (e armazenados em um sistema de arquivos compartilhado, por exemplo), 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 *tarefas* a serem transferidas para as máquinas virtuais envolvidas na execução paralela.

O escalonador define quais máquinas virtuais receberão uma *tarefa* 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. O escalonador do SciCumulus 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 levar a problemas de desempenho. Por conta disto, este modo de trabalho do escalonador não é indicado.

Já 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. Baseado no escalonamento escolhido, o escalonador transfere as *tarefas* 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 conexão entre a camada de distribuição e a camada de execução. Ele é também responsável por controlar o fluxo de execução na instância quando a *tarefa* é associada a dois ou mais aplicativos para serem executados (um sub-*workflow*, por exemplo). O configurador do ambiente é responsável por montar toda a estrutura para a execução do *workflow*, definindo as variáveis de ambiente, e criando diretórios necessários para gerenciar a execução. Ele desencapsula a tarefa, 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 tarefa específica) para armazenar 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.

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 se situam na fase de análise do ciclo de vida do *workflow* executado em nuvem. O repositório de proveniência contém dados importantes de proveniência (Freire et al. 2008) 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 que monitora o ambiente à 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 tarefas durante a execução paralela.

Com as funcionalidades acima descritas temos um conjunto mínimo necessário para a execução paralela de *workflows* científicos em nuvens computacionais.

3.2 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 tem como base o *Open Provenance Model* (OPM) (Moreau et al. 2008a, 2008b) que visa facilitar a interoperabilidade de metadados de proveniência oriundos de ambientes heterogêneos. O OPM é uma recomendação aberta e focada no ponto de vista da 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. O OPM 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) e ao utilizá-lo, na verdade estamos seguindo um padrão de representação/armazenamento de dados já consolidado, o que dá credibilidade a nosso trabalho.

A Figura 4 apresenta o esquema conceitual projetado para o repositório de proveniência do SciCumulus. Todas as informações relacionadas a execuções anteriores de *workflows* científicos são recuperadas a partir do repositório de proveniência do SciCumulus. Este modelo é representado por meio de um diagrama de classes da *Unified Modeling Language* (UML) (Bezerra 2007, Fowler 2003) e foi modelado com base nos requisitos levantados com cientistas. Parte das informações presentes (aquelas relativas a tempos de execução e estado da execução) no modelo de proveniência é capturada pelos componentes do SciCumulus durante a execução do *workflow* enquanto que o agente capta as informações sobre o ambiente de nuvem. Este modelo de proveniência 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.

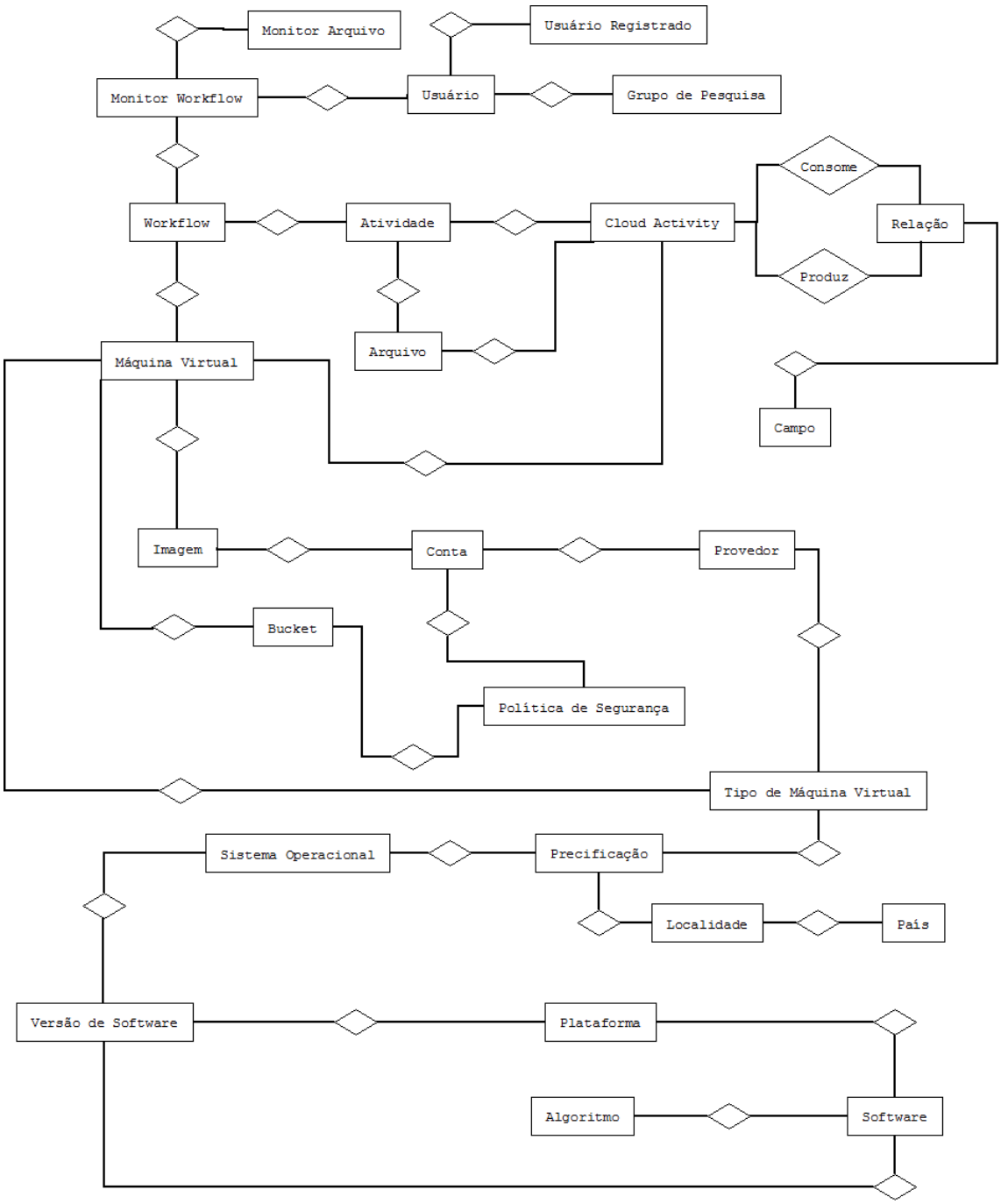

Figura 4 Modelo conceitual do repositório de proveniência do SciCumulus (Oliveira 2012)

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 atividades já existentes, 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 está sendo executado para fins de anotação.

A entidade *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 que descrevem em que estado cada atividade se encontra. 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 *tarefas* 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 *tarefas*. Os atributos *início* e *fim* contêm, respectivamente, a informação do tempo de início da execução da primeira *tarefa* desta atividade e a informação do tempo de término da execução da última *tarefa* desta atividade.

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

O SciCumulus está baseado em um modelo algébrico para representação de *workflows,* proposto por Ogasawara *et al.* (2011) e dessa forma, 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 *tarefa*. 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 *tarefa*, ou seja, *nome*, *data* e *tamanho*.

Detalharemos agora as classes que representam os dados coletados do ambiente de nuvem por meio do agente responsável por monitorar a execução do experimento. 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. Como em muitos casos estamos lidando com ambientes de nuvem públicos, que cobram pelo serviço, pode existir 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 cada *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. De acordo com o

experimento a ser executado, podem se utilizar máquinas virtuais diferentes e até no mesmo experimento é possível haver mais de um tipo de máquina virtual. Cada imagem possui um *ID*, um *nome* e um *tipo*. Os dados que são consumidos pelas *tarefas* 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. As classes *Atividade* e *CloudActivity (tarefa)* 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 de proveniência em tempo de execução para escalonamento e monitoramento. Da mesma maneira o SciMultaneous irá se beneficiar desta estratégia, como veremos no próximo capítulo. Nas abordagens existentes, tais como o VisTrails, os dados de proveniência são gerados somente no final da execução do *workflow*.

Capítulo 4 - SciMultaneous

Experimentos científicos de grande complexidade são dependentes de capacidade de processamento de larga escala. Para oferecer uma resposta eficiente frente às necessidades que se apresentam, estratégias como distribuição de tarefas com controle da execução e tolerância a falhas devem ser aplicadas ao ambiente de nuvens. Neste capítulo apresentamos o SciMultaneous, uma ferramenta desenvolvida para trabalhar em conjunto com um gerenciador de execução de *worklfows* científicos, capaz de verificar se alguma atividade da cadeia de execução do workflow deve ser re-executada e efetivamente executá-la novamente em decorrência de algum tipo de falha, aumentando assim as chances de sucesso do experimento.

4.1 Arquitetura

Quando pensamos em computação em nuvem, dentre outros assuntos, pensamos nos desafios referentes à tolerância às falhas. No que tange à execução de *workflows* em nuvem, ainda existem muitos pontos a serem aperfeiçoados (Oliveira et al. 2010b) principalmente na seara de controle de falhas e de re-execução de atividades. O arcabouço mínimo dos programas de controle de execução em paralelo de *workflows* científicos, tais como o SciCumulus, não fornece apoio à re-execução de atividades e tarefas que falharam. Porém, idealmente, estes deveriam ser capazes de identificar e contornar falhas, desta forma apoiando execuções confiáveis. Neste contexto o SciMultaneous funcionaria como uma agente autonômico possibilitando que os SGWfCs se recuperem das falhas sem intervenção humana semelhante a abordagem apresentada por Pinheiro et al. (2006).

Existem dois níveis de tolerância que podemos considerar: nível de tarefa e nível de *workflow*. O primeiro tipo é utilizado em sistemas paralelos e distribuídos, e está relacionado às falhas das tarefas das atividades do *workflow* que foram executadas em paralelo. O segundo requer técnicas aplicadas ao fluxo de execução, usadas para tratar condições de erro do *workflow* como um todo, o que está fora do escopo do nosso trabalho e soluções já foram propostas para casos como este (Ferreira et al. 2010).

Temos grandes vantagens no tratamento de erros e re-execuções no ambiente de nuvem, pois devido à sua elasticidade e disponibilidade, podemos traçar heurísticas variadas, de acordo com a importância da atividade e de suas tarefas associadas, que

precisam ser monitoradas. Assim, foi proposto e desenvolvido o SciMultaneous, uma ferramenta que implementa um conjunto de heurísticas para re-execução de atividades de *workflows*, inspiradas nas estratégias de controle presentes em SGBD.

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 poucas tarefas 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 associada) e sejam capazes de tomar decisões, para contornar problemas na execução. A ideia é trabalhar no nível de tarefa, avaliando o repositório de proveniência (que possui dados em tempo real) em busca de tarefas que tenham sido executadas com erro. Tais tarefas são então re-executadas pelo SciMultaneous.

O SciMultaneous é uma arquitetura de serviço que visa prover mecanismos de re-execução de tarefas com problemas, explorando as características do ambiente de nuvem. O SciMultaneous baseia-se na utilização dos dados de proveniência gerados em tempo de execução coletados da execução de um *workflow* científico. A Figura 5 Arquitetura conceitual do SciMultaneous acoplado ao SciCumulus apresenta a arquitetura conceitual do SciMultaneous. A arquitetura do SciMultaneous é composta por um conjunto de serviços independentes baseada no modelo de *software* como um serviço (SaaS) (Vaquero et al. 2009). Três principais serviços são parte da arquitetura do SciMultaneous: Mapeador de Tarefas (MT), Configurador de MV (CMV) e o Executor de Tarefas (ET). O MT consulta a base de proveniência para analisar os dados gerados em tempo de execução procurando qualquer indicativo de falha nas tarefas, ou seja, este é o serviço responsável por identificar os erros. Neste momento é fundamental ter acesso à proveniência em tempo de execução, porque o MT pode começar a re-execução o mais rápido possível, melhorando assim o desempenho. Uma vez que o MT identifica tarefas que devem ser executadas novamente, cada uma delas marcadas para re-execução é colocada em uma fila para ser consumida pelo ET. O MT também define qual heurística deve ser utilizada (mais detalhes a seguir). Com a heurística escolhida, MT informa ao CMV que existe uma demanda por re-execuções. CMV analisa se há alguma MV ociosa que pode ser usada ou se CMV tem que instanciar novas máquinas virtuais para a re-execução. Com o ambiente configurado, CMV invoca ET que

efetivamente re-executa tarefas em cada máquina virtual criada por CMV. ET também é responsável pela captura e armazenamento de dados de proveniência (relacionado com as re-execuções) no repositório de proveniência.

A fim de armazenar dados de proveniência relacionados com as re-execuções, SciMultaneous estende o modelo de proveniência utilizado pelo sistema de gerenciamento de execução de *workflows* escolhido. Neste trabalho, estendemos o modelo de proveniência utilizado pelo SciCumulus para armazenar dados adicionais de proveniência referentes à re-execução das atividades. Na seção 4.3 detalhamos as novas estruturas de dados criadas para tornar possível o armazenamento da proveniência das re-execuções. Lá é apresentado um trecho do modelo de proveniência do SciCumulus junto com as classes que pertencem ao modelo do SciMultaneous. Tanto o modelo do SciCumulus quanto o do SciMultaneous baseiam-se na recomendação do modelo de proveniência OPM (Moreau et al. 2008a) transformado em PROV.

Conforme foi dito, toda a informação do modelo de proveniência do SciCumulus é capturada por seus componentes na camada de execução. As classes representam a estrutura do *Workflow* e suas atividades. Cada atividade é decomposta em várias tarefas paralelas, que consomem os valores dos parâmetros (Relação, Campo, e valor) e arquivos de dados (arquivo).

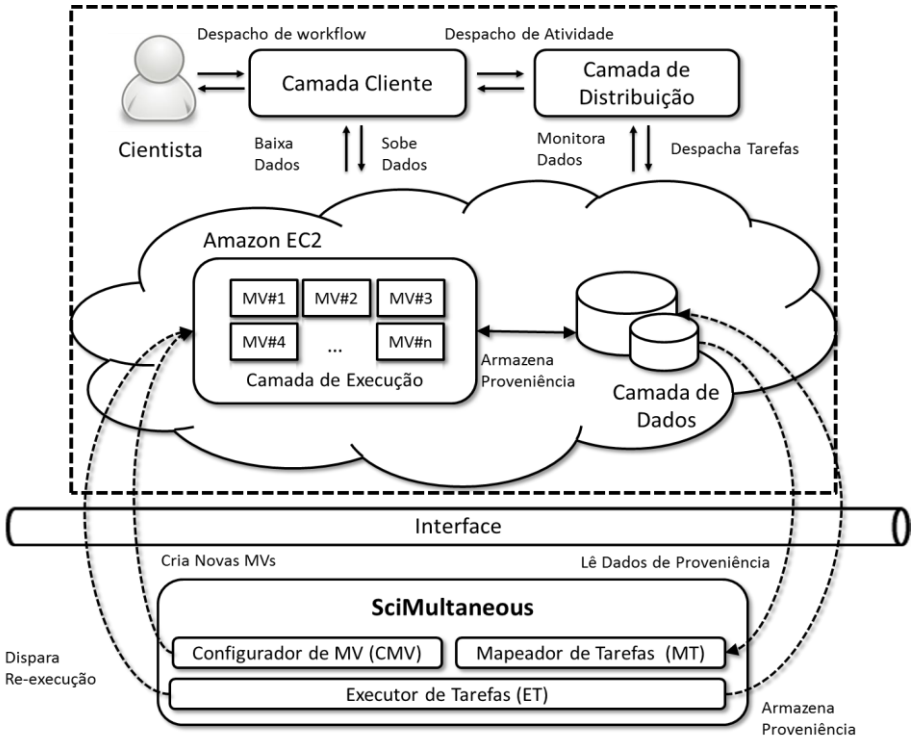

Figura 5 Arquitetura conceitual do SciMultaneous acoplado ao SciCumulus

Para aumentar a taxa de sucesso da execução de um *workflow*, o SciMultaneous adota duas heurísticas básicas, sendo a primeira (H1) a execução com redundância de tarefas consideradas fundamentais, estando assim mais preparado 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 a classificação das *tarefas*. Esta primeira heurística então, busca nos princípios já consolidados em SGBD sua inspiração (Elmasri e Navathe 2006). As atividades classificadas como fundamentais terão sua execução replicada em mais de uma máquina virtual, seguindo a mesma ideia de um *backup* de banco de dados. No entanto, quando pensamos no *backup* de banco de dados, podemos classificá-lo como um ponto de recuperação, pois os *backups* irão armazenar um estado consistente até certo ponto. Tudo aquilo que for realizado após este ponto, não existirá em sua massa de dados. Pensando de novo em termos de bancos de dados, para recuperar totalmente essa base, teríamos que recorrer, caso exista, a um *log* de transação (do inglês *transaction log*) (Elmasri e Navathe 2006), de modo a re-executar todos os comandos que foram feitos na base, após o *backup* restaurado. Voltando para o nosso controle de execução do *workflow*, nosso *backup* seria feito em tempo real, criando uma redundância da execução da tarefa em questão. No caso de falha de processamento na máquina virtual original, a execução do *workflow* seria desviada para a máquina virtual redundante, sem perda significativa de desempenho.

A segunda heurística (H2) trata do monitoramento contínuo das tarefas. No caso de falha, podemos direcioná-las 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 temos que buscar uma máquina virtual que já possua os programas que a tarefa irá invocar, previamente instalados. Nesta segunda heurística, nos casos de falha seguiremos a seguinte estratégia: Primeiro tentar-se-á 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, partimos para a execução em outra máquina virtual da nuvem. Por fim, se ainda assim não obtivermos sucesso, partimos para a execução em outro provedor de nuvem, com maior poder de processamento por máquina virtual (ou mais confiável). Esta abordagem busca

aproveitar a localidade dos dados, da primeira nuvem, pois se a tarefa em questão for dependente de dados, estes deverão ser migrados entre as nuvens, o que demandará um gasto. Esta abordagem se assemelha à re-execução de *log* de transação. Partindo de um estado consistente de execução das tarefas do *workflow*, re-executamos a tarefa que apresentou problemas. Esta tentativa de re-execução será realizada um número finito de vezes, e no caso de falha em todas as tentativas, este *workflow* será marcado como inválido, e deverá ser avaliado pelo cientista. Assim como nos SGBDs, acreditamos que quanto maior a preocupação com falhas e recuperação, maior será a sobrecarga computacional. No entanto, com as heurísticas propostas nessa dissertação, buscamos minimizar ou corrigir as falhas que possam surgir, melhorando assim, na média, o desempenho de execuções do *workflow*. Em ambas heurísticas tomamos como premissa que o cientista dispõe de duas ou mais nuvens computacionais para que o SciMultaneous 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.

4.2 Desenvolvimento do SciMultaneous

A última versão do SciMultaneous foi desenvolvida utilizando o Java Versão 6.15 com algumas bibliotecas adicionais, destacando-se, em virtude de sua importância 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 utiliza a 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*. Em alguns casos precisamos disparar diversas tarefas em paralelo, sendo essa técnica uma ótima alternativa. 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) o paralelismo aninhado do Java com MPJ é bem adequado. 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 (MP4-5b.dyndns.info). O executável do SciMultaneous com as dependências embutidas encontra-se em fase de liberação e será disponibilizado em

https://sourceforge.net/projects/scicumulus/. A seguir explicamos com mais detalhes passos importantes da implementação do SciMultaneous.

Conforme descrito anteriormente, para tratar as falhas ocorridas, um ambiente de nuvem, independente do qual o experimento está sendo executado, se faz necessário. É preciso informar ao SciMultaneous onde este poderá re-escalonar as tarefas que precisam ser re-executadas. Os parâmetros de configuração do ambiente de re-execução são informados por meio de um arquivo de inicialização, com as seguintes informações:

i. Provedor de nuvem computacional disponível para utilização. É possível indicar mais de um provedor para esse propósito. Provedores diferentes fornecem máquinas virtuais de diferentes capacidades. Sendo assim, adotamos a estratégia de classificar o poder de processamento dessas máquinas em cinco níveis, para assim, equalizar a classificação de instâncias de diferentes nuvens, independente da nomenclatura original utilizada pelo provedor do serviço. Adotamos a nomenclatura utilizada pela Amazon, por se tratar do provedor com o qual mais trabalhamos. A nomenclatura adotada segue hierarquicamente como:
    a. Micro
    b. Large
    c. Extra-Large
    d. Extra-Large de Alta Capacidade
    e. Extra-Large Quádrupla
ii. Usuário e senha de conexão com cada um dos provedores.
iii. Classificação dos provedores em dois níveis, A e B onde A é o mais confiável e em geral mais caro.

Os demais parâmetros necessários para a re-execução de uma tarefa, serão obtidos dos dados de proveniência da execução do experimento.

Outro ponto importante para o bom resultado da heurística de execução redundante proposta, trata-se da forma como classificamos as tarefas quanto ao seu grau de complexidade. Atualmente o único parâmetro utilizado para este propósito é o tempo que esta leva para terminar e o tipo da máquina utilizada para atingir esse desempenho. Para chegar a um valor absoluto de tempo de execução, estamos utilizando a média mais 2,33 vezes o desvio padrão das execuções históricas. Uma vez de posse desses

valores, por um processo empírico, classificamos as atividades em quatro níveis de complexidade:

a. Simples
b. Médias
c. Complexas
d. Muito Complexas

Observamos ainda se historicamente a tarefa analisada apresentou algum tipo de falha. Estas tarefas têm maior probabilidade de apresentar algum tipo de mau funcionamento e merecem ser tratadas de forma diferenciada.

4.3 Representação da Proveniência para Re-execuções de Tarefas com Falhas

A fim de armazenar dados de proveniência relacionados com as re-execuções, SciMultaneous estende o modelo de proveniência utilizado pelo sistema de gerenciamento de execução de *workflows* escolhido. Neste trabalho, estendemos o modelo de proveniência utilizado pelo SciCumulus para armazenar dados adicionais de proveniência referentes às re-execuções. Tanto o modelo do SciCumulus quanto o do SciMultaneous baseiam-se na recomendação do modelo de proveniência OPM, transformado em PROV.

Devido à complexidade dos experimentos científicos e o enorme número de variáveis envolvidas no controle de execução das atividades nos ambientes de nuvem, o modelo de dados do SciCumulus conta com muitas tabelas, sendo as principais as apresentadas na figura 6. A partir dessas tabelas podemos estender o modelo original para tornar-se apto a armazenar a proveniência da re-execução.

À medida que SciMultaneous re-executa tarefas que falharam, nós temos que representar explicitamente estas re-execuções, para manter a coerência dos dados de proveniência. Sendo assim, SciMultaneous tem o seu próprio modelo de proveniência que é complementar ao modelo de proveniência do SciCumulus. O Modelo proveniência do SciMultaneous representa as re-execuções das tarefa (SciMultan.Tarefas), que estão associadas à execução de tarefa original (neste caso Tarefa); os arquivos produzidos (SciMultan.Arquivo) e o usuário que tenha configurado SciMultaneous (SciMultan.Usuario). Como SciMultaneous é baseado em estratégias

para re-execuções em nuvem (heurísticas), é interessante também representar as máquinas virtuais utilizadas (SciMultan.MaquinaVirtual), seus tipos (SciMultan.TipoMaquinaVirtual) e os provedores do serviço (SciMultan.Provedor). As SciMultan.MaquinaVirtual podem corresponder à representação conceitual de um artefato, uma vez que tem a mesma semântica, ou seja, ele modela as mesmas estruturas digitais em sistemas computacionais (parâmetros, base de dados, arquivos e instâncias). A classe SciMultan.User representa claramente o conceito de agente enquanto SciMultan.Task também é mapeado como um processo de OPM. As informações de re-execução poderiam ser armazenadas no modelo de SciCumulus, no entanto optamos por armazenar as informações de re-execução em separado, para tornar o SciMultaneous independente da ferramenta a que for acoplado.

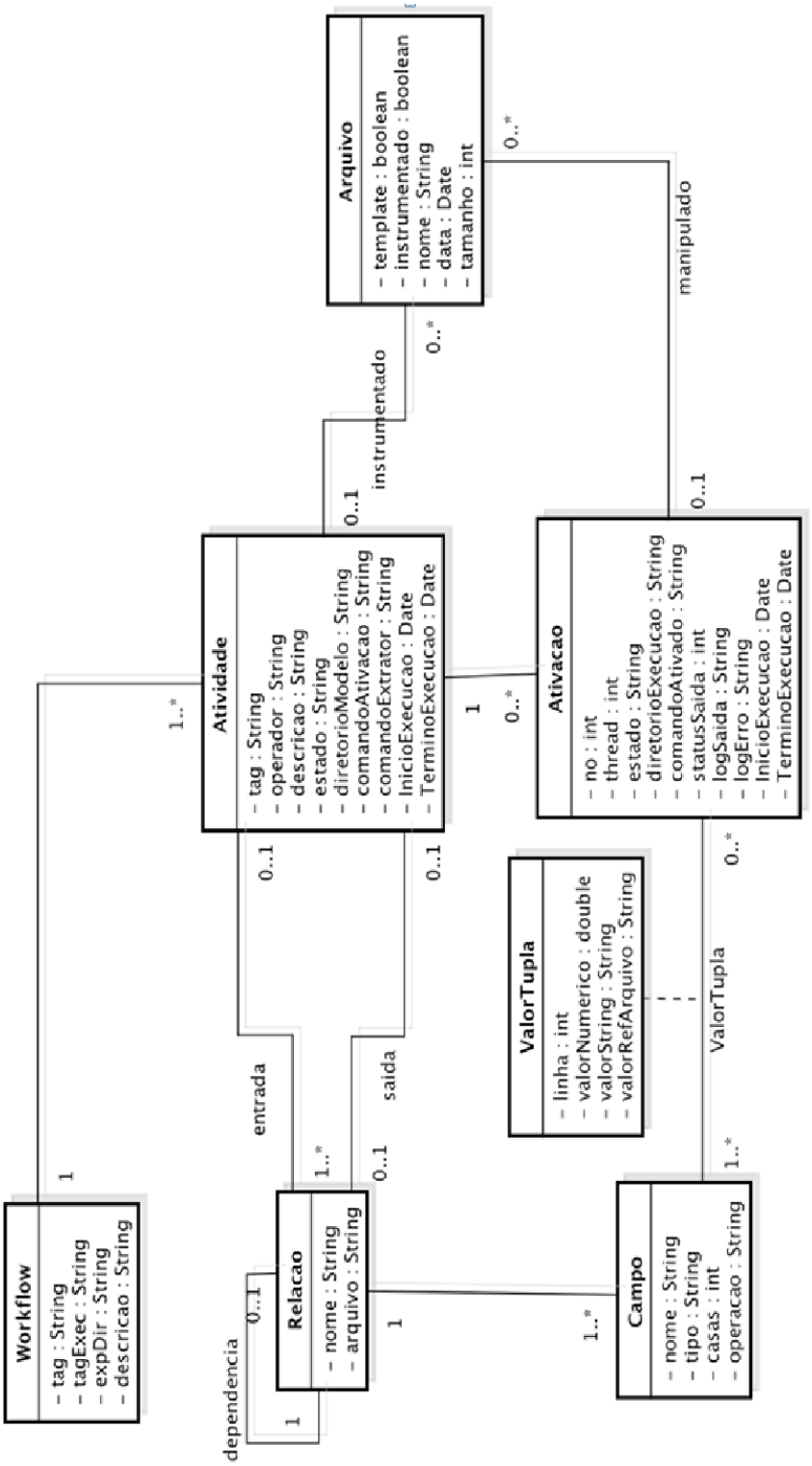

Figura 6 Modelo de Dados Simplificado do SciCumulus

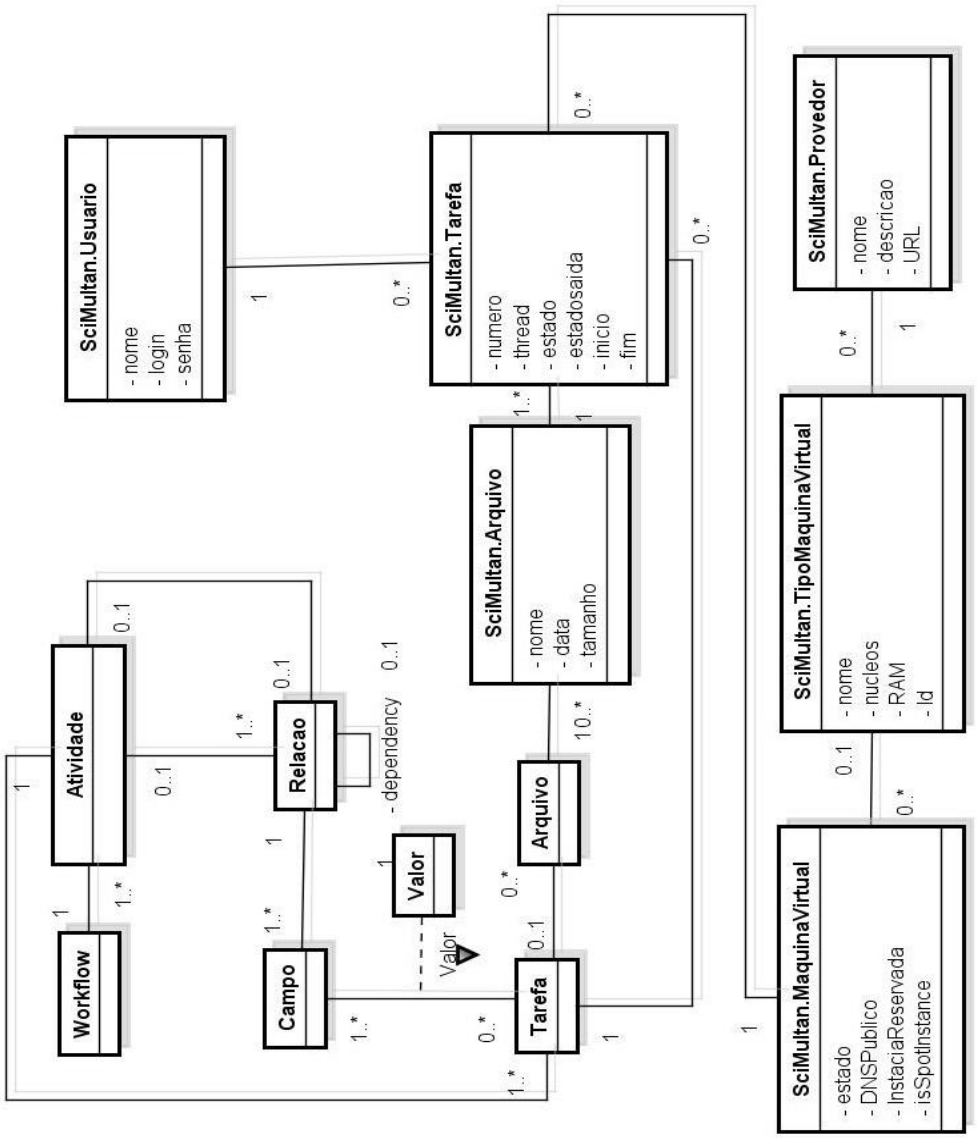

Figura 7 Novas Entidades para Registro da Proveniência de Re-execução.

Figura 7 apresenta alguns elementos do modelo do SciCumulus que poderiam ser usados para o armazenamento das informações referentes às re-execuções.

Capítulo 5 - Avaliação Experimental

Este capítulo apresenta as configurações utilizadas para modelar e executar o *workflow* SciPhy utilizando o SciMultaneous acoplado ao SciCumulus, bem como os resultados experimentais obtidos. A ideia central deste capítulo é analisar o desempenho e a escalabilidade do SciMultaneous. Na seção 5.1 descrevemos as características do workflow utilizado neste estudo de caso dando uma dimensão das dificuldades envolvidas na sua utilização. Nas seções 5.2 e 5.3 apresentamos o ambiente onde o experimento foi executado e como procedemos para prepara-lo. Nas seções seguintes, a 5.4, 5.5 e 5.6 discutimos algumas questões relacionadas às observações realizadas no que tange à revocação e precisão, desempenho e custo financeiro.

5.1 O SciPhy

Experimentos de bioinformática estão evoluindo rapidamente no contexto de projetos genômicos que analisam grandes quantidades de dados. Este fato traz consigo uma maior necessidade de infraestrutura computacional. O SciPhy (Ocaña et al. 2011b) é um workflow científico que foi desenvolvido para gerar árvores filogenéticas com máxima verossimilhança. 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: A construção do alinhamento genético, a conversão de formato do alinhamento, a pesquisa sobre o melhor modelo evolutivo a ser usado, a construção da árvore filogenética, a concatenação dos alinhamentos produzidos de forma a gerar um super-alinhamento que servirá para inferir a relação do genoma completo, a escolha do melhor modelo evolutivo para o super-alinhamento e a construção da árvore filogenética que contém informações sobre todos os organismos associados. Estamos focados em soluções que tornem a execução de workflows como este, uma tarefa mais simples, eficiente e confiável. Especificamente, esta dissertação trata das estratégias para aumentar o grau de confiabilidade da execução de workflows científicos usando como estudo de caso o workflow SciPhy. As atividades acima citadas executam alguns programas da área biomédica, tais como: o MAFFT, o Kalign, o ClustalW, o Muscle, o ProbCons, o ReadSeq(Gilbert 2003) o ModelGenerator e o RAxML, conforme apresentado na Figura 8. 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 de suas tarefas. Uma alternativa dentro deste contexto é utilizar o processamento paralelo na nuvem, onde é possível instanciar tantas máquinas virtuais quanto se julgar necessário, obtendo-se assim o poder de processamento para que o cientista obtenha o resultado do seu experimento em um tempo satisfatório.

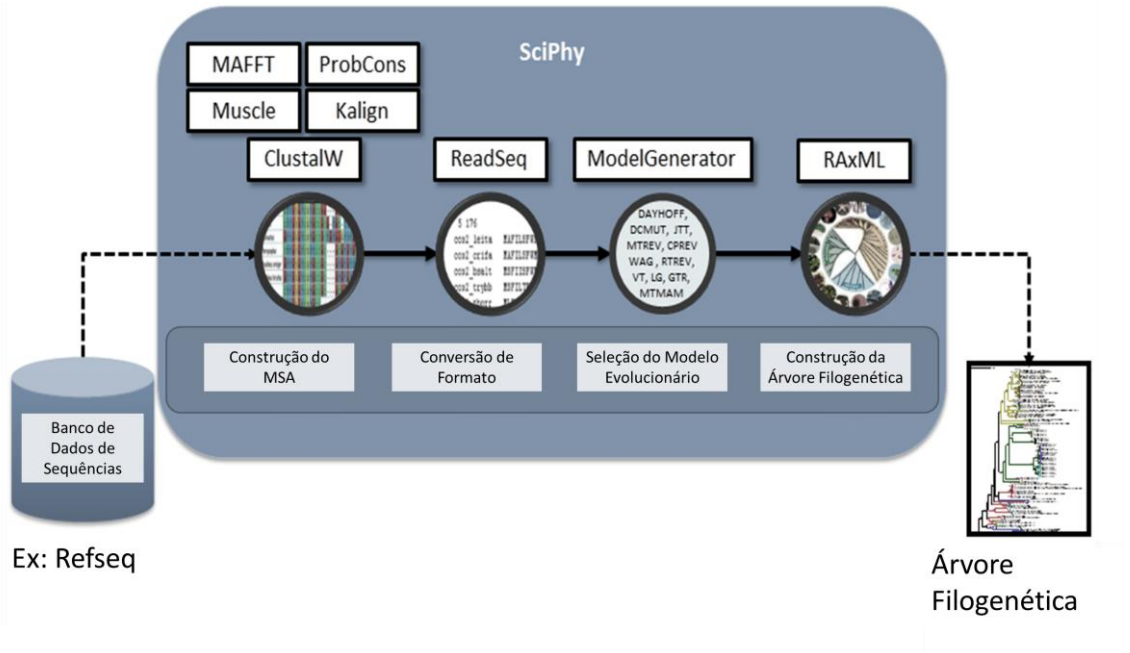

Figura 8 Fases da execução do SciPhy

Normalmente, uma execução do SciPhy em paralelo na nuvem, consumindo 50 arquivos multi-fasta gera 1.250 atividades paralelas e exige cerca de 4 dias utilizando-se 16 núcleos virtuais. Principalmente devido às flutuações de desempenho no ambiente de nuvem, uma execução típica de SciPhy apresenta entre 2% e 9% de falha de atividades,

falhas estas que podem exigir até mais 10 horas de processamento para o exemplo analisado

5.2 Configuração do Ambiente

Para os experimentos executados nesta dissertação utilizamos a nuvem da Amazon (Amazon EC2) que fornece vários tipos de máquinas virtuais aos 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).

Cada uma das instâncias utiliza processadores equivalentes ao Intel Xeon quad-core. Cada máquina virtual instanciada nos experimentos de análise filogenética nesta dissertação 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 dissertação).

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. 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.

5.3 Configuração do Experimento

Para executar o SciPhy, variamos o programa de alinhamento e o modelo evolucionário utilizado no *workflow* científico. 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 de 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 filogenéticas usando o RAxML-7.2.8-alpha. O *workflow* SciPhy foi modelado utilizando o SciCumulus.

5.4 Análise de Revocação e Precisão

O SciMultaneous utilizado em conjunto com o SciCumulus foi avaliado no ambiente Amazon EC2. No experimento apresentado neste trabalho foram utilizadas MVs do tipo large (7,5 GB RAM, 850 GB de armazenamento, 2 núcleos). Cada uma das MVs usa processadores quad-core Intel Xeon processors@2.33GHz. Cada MV instanciada apresentada neste artigo usa sistema operacional Cent Linux OS 5 (64 bits). Nesta dissertação, executamos o *workflow* SciPhy como estudo de caso. O conjunto de dados

contém 250 arquivos multi-fasta e cada arquivo multi-fasta é constituído por uma média de 10 sequências biológicas. Para cada arquivo de entrada, o SciPhy (representado por cinco atividades) é executado cinco vezes (uma para cada método de alinhamento: ClustalW, Kalign, MAFFT, músculo e ProbCons) gerando assim um total de 6.250 tarefas paralelas, onde aproximadamente 1,96% (totalizando assim 123 tarefas) apresentaram algum tipo de falha, conforme ilustrado na Tabela 1

Tabela 1 Cenário de Falhas do Estudo de Caso

Tipo de Falha	Total	Identificadas	Corrigidas	Precisão	Revocação
Terminação Inesperada	56	56	54	96.42%	100%
Arquivo não Produzido	58	58	57	98.27%	100%
Arquivo de Entrada não Encontrado	9	9	0	0%	100%

O SciMultaneous considera que há três tipos de falhas ao analisar dados de proveniência gerados em tempo de execução: (i) encerramento inesperado - quando a tarefa termina sem razão, (ii) nenhum arquivo produzido - quando a tarefa não produz quaisquer arquivos, mas termina sem apresentar mensagem de falha, e (iii) dados de entrada não encontrados - quando a tarefa não pode ser processada por falta de dados de entrada. Em primeiro lugar, para analisar se o SciMultaneous pode melhorar a confiabilidade de execução do *workflow*, foi realizado um experimento para avaliar se esse é capaz de identificar falhas das tarefas e reprocessá-las adequadamente. Para realizar esta análise, foram utilizados indicadores de Recuperação da Informação, tais como precisão e revocação (Baeza-Yates e Ribeiro-Neto 2011). Neste trabalho, a revocação indica o percentual de falhas de tarefas que foram corretamente identificadas pelo SciMultaneous. A precisão indica a porcentagem de tarefas que foram re-executadas com sucesso depois de identificadas como tarefas que falharam.

Para cada um dos três tipos de erros, o SciMultaneous apresentou um valor de perfeito revocação, o que significa que é capaz de identificar todas as falhas com base em dados de proveniência gerados em tempo de execução. Isto pode ser explicado por conta do bom registro de *log* gerado pela execução do SciCumulus. O SciMultaneous

busca no repositório de proveniência do SciCumulus pelos códigos indicativos de falhas nas tarefas. Por outro lado SciMultaneous não pode executar corretamente todas as tarefas identificadas. Em 90,24% dos casos, SciMultaneous re-executou tarefas de forma adequada uma vez que os problemas identificados foram devidos a flutuações no ambiente ou problemas intermitentes nas MVs. Nos casos em que as falhas foram decorrentes de falta de dados de entrada, SciMultaneous tentou reprocessá-las usando as heurísticas H1 e H2, mas não poderia corrigir o problema, já que nas novas re-execuções (mesmo em máquinas virtuais diferentes) os dados não existiam e SciMultaneous não pode obtê-los automaticamente.

5.5 Análise de Desempenho

Em todos os cenários testados a utilização das heurísticas representou ganhos, mas precisamos deixar claro que tipos de ganhos foram esses.

Passada a análise de revocação e precisão, um segundo experimento foi realizado. Buscamos avaliar se a partir de dados de proveniência gerados em tempo de execução, o SciMultaneous poderia, além de aumentar a confiabilidade na execução, também melhorar o desempenho (quando comparado com a re-execução posterior de tarefas que falharam) da execução do *workflow*. Este é um ponto muito importante neste contexto, uma vez que reduzir o tempo total de execução é uma prioridade em ambientes como nuvens, devido aos custos financeiros (Cavalcante et al. 2012). Deste modo, executamos o *workflow* SciPhy variando o número de núcleos virtuais utilizados em cada execução. Foram comparadas as duas heurísticas propostas pelo SciMultaneous (representadas como H1 e H2 na Figura 9) com uma abordagem *ad-hoc* na qual as tarefas que falharam são re-executadas no final da execução do *workflow*, uma vez que nas abordagens existentes a proveniência é fornecida no final da execução do *workflow*, tais como no SGWfC Kepler.

Tanto a heurística H1 quanto a H2 apresentaram um tempo total de execução inferior à abordagem *ad-hoc* em todos os casos. Por exemplo, usando 32 núcleos, a heurística H1 levou 9,90 dias para finalizar a execução do *workflow*, a heurística H2 executou em 9,63 dias e a *ad-hoc* executou em 10,39 dias. No caso da heurística H1, 9,2% (170 tarefas) das tarefas de longa duração associadas aos programas ModelGenerator e RAxML (Para detalhes, consulte Ocaña et al (Ocaña et al. 2011b)) foram executados de forma redundante (criando uma tarefa original e uma redundante -

uma vez que estes programas são classificados como críticos). Aproximadamente 8% das tarefas originais de longa duração falharam e foram automaticamente substituído pela tarefa redundante, SciMultaneous por meio da consulta à base de proveniência as identificou. Embora a heurística H1 antecipe a re-execução de muitos erros de tarefas críticas, também reduz a taxa de vazão, uma vez que executa tarefas que não necessariamente precisariam ser re-executadas. Desta forma, a heurística H1 cria mais processamento no experimento (quando comparada à abordagem *ad-hoc*), mas reduz a necessidade de re-execuções posteriores, reduzindo assim o tempo total de execução em relação à abordagem *ad-hoc* e a própria H2. No caso da heurística H2, apenas as tarefas que falharam foram re-executadas (1,96% do total), assim que o erro seja identificado na base de proveniência (o tempo de varredura é configurável, mas pode ser bem pequeno, da ordem de 1 segundo, tempo aproximado para execução da consulta). Esta é uma vantagem, uma vez que várias falhas em tarefas de longa duração são causadas por falhas anteriores em tarefas dependentes de menor duração (devido à dependência de dados). Quando corretamente re-executadas, muitas dessas tarefas de curta duração (por exemplo, MAFFT ou ClustalW) evitam problemas futuros nas tarefas de longa duração, reduzindo assim o tempo de execução global, melhorando a velocidade do experimento quando comparado com as outras abordagens. SciMultaneous alcançou uma melhoria de desempenho de até 11,5% quando se utiliza a heurística H1 comparado a estratégia *ad-hoc*. Esse percentual pode representar um ganho enorme quando nos referimos a valores absolutos, pois pode significar mais que um dia da execução para muitos experimentos. Além disso, em experimentos maiores, esse ganho de desempenho pode ser ainda maior do que o alcançado nesta avaliação. Destacamos ainda que essas melhorias de desempenho levaram a uma redução de até US$ 373,24 em encargos financeiros, quando comparados com a execução *ad-hoc*. A Figura 9 apresenta graficamente uma análise comparativa das diferentes estratégias apresentadas, em termos do tempo de execução.

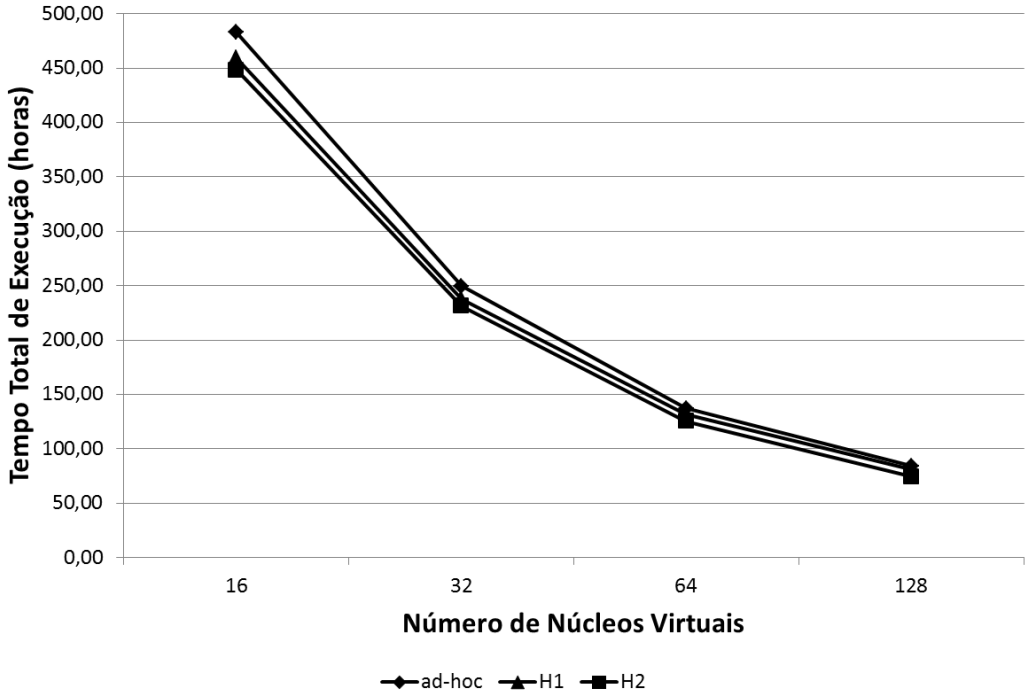

Figura 9 Tempo de Execução de Diferentes Heurísticas

Na Figura 9 podemos observar que o comportamento do tempo de execução do *workflow* segue o mesmo padrão nas três abordagens. O gráfico nos passa a sensação que o ganho obtido com a utilização das heurísticas H1 e H2 não é relevante quando comparamos com a abordagem ad-hoc. No entanto vale lembrar que quando olhamos para os valores absolutos de economia de tempo chegamos a 10 horas a menos de esforço. Sem contar as já mencionadas economia financeira.

A Figura 10 apresenta uma análise semelhante, focando na aceleração do experimento.

Podemos observar que a heurística H2 apresentou o melhor resultado de aceleração uma vez que só re-executa atividades que apresentam falhas. Dessa forma não cria processamento extra desnecessário. Já a heurística H1 apresenta uma redução da aceleração do experimento pois executa tarefas de forma redundante. No entanto, estas tarefas redundantes fazem com que o experimento economize na média tempo total de execução. As tarefas redundantes serão utilizadas sempre que uma tarefa da cadeia principal falhe. Quando isto ocorre, basta utilizar-se o resultado da tarefa redundante (caso exista) economizando assim tempo de reprocessamento.

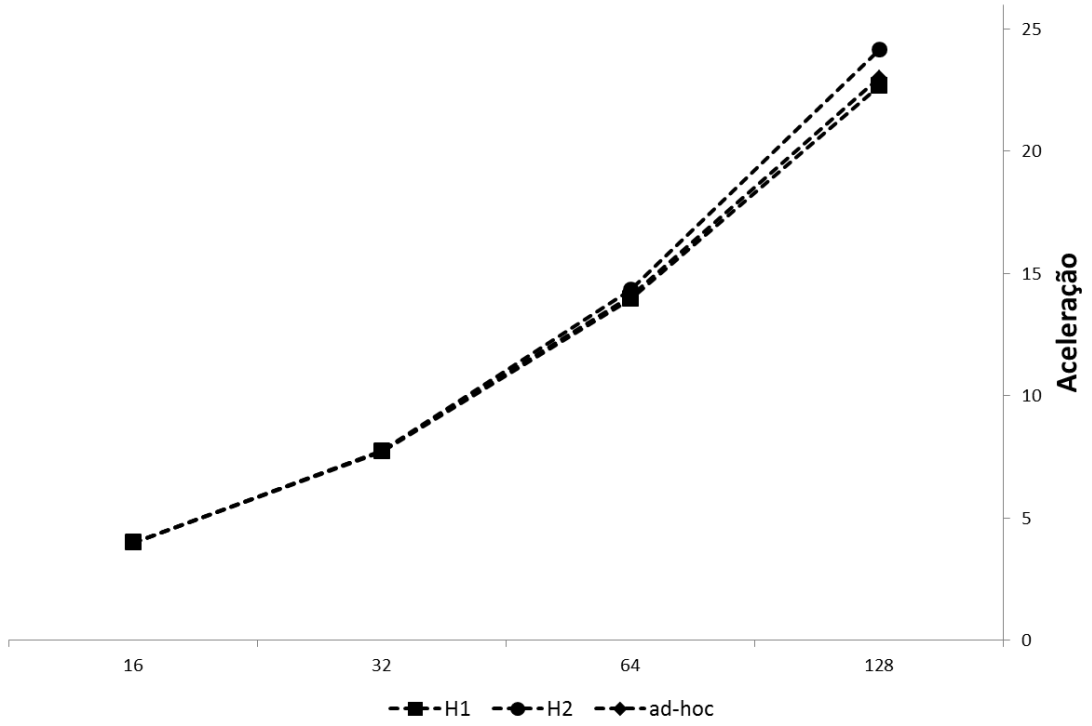

Figura 10 Aceleração do Experimento para as Diferentes Abordagens

Neste segundo experimento utilizamos quatro MVs da Amazon EC2 (2 micro e 2 grandes). As MVs grandes foram classificadas como de maior poder de processamento e foram usadas para a re-execução de tarefas consideradas essenciais (ModelGenerator e os RAxML). Utilizando ambas as heurísticas, SciMultaneous foi capaz de identificar corretamente as falhas e corrigir a maioria delas, garantindo assim um aumento na confiabilidade da execução do *workflow*, sem descuidar do desempenho.

Quando o cientista utiliza a abordagem *ad hoc,* este inicia o experimento e irá verificar se problemas ocorreram ao término da execução. Supondo que determinado experimento ocorresse sem nenhum erro, a abordagem *ad hoc* apresentaria o mesmo desempenho de quando utilizamos a heurística H2 e seria melhor que a heurística H1. Neste cenário hipotético, todas as atividades executadas de forma redundante pela heurística H1 representariam um trabalho desnecessário e a heurística H1 não iria executar nenhuma tarefa, haja vista, que não seriam identificados erros. Mas a realidade das execuções na nuvem é diferente. Mesmo com todo o cenário de execução perfeito, com todos os parâmetros de entrada disponíveis, não conseguimos executar SciPhy sem que algum erro ocorresse. Sendo assim, sem nenhum tipo de abordagem especial, fica a cargo do cientista, aguardar o término do experimento para verificar se houve falha em

alguma tarefa e re-executá-la com uma gerência ad-hoc. Ao longo do texto desta dissertação, o tempo ganho em relação à abordagem *ad hoc* representa o tempo de processamento que será gasto nesta atividade de re-execução pós término do *workflow*. Poderíamos ainda incluir o tempo de análise gasto pelo cientista na tentativa de descobrir tarefas com problemas. Por se tratar de algo subjetivo, deixamos esta situação fora do escopo da análise.

Verificamos ainda que a quantidade total de falhas aumenta quando utilizamos a abordagem *ad hoc*. Isto é explicado pela característica do *workflow*. Como estamos lidando com um experimento onde existe uma forte dependência de dados, em que a entrada de um programa muitas vezes é a saída de outro executado previamente, caso ocorra uma falha ao longo dessa sequência de encadeamento, toda a cadeia falha. Ao utilizarmos alguma das heurísticas propostas, minimizamos esse problema, pois ao resolver falhas tão logo essas ocorram, diminuímos o risco de problemas em cadeia.

5.6 Análise de Custo Financeiro

A fim de verificar o custo monetário envolvido na execução do experimento, analisamos os dados de proveniência gerados e calculamos o custo final utilizando o modelo de cobrança baseado em uma janela de tempo de 1 hora (método de pagamento da Amazon EC2). Existe ainda outra forma de precificação que é baseada 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). A tabela 2 resume os dados de desempenho e precificação dos tipos de máquinas virtuais passíveis de uso.

Tabela 3 Configuração de *Hardware* e Preços de Utilização

Tipo de Máquina	Memória	Disco	UC[1]	Núcleos	Arquitetura	Preço[2]
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

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

[2] Preço calculado por hora em dólares

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

É importante destacar que apesar do custo elevado alcançado nestes cenários, conforme podemos ver na Figura 11 estes valores não são de forma alguma proibitivos para a execução do experimento, ou seja, a maioria dos cientistas é capaz de arcar com as despesas associadas a estas re-execuções. Não podemos perder de vista a economia em infraestrutura já comentada anteriormente. Na Figura 11 apresentamos um gráfico ilustrando a variação do custo do uso do *cluster* virtual por hora, ao variarmos as quantidades de núcleos. Ressaltamos que a cobrança do provedor é realizada por hora de MV alocada.

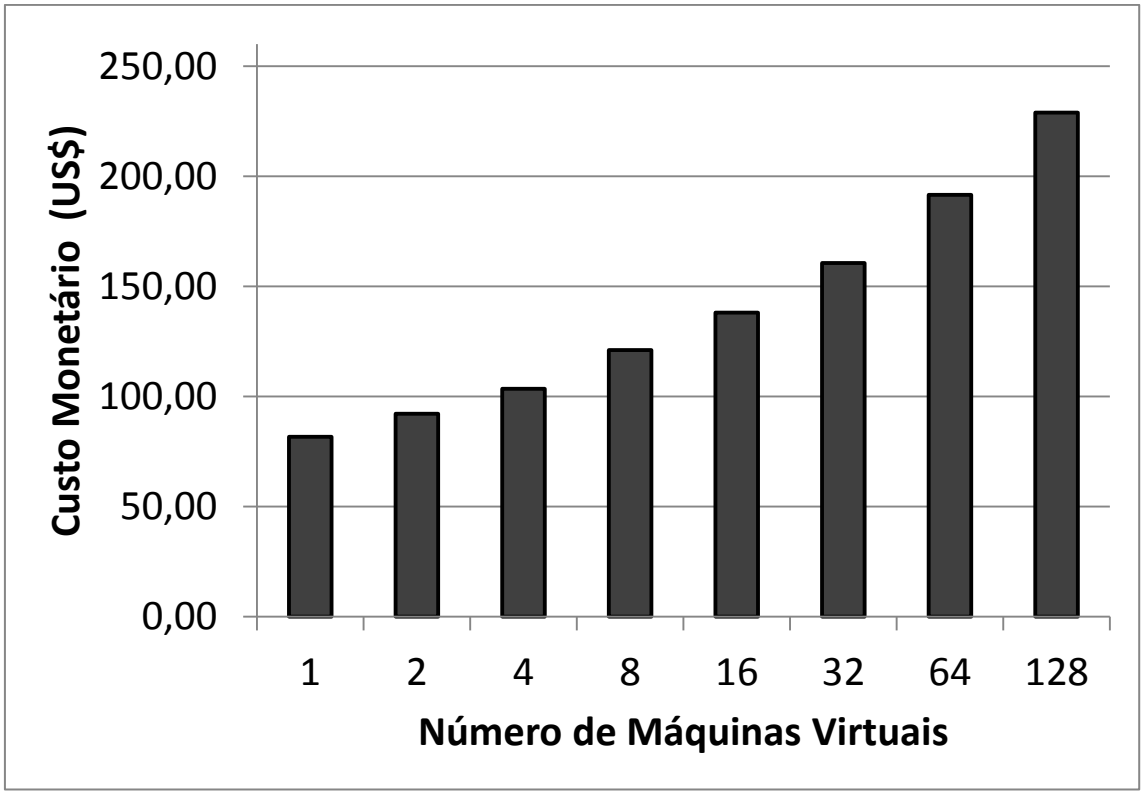

Figura 11 Análise de custo financeiro variando-se o número de nós

Capítulo 6 - Conclusões

A preocupação em evitar o retrabalho devido a falhas de processamento em *workflows* científicos é comum, seja em ambientes de supercomputadores, grids ou nuvens. A importância do apoio ao cientista na gerência dos experimentos científicos em larga escala tem obtido grande destaque, pois com apoio de novas soluções computacionais a ciência avança mais rapidamente e é impulsionada a novos desafios. Especificamente no Brasil, o documento dos "Grandes Desafios" da SBC ressaltou a importância desta gerência na pesquisa brasileira nos próximos anos.

Usando a proveniência gerada em tempo de execução o SciMultaneous é capaz de re-executar tarefas que apresentaram falhas duramente a execução do *workflow*. Com isso conseguimos aumentar a confiabilidade da conclusão do *workflow*, melhorando o seu desempenho e reduzindo o seu custo financeiro total. Experimentos demonstraram que SciMultaneous pôde identificar e re-executar 90,24% (9,76% das tarefas não poderia ser re-executado principalmente devido a falta de arquivos de entrada ou arquivos com dados inválidos) das tarefas com erro. Dessa forma, melhorando o desempenho em até 11,5% usando heurística H1, quando comparado com a abordagem ad-hoc. Essas melhorias também levaram a uma redução de custos financeiros de até US$ 373,24. Tendo em vista estas observações, podemos concluir que a estratégia proposta nesta dissertação foi validada, caracterizando dessa forma que a utilização de heurísticas para controle de re-exucação paralela de atividades com erro é uma forma viável de aumentar o grau de confiabilidade na execução dos *workflows* científicos.

6.1 Contribuições

Essa dissertação apresentou os resultados de uma pesquisa desenvolvida ao longo dos últimos 2 anos e suas principais contribuições são:

i. Estratégias que visam à melhora da eficácia da re-execução de *workflows* científicos em ambientes de nuvens computacionais. Em função da complexidade do escalonamento de atividades computacionais nesses ambientes e também da volatilidade do ambiente, onde máquinas virtuais podem falhar a qualquer momento, buscamos inspiração nas estratégias já consolidadas dos Sistemas de Gerência de Bancos de Dados (SGBDs), para a criação de um

mecanismo baseado em heurísticas para o monitoramento e re-execução das atividades de um *workflow* científico.

ii. Armazenamento diferenciado de dados da proveniência da re-execução possibilitando a avaliação das atividades que foram executadas com erro.

iii. A especificação e o desenvolvimento do SciMultaneous, uma arquitetura baseada em serviços para trabalhar em conjunto com um gerente de execução de *workflows* científicos, capaz de verificar se alguma atividade da cadeia de execução do *workflow* deve ser re-executada em decorrência de algum tipo de falha, aumentando assim as chances de sucesso do *workflow* (e consequentemente do experimento a ele associado).

Os resultados indicaram que o uso da abordagem proposta nessa dissertação propiciou uma melhora na confiabilidade da execução do *workflow* executado em paralelo em ambientes de nuvem computacional.

Os resultados obtidos nesta dissertação foram aceitos para apresentação e publicação em eventos da área, destacando o *4th International Provenance and Annotation Workshop*. Evento tradicional (bianual, desde 2006) do uso de proveniência que apresenta resultados referentes à execução de *workflows* científicos e coleta de dados de proveniência.

i. COSTA, F. S. ; OLIVEIRA, D. ; OCANA, K. ; OGASAWARA, E. ; MATTOSO, M. . Enabling Re-Executions of Parallel Scientific *Workflows* Using Runtime Provenance Data. In: 4th International Provenance and Annotation Workshop, 2012, Santa Barbara, CA. Springer Verlag, 2012. v. 7525. p. 229-232. (Costa et al. 2012)

Ao longo do mestrado, foram publicados outros dois artigos relevantes:

Ainda em fase de evolução, a proposta dessa dissertação foi aceita para publicação no workshop de teses da SBBD 2011

ii. Workshop de Teses e Dissertações em Banco de Dados (WTDBD), 2011, Florianópolis.
COSTA, F. S. ; OLIVEIRA, D. ; MATTOSO, M. . Heurísticas para Controle de Execução de Atividades de *Workflows* Científicos na Nuvem. In: Workshop de

Teses e Dissertações em Banco de Dados (WTDBD), 2011, Florianópolis.(Costa et al. 2011)

Os resultados dos primeiros experimentos do SciCumulus com nuvens computacionais foram descritos em um artigo apresentado no RED 2010 - Third International Workshop on REsource Discovery na França

iii. COSTA, F. S. ; OLIVEIRA, D. ; OGASAWARA, E. ; LIMA, A. A. B ; MATTOSO, M. . Athena: Text Mining Based Discovery of Scientific *Workflows* in Disperse Repositories. In: Third International Workshop on REsource Discovery (RED'2010), 2010, França. Lecture Notes in Computer Science, 2012, Volume 6799/2012, 104-121, DOI: 10.1007/978-3-642-27392-6_8 (Costa et al. 2010)

6.2 Limitações

Toda pesquisa científica possui determinadas limitações. Desta maneira, algumas limitações foram identificadas a partir de uma análise criteriosa da prova de conceito implementada, dos *workflows* executados e dos dados de proveniência:

i. Por decisões de implementação, os componentes do SciMultaneous estão hoje especializados para o ambiente da Amazon EC2. Essa especialização é necessária devido à incipiência dos ambientes de nuvem. Hoje em dia, uma mesma aplicação precisa ser adaptada para executar em diferentes provedores de nuvem (Souza et al. 2012). Idealmente, os componentes devem ser independentes de ambiente de nuvem, mas este cenário ainda está distante. O ambiente da Amazon EC2 foi escolhido por ser o mais estável (senão o único disponível de fato) para que pudéssemos realizar os experimentos.

ii. A versão atual do SciMultaneous é baseada em uma arquitetura de disco compartilhado. Embora este seja o cenário da maioria das aplicações científicas, a solução não se aplica a todos os modelos de armazenamento em nuvens computacionais. Nem sempre poderemos contar com um disco compartilhado ao qual todas as MVs tenham acesso. Desta forma, uma limitação existente é que o SciMultaneous atualmente não se encontra preparado para execuções em ambientes de arquitetura de disco não-compartilhado.

iii. Ao refletirmos sobre a questão da segurança na computação em nuvem, um dos primeiros itens a serem pensados é a mudança de paradigma em relação à

proteção da informação. O uso de um serviço externo, traz consigo a necessidade de se repensar as estratégias de proteção à informação. Nossos dados e processos ficarão em um ambiente onde não teremos o domínio, onde não seremos os administradores, como em geral ocorre em ambientes centralizados. Nesse cenário, temos que buscar informação confiável sobre os serviços disponíveis, os fornecedores existentes e as cláusulas contratuais. Procurar saber como é o controle aos dados de terceiros. Não incluímos no escopo desta dissertação questões referentes à segurança da informação.

iv. Para validar as heurísticas aqui apresentadas utilizamos o SciCumulus como gerenciador de execução paralela. Para poder utilizar o SciMultaneous com outros gerenciadores de *workflow* pequenos ajustes deveriam ser feitos para torná-lo totalmente independente do ambiente onde precise ser acoplado. Um módulo de configuração deverá ser construído, para que por meio de passagem de alguns parâmetros ele possa ser capaz de acoplar-se a outros gerenciadores de *workflow*.

6.3 Trabalhos Futuros

Nos experimentos científicos, os dados consumidos e os dados gerados são de grande importância para os cientistas. Da mesma maneira, os dados de proveniência são fundamentais nesse contexto. Sendo assim, buscaremos com nossas pesquisas futuras encontrar formas de utilização e armazenamento desses objetos buscando soluções que se adaptem melhor às nuvens computacionais que os modelos centralizados. Num estudo ainda incipiente, Os bancos de dados NoSQL parecem ser uma solução mais adequada para aplicativos que não precisem de capacidade complexa de consultas como transações e junções, do que os SGBD tradicionais. Estes novos bancos de dados oferecem alto desempenho, resultado previsível e baixo custo. São fáceis de configurar, operar e escalar. Com esta abordagem, é possível começar, em escala reduzida, a especificar a velocidade de processamento e armazenamento necessários e aos poucos escalar suas necessidades de capacidade. Existem soluções que fragmentam automaticamente os dados em vários nós da nuvem o que pode trazer vantagens na exploração da proximidade dos dados.

Seguindo ainda nesta linha de tirar proveito da proximidade dos dados, para assim ganhar tempo e economia financeira, exploraremos estratégias de fragmentações

de dados, horizontal e vertical, buscando assim manter em cada nó apenas os dados necessários naquele contexto, mas ainda assim fornecer para o usuário um ambiente transparente de consulta aos dados de proveniência.

Com a descentralização do controle da execução e a descentralização dos dados, será importante buscar técnicas que auxiliem na gerência deste ambiente distribuído. Iniciaremos nossas pesquisas, para solucionar esta questão, pelas técnicas de *Peer-to-Peer (P2P)* que podem trazer soluções interessantes para as questões que serão geradas.

Buscaremos também maneiras de isolar os cientistas das especificidades de cada provedor de nuvem, criando uma camada entre o SciMultaneous e os serviços de nuvem buscando inspiração no que foi proposto por Cavalcante et al. (2011).

Experimentaremos também combinar as duas heurísticas avaliando os mesmos indicadores aqui apresentados.

Podemos ainda explorar questões relacionadas a escolha da máquina para re-execução através de uma análise histórica dos tratamentos de falha, melhorando assim a heurística H2. Ou ainda criar um módulo onde o usuário pudesse configurar alguns parâmetros referentes às estratégias de re-execução.

Outra questão ainda em aberto é aquela que trata dos dados consumidos pelos experimentos. Sempre que uma tarefa precisa ser executada, os dados de entrada devem estar à disposição. A ausência desta informação é fonte de erro comum nos experimentos científicos. Seria interessante que as soluções de tratamento de falha, como a que propusemos, fossem capazes de identificar o dado necessário e procurassem por ele não nó na instância onde a tarefa está sendo executada, mas sim, em todo o ambiente da nuvem.

Referências Bibliográficas

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

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., Sabino, E. C., For the NHLBI Retrovirus Epidemiology Donor Study-II (REDS-II), I. C., (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.

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

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

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,

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

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

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

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.

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

Cavalcante, E., Almeida, A., Batista, T., Cacho, N., Lopes, F., C. Delicato, F., Souza, D., Sena, T., F. Pires, P., (2012), "Exploiting Software Product Lines to Develop Cloud Computing Applications". In: *2nd International Workshop on Services,*

Clouds and Alternative Design Strategies for Variant-Rich Software Systems, 2012, Salvador.

Cavalcante, E., Lopes, F., Batista, T., Cacho, N., Delicato, F. C., Pires, P. F., (2011), "Cloud Integrator: Building Value-Added Services on the Cloud". In: *2011 First International Symposium on Network Cloud Computing and Applications (NCCA)*, p. 135 -142

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.

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*, Florianópolis, SC, Brasil.

Costa, F., Oliveira, D., Ocana, K., Ogasawara, E., Mattoso, M., (2012), "Enabling Re-Executions of Parallel Scientific Workflows Using Runtime Provenance Data. In: 4th International Provenance and Annotation Workshop"

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.

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

Crawl, D., Altintas, I., (2008), "A Provenance-Based Fault Tolerance Mechanism for Scientific Workflows", In: Freire, J., Koop, D., Moreau, L. [orgs.] (eds), *Provenance and Annotation of Data and Processes*, Berlin, Heidelberg: Springer-Verlag, p. 152–159.

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

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.

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., Triana, O., Cruz, S. M.

S., Jucá, H. C. L., Cury, J. C., Silva, F. N., Geronimo, G. A., et al., (2008), "ProtozoaDB: dynamic visualization and exploration of protozoan genomes", *Nucleic Acids Research*, v. 36, n. Database issue, p. D547-D552.

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.

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 Science*, p. 31-36, Seattle, WA, USA.

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., Wieczorek, M., (2005), "ASKALON: a Grid application development and computing environment". In: *6th IEEE/ACM International Workshop on Grid Computing*, p. 122-131, Seattle, Washington, USA.

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 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.

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

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

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J., (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.

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., Custer, B., (REDS-II), for the N. R. E. D. S.-I., Component, I., (2011), "Blood transfusion utilization and recipient survival at Hospital das Clinicas in São Paulo, Brazil", *Transfusion*, p. no–no.

Gopi Kandaswamy, Anirban Mandal, Daniel A. Reed, (2008), "Fault Tolerance and Recovery of Scientific Workflows on Computational Grids". , p. 777 - 782, Lyon.

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., Quinn, T., Madau, P., (2010), "Bulgeless dwarf galaxies and dark matter cores from supernova-driven outflows", *Nature*, v. 463, n. 7278 (jan.), p. 203-206.

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.

Gustavo Alonso, Claus Hagen, Divyakant Agrawal, Amr El Abbadi, C. Mohan, (2000), "Enhancing the Fault Tolerance of Workflow Management Systems", *IEEE Educational Activities Department Piscataway, NJ, USA IEEE Concurrency table of contents archive*, v. 8 Issue (jul.), p. 74 - 81.

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.

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

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

Iosup, A., Ostermann, S., Yigitbasi, N., Prodan, R., Fahringer, T., Epema, D., (2011), "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing", *IEEE Trans. Parallel Distrib. Syst.*, v. 22, n. 6 (jun.), p. 931–945.

Jackson, K. R., Ramakrishnan, L., Runge, K. J., Thomas, R. C., (2010), "Seeking supernovae in the clouds: a performance study". In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, p. 421–429, New York, NY, USA.

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

Juve, G., Deelman, E., (2010), "Scientific workflows and clouds", *Crossroads*, v. 16 (mar.), p. 14–18.

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, 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.

Lee, Y. C., Zomaya, A. Y., (2010), "Rescheduling for reliable job completion with the support of clouds", *Future Generation Computer Systems*, v. 26 (out.), p. 1192-1199.

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.

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.

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.

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., Werner, C., Travassos, G. H., Braganholo, V., Murta, L., Ogasawara, E., Oliveira, D., Cruz, S. M. S. da, Martinho, W., (2010), "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.

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.

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., Cohen-Boulakia, S., Davidson, S., Deelman, E., Digiampietri, L., Foster, I., Freire, J., 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.

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.

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.

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.

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.

Oliveira, D., (2012), *Uma Abordagem de Apoio à Execução Paralela de Workflows Científicos em Nuvens de Computadores*. Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2012., UFRJ/COPPE

Oliveira, D., Cunha, L., Tomaz, L., Pereira, V., Mattoso, M., (2009), "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., 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.

Oliveira, D., Ogasawara, E., Baiao, F., Mattoso, M., (2010a), "An Adaptive Approach for Workflow Activity Execution in Clouds". In: *International Workshop on Challenges in e-Science - SBAC*, p. 9-16, Petrópolis, RJ - Brazil.

Oliveira, D., Ogasawara, E., Baião, F., Mattoso, M., (2010b), "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., 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)

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., Busch, M. P., Sabino, E. C., for the NHLBI Retrovirus Epidemiology Study-II (REDS-II), I. C., (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.

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.

Pinheiro, W. A., Oliveira, J., de Souza, J. M., Xexéo, G., Perazolo, M., (2006), "Using autonomic computing and click stream analysis for problem identification in continuous production". In: *Proceedings of the Third international conference on Cooperative Design, Visualization, and Engineering*, p. 17–24, Berlin, Heidelberg.

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*

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.

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., Busch, M., for the NHLBI Retrovirus Epidemiology Donor Study-II (REDS-II), I. C., (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.

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.

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.

Sindrilaru, E., Alexandru Costan, Valentin Cristea, (2010), "Fault Tolerance and Recovery in Grid Workflow Management Systems".

Soanes, C., Stevenson, A., (2003), *Oxford Dictionary of English*. 2nd Revised edition ed. Oxford University Press.

Souza, D., Sena, T., Cavalcante, E., Cacho, N., Batista, T., Lopes, F., Almeida, A., Diniz, T., C. Delicato, F., F. Pires, P., (2012), "Implantação de aplicações em múltiplas plataformas de nuvem". In: *X Workshop em Clouds e Aplicações - WCGA, 2012, Ouro Preto. Anais do X Workshop em Clouds e Aplicações. Porto Alegre, RS: SBC, 2012*

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.

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.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., Lindner, M., (2009), "A break in the clouds: towards a cloud definition", *SIGCOMM Comput. Commun. Rev.*, v. 39, n. 1, p. 50-55.

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.

WfMC, I., (2009), *Binding, WfMC Standards*, WFMC-TC-1023, http://www. wfmc. org, 2000.

Yang Zhang, Anirban Mandal, Charles Koelbel, Keith Cooper, (2009), "Combined Fault Tolerance and Scheduling Techniques for Workflow Applications on Computational Grids". , IEEE Computer Society 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.

Zvelebil, M., Baum, J., (2007), *Understanding Bioinformatics*. 1 ed. Garland Science.