UM SERVIÇO DE CUSTO PARA A EXECUÇÃO PARALELA DE *WORKFLOWS* CIENTÍFICOS EM NUVENS DE COMPUTADORES

Vitor de Gamboa Viana

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

Outubro de 2012

UM SERVIÇO DE CUSTO PARA A EXECUÇÃO PARALELA DE *WORKFLOWS* CIENTÍFICOS EM NUVENS DE COMPUTADORES

Vitor de Gamboa Viana

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:

_____

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

_____

Prof. Daniel Cardoso Moraes de Oliveira, D.Sc.

_____

Prof. Geraldo Zimbrão da Silva, D.Sc.

_____

Prof. Paulo de Figueiredo Pires, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

OUTUBRO DE 2012

Viana, Vitor de Gamboa

Um serviço de custo para a execução paralela de *workflows* científicos em nuvens de computadores / Vitor de Gamboa Viana. – Rio de Janeiro: UFRJ/COPPE, 2012.

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

Orientadores: Marta Lima de Queirós Mattoso

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

Referências Bibliográficas: p. 59-67.

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

Aos meus pais Luiz e Vitoria, a minha irmã Taisa,

aos demais familiares e aos meus amigos,

por tudo o que representam para mim.

Agradecimento

Primeiramente, agradeço a Deus;

Agradeço também aos meus pais, Luiz e Vitoria, por todo suporte, e por sempre me darem condições de atingir meus objetivos. Obrigado por tudo;

À minha irmã Taisa; meus avós Alberto, Nelita, Judith (*in memorian*); e todas as outras pessoas que fazem parte da minha família, por estarem comigo em todos os momentos;

Agradeço à professora Marta Lima de Queirós Mattoso pela orientação nessa dissertação e pelos conhecimentos transmitidos durante todo o mestrado. Foi uma honra ter tido esse contato com quem considero um exemplo de inteligência, responsabilidade e profissionalismo;

Aos professores Geraldo Zimbrão da Silva e Paulo de Figueiredo Pires por terem aceitado fazer parte desta banca;

Ao professor Jano Moreira pelo trabalho exemplar que desenvolve na COPPE. E também por ter me dado a oportunidade de aprender com sua sabedoria e experiência;

Agradeço muito ao Daniel de Oliveira por toda ajuda durante todo o mestrado, pela sua orientação, paciência e boa vontade em todos esses meses. Esse trabalho só foi possível graças à sua ajuda. Assim como a professora Marta, é um exemplo para mim de bom profissional e foi uma honra ter tido a oportunidade de trabalhar com ele;

Ao Eduardo Ogasawara por sempre se disponibilizar para ajudar e pelas contribuições dadas ao longo desse trabalho.

Aos meus amigos que me auxiliaram durante esse processo, em especial ao Wendel por ter contribuído com seus conhecimentos sobre otimização;

Aos revisores de artigos, que de alguma forma contribuíram para a melhoria desse trabalho. E agradeço também a todos os funcionários da COPPE que sempre me ajudaram nas questões administrativas.

Agradeço.

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

UM SERVIÇO DE CUSTO PARA A EXECUÇÃO PARALELA DE *WORKFLOWS* CIENTÍFICOS EM NUVENS DE COMPUTADORES

Vitor de Gamboa Viana

Outubro/2012

Orientador: Marta Lima de Queirós Mattoso

Programa: Engenharia de Sistemas e Computação

O conceito de computação em nuvem vem se firmando como um novo modelo de computação que proporciona aos cientistas uma oportunidade de se utilizar diversos recursos distribuídos para a execução de experimentos científicos. Muitos dos experimentos científicos existentes, modelados como *workflows* científicos, devem controlar a execução de atividades que consomem e produzem grandes volumes de dados. Há uma demanda por paralelismo e ambientes de alto desempenho na execução destes experimentos, uma vez que muitas destas atividades são computacionalmente intensivas. Entretanto, paralelizar um *workflow* científico em um ambiente de nuvem não é uma tarefa trivial. Uma das tarefas mais complexas é definir a melhor configuração possível do ambiente de nuvem, *i.e.*, o número ideal de máquinas virtuais a serem utilizadas e como projetar a estratégia de execução paralela. Devido ao grande número de opções para configurar o ambiente de nuvem, esta tarefa de configuração se torna inviável de ser executada manualmente e, caso não seja realizada da melhor forma possível, pode produzir impactos negativos de desempenho, ou aumento excessivo no custo financeiro da execução. Este trabalho propõe o SciCumulus-ECM (SciCumulus *Environment Cost Model*), um serviço baseado em um modelo de custo, que determina a melhor configuração possível para o ambiente de acordo com restrições impostas pelos cientistas, através de um otimizador baseado em algoritmo genético.

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

A SERVICE OF COST FOR PARALLEL EXECUTION OF SCIENTIFIC *WORKFLOWS* ON COMPUTE CLOUDS

Vitor de Gamboa Viana

October/2012

Advisor: Marta Lima de Queirós Mattoso

Department: Systems and Computer Engineering

The concept of cloud computing has established itself as a new computing model that provides scientists an opportunity to use multiple distributed resources to perform scientific experiments. Many of the existing scientific experiments, modeled as scientific workflows, shall monitor the implementation of activities that consume and produce large volumes of data. There is a demand for parallelism and high performance environments in carrying out these experiments since many of these activities are computationally intensive. However, parallelizing a scientific workflow in a cloud environment is not a trivial task. One of the hardest tasks is to define the best possible configuration of the cloud environment, i.e., the optimal number of virtual machines to be used and how to design a strategy for parallel execution. Due to the large number of options to configure the cloud environment, this configuration task becomes impossible to be performed manually, and if not done in the best possible way, can produce negative impacts on performance, or excessive increase in the financial cost of running. This paper proposes the SciCumulus-ECM (SciCumulus Environment Cost Model), a service based on a cost model to determine the best possible configuration for the environment in accordance with restrictions imposed by the scientists, using a genetic algorithm-based optimizer.

Índice

Índice de Figuras

Capítulo 1 - Introdução

Com a evolução da ciência da computação, os cientistas vêm utilizando diversos programas para a execução de seus experimentos (Hey, Tansley, e Tolle 2009). Este conjunto de programas pode ser modelado como um *workflow* científico. Desta forma, o cientista enxerga todo o conjunto de atividades como sendo um único "objeto" que produz um resultado final concreto (M. Mattoso et al. 2008). Em um *workflow* científico, os dados produzidos por uma atividade são transferidos para a atividade seguinte como dado de entrada. Este fluxo de dados pode ser gerenciado de um modo manual, mas é mais adequado que seja gerenciado por complexos mecanismos, chamados Sistemas de Gerência de *Workflows* Científicos (SGWfC) (I. Taylor *et al.* 2007).

Os experimentos científicos modelados como *workflows* científicos são caracterizados pela composição de algumas variações de *workflows* dentro de um mesmo experimento (Marta Mattoso *et al.* 2010). Em muitos casos, os programas utilizados nos *workflows* são computacionalmente intensivos e necessitam de técnicas de paralelismo e ambientes de processamento paralelo. O ambiente de nuvem pode ser uma opção para prover essa capacidade de alto desempenho, atuando como um ambiente para processamento paralelo.

A computação em nuvem (do inglês *Cloud Computing*) vem sendo utilizada por cientistas, para a execução de seus experimentos, devido à facilidade de acesso a recursos caros, como *software* e infraestrutura computacional (L. Wang *et al.* 2006). A computação em nuvem oferece uma grande gama de recursos, que são disponibilizados por meio de serviços *Web*, e que podem ser dinamicamente reconfigurados para se ajustar a uma demanda variável (o que é conhecido como elasticidade (Vaquero et al. 2009)). Dessa forma, diferentes tipos de usuários (ou no contexto dessa dissertação – cientistas) obtêm uma grande variedade de recursos. Diferentemente de usuários de *clusters* e grades computacionais, os usuários da nuvem se beneficiam justamente do conceito de elasticidade, uma vez que eles podem facilmente dimensionar a quantidade de recursos utilizados para mais, ou para menos, de acordo com sua demanda de processamento, de forma teoricamente ilimitada (Vaquero *et al.* 2009). A utilização desses serviços é tipicamente explorada por um modelo *pague pelo uso* (Vaquero *et al.* 2009),onde o usuário paga apenas pelo tempo (ou volume de dados) utilizado. E é

fortemente baseada no conceito de virtualização (Barham et al. 2003) em que máquinas virtuais (MV) desempenham um papel fundamental, criando ambientes isolados e independentes implementados sobre uma mesma máquina física. Apesar dos provedores de nuvem explorarem seus serviços de acordo com três modelos fundamentais: Infraestrutura como Serviço, do inglês *Infrastructure as a service* (IaaS), Plataforma com Serviço, do inglês *platform as a service* (PaaS) e *Software* como Serviço, do inglês *software as a service* (SaaS), apenas o modelo IaaS está no escopo dessa dissertação.

Entretanto, executar um *workflow* em paralelo em um ambiente de alto desempenho não é uma tarefa trivial, especialmente em um ambiente "novo" e incipiente como as nuvens de computadores. Para resolver essa questão, em 2008, o paradigma da computação de muitas tarefas (CMT ou MTC – do inglês *Many Task Computing*) foi proposto (Raicu, I. T. Foster, e Yong Zhao 2008). Este paradigma consiste em definir políticas e mecanismos para que alguns recursos computacionais sejam usados, em curtos períodos de tempo, para realização de diferentes tarefas computacionais. Duas das técnicas de paralelismo exploradas pelos SGWfCs, atualmente, são a variação de parâmetros (ou exploração de parâmetros) e a fragmentação de dados (Eduardo Ogasawara *et al.* 2009). Alguns SGWfC já exploram paralelismo de *workflows* utilizando CMT em ambientes como *clusters* (Zhao *et al.* 2007) e grades (I. Foster e C. Kesselman 2004).

Todavia, o gerenciamento da execução paralela de *workflows* em ambientes heterogêneos e dinâmicos, como as nuvens computacionais, não se encontra disponível nos principais SGWfC da atualidade. Como uma alternativa para essa questão, surgiu o SciCumulus (Daniel Oliveira *et al.* 2010), que é um motor de execução projetado para distribuir, controlar e monitorar execução paralela de atividades de *workflows* científicos (ou mesmo *workflows* científicos inteiros) disparadas a partir de um SGWfC em um ambiente na nuvem, como a Amazon EC2 (Amazon EC2 2010).

Apesar do SciCumulus e outras abordagens existentes auxiliarem na execução paralela de atividades de *workflows* científicos, em nuvens de computadores, existem algumas questões ainda em aberto. Por exemplo, em *clusters* e grades computacionais, os recursos são físicos e limitados. O cientista tem apenas aqueles recursos à disposição e pode agendar execuções utilizando esses recursos. Ou seja, os recursos disponibilizados são fixos e o usuário não tem a possibilidade de aumentar ou reduzir a quantidade de memória, ou o número de processadores a serem utilizados, por exemplo.

Em contrapartida, as nuvens computacionais, pela sua característica de elasticidade, permitem que o usuário varie, para mais, ou para menos, a quantidade de recursos alocados para a execução de determinada tarefa. Além disso, no ambiente de nuvem, não existe recurso criado *a priori.* Sempre que um cientista necessitar executar um *workflow,* o ambiente completo deve ser criado. Para tal, as máquinas virtuais devem ser instanciadas sob demanda no momento da execução do *workflow*. Mas essa instanciação sob demanda leva a novos problemas e questões em aberto. Cada provedor de nuvem disponibiliza uma grande gama de tipos de máquinas virtuais com diferentes capacidades e preços. O usuário passa a enfrentar o problema de dimensionar o ambiente para a sua execução do *workflow*. Qual tipo de máquina virtual utilizar? Quantas máquinas instanciar? Como otimizar a utilização dos ambientes de nuvens computacionais, visto que a utilização desses serviços é tipicamente explorada por um modelo *pague pelo uso*? Assim sendo, para se otimizar a utilização dos ambientes de nuvem, sem gastar mais recursos financeiros do que o desejado, nem levar mais tempo do que o programado para a execução do *workflow*, é primordial que se instancie um número ótimo, ou o mais próximo do ótimo, de máquinas virtuais de cada tipo de máquinas virtuais disponibilizado pelo provedor do serviço. Um erro na avaliação do dimensionamento e na configuração do ambiente, onde será executado um *workflow,* pode conduzir a um custo financeiro, ou tempo de execução, muito elevado. Dessa forma, tornando a execução do *workflow* inviável.

1.1 Caracterização do Problema

De acordo com o que foi explicitado anteriormente, e levando em consideração que os ambientes de nuvem cobram pela utilização dos seus serviços por janela de tempo (ou por volume de dados manipulados) e que os cientistas necessitam, muitas vezes, de uma resposta rápida para seus experimentos, o problema que essa dissertação busca solucionar é:

"*Como dimensionar a quantidade de recursos a ser instanciada, no ambiente de nuvem computacional, considerando os diversos tipos de máquinas virtuais existentes, de modo que sejam atendidos os critérios informados pelos cientistas sobre custo financeiro e tempo máximo de execução, sem que o impacto dessa otimização seja prejudicial à execução do workflow?*"

1.2 Hipótese: Otimização da Configuração do Ambiente de Nuvem Computacional

A hipótese geral dessa dissertação é que **SE** adotarmos um modelo de custo focado na distribuição das atividades de um *workflow* na nuvem e em uma abordagem de otimização baseada em metaheurísticas (*e.g.* algoritmos genéticos), **ENTÃO** podemos melhor dimensionar o ambiente de nuvem de forma a reduzir tempo e custo financeiro da execução de um *workflow* quando comparados a ambientes que não tiveram a quantidade de recursos dimensionada.

Desta forma, essa dissertação propõe uma solução chamada SciCumulus-ECM (SciCumulus *Environment Cost Model*), para determinar o melhor dimensionamento e configuração possíveis de máquinas virtuais, em um ambiente de nuvem computacional. O SciCumulus-ECM é composto por um componente de captura, um modelo de custo e um componente otimizador. O componente de captura coleta informações do ambiente de nuvem de forma independente das execuções dos *workflows* e as armazena em um banco de dados de proveniência (Freire *et al.* 2008) (também localizado na nuvem). O componente otimizador, em conjunto com o modelo de custo, inspirado em modelos de custo para execução de consultas de banco de dados (Elmasri e Navathe 2005) determina uma configuração de ambiente adequada para a execução do *workflow*, de acordo com as restrições informadas pelos cientistas. Desta forma, o objetivo dessa dissertação é preparar todo o ambiente de acordo com a demanda de processamento do *workflow* e do cientista, antes de um mediador escalonar as atividades do *workflow* na nuvem, para sua execução paralela. O modelo de custo foi elaborado para ser orientado a métricas de Qualidade de Serviço (do inglês QoS – *Quality of Service*) (Costa e Furtado 2009), o que significa que seu objetivo é aumentar a satisfação dos cientistas, mantendo um bom nível de QoS. A contribuição apresentada, nessa dissertação, visa explorar a elasticidade presente nos ambientes de nuvem. Porém, outros sistemas de alto desempenho, caso possuam algum grau de flexibilidade, também podem se beneficiar.

1.3 Organização da Dissertação

Além desse capítulo introdutório, essa dissertação está organizada em outros seis capítulos. O Capítulo 2 apresenta os trabalhos relacionados. O Capítulo 3 apresenta conceitos relacionados à computação em nuvem, *workflows* científicos, otimização e algoritmos genéticos. O Capítulo 4 apresenta o SciCumulus, uma abordagem para a execução de *workflows* desenvolvida para explorar paralelismo de dados e de parâmetros em larga escala em ambientes de nuvens computacionais. O Capítulo 5, apresenta a abordagem contida nessa dissertação, o SciCumulus-ECM, detalhando o componente de captura de informações da nuvem computacional, além do modelo de custo proposto e o componente otimizador. O Capítulo 6 apresenta a avaliação experimental da abordagem proposta, contemplando a avaliação do modelo de custo e do algoritmo genético utilizado para a otimização do ambiente. Finalmente, o Capítulo 7 conclui a dissertação apresentando os principais resultados alcançados e as possibilidades de trabalhos futuros.

Capítulo 2 - Trabalhos Relacionados

Baseados nos artigos que foram analisados, estruturamos este capítulo em três seções principais. Na Seção 2.1, são apresentados os trabalhos existentes sobre QoS e modelos de custos. A Seção 2.2 discute sobre os algoritmos existentes para escalonamento de tarefas em nuvens e grades que se baseiam em modelos de custo multiobjetivo. E por fim, a Seção 2.3 discute sobre algoritmos genéticos e sua relação com o escalonamento de tarefas.

2.1 *QoS* e Modelos de Custo

Singh, Carl Kesselman, e Deelman (2007) propõem um modelo de custo, onde os nós de uma grade devem anunciar a disponibilidade de recursos para a comunidade de usuários. Além disso, apresentam um algoritmo para selecionar o conjunto de recursos a ser provido que, de forma semelhante a este trabalho, otimiza o desempenho das aplicações, minimizando os custos relacionados aos recursos. Difere do trabalho apresentado nesta dissertação, pois não tem a intenção de configurar ambientes distribuídos. E nesse trabalho, os provedores não necessitam anunciar os recursos disponíveis, mas devem possuir um serviço capaz de disponibilizar essas informações.

O termo "Qualidade de Serviço" (do inglês *Quality of Service* - QoS) (Costa e Furtado 2009) refere-se a todas as características não funcionais de um serviço (no nosso caso, a execução paralela de um *workflow* na nuvem) que pode ser usado por um cliente (ou um cientista, no contexto deste trabalho) para avaliar a qualidade do serviço. Importantes aspectos são: (i) o Desempenho, (ii) Confiabilidade (iii) Capacidade e (iv) Custo. No entanto, nesta dissertação, os estudos ficaram concentrados apenas no desempenho e custo.

Costa e Furtado (2008) analisam as arquiteturas centralizada e hierárquica de acordo com as restrições de *throughput* e QoS para serviços distribuídos, de forma semelhante a este artigo. Sobre as restrições de QoS, os autores afirmam que as limitações de tempo estão entre os critérios mais utilizados de QoS. Embora os autores usem restrições de QoS em um modelo de custo, seu objetivo é diferente do que os

apresentados neste trabalho, porque eles não têm a intenção de configurar ambientes distribuídos.

Owonibi e Baumann (2010) contribuem com um modelo de custo para processamento distribuído. O que inclui determinar os diferentes custos de execuções de operações de processamento de dados, em diferentes servidores, assim como os custos de transferência de dados, custos de entrada e saída de dados, além dos custos relacionados à codificação decodificação. Embora utilizem o ambiente de nuvens computacionais no trabalho, não analisam restrições de QoS e também não incluem o custo monetário, que é importantíssimo para essa dissertação, no modelo de custo. Além de não estar no escopo do artigo fazer configuração do ambiente.

Nguyen *et al.* (2012) contribuem com novos modelos de custo para otimização de consultas em banco de dados em ambientes de nuvem. O critério para otimizar é, de fato, pelo menos, bidimensional, com o custo monetário para utilização da nuvem e o tempo de resposta da consulta. Os autores complementam os modelos de custo existentes com um componente de custo monetário que é primordial em nuvens computacionais. Porém, apesar de levar em consideração o custo monetário, visam a otimização de consultas, ao invés da configuração do ambiente.

2.2 Escalonamento

Segundo Gounaris *et al.* (2004), no contexto do processamento de consultas de banco de dados, técnicas de paralelização existentes não podem funcionar bem em ambientes de grade, porque o modo de selecionar máquinas e atribuir tarefas compromete o paralelismo particionado. Os autores propõem um algoritmo de escalonamento de baixa complexidade, que permite a utilização de paralelismo particionado em consultas, a fim de alcançar um melhor desempenho em um ambiente de grade. Entretanto, apesar de utilizarem um algoritmo para escolher quais máquinas utilizar, o ambiente já está todo configurado e não levam em consideração o custo monetário, nem restrições de qualidade.

Garg, Buyya, e Siegel (2009) apresentam duas novas heurísticas para escalonar aplicações paralelas em grades que gerenciam e otimizam restrições de tempo e de custo. Porém, diferem do trabalho apresentado nessa dissertação, pois não tem como objetivo a configuração o ambiente.

O Quincy (Isard *et al.* 2009) é um *framework* para escalonar tarefas concorrentes e distribuídas, abordando o problema do agendamento de trabalhos simultâneos em *clusters,* onde os dados de aplicação são armazenado nos nós, e precisam ficar pertos de onde são processados. Tratam o problema de escalonamento com restrições de localização e justiça. O problema de escalonamento é mapeado para uma estrutura de dados gráfica, onde os pesos das arestas e capacidades definem suas características, como as demandas de dados locais e justiça. Um solucionador calcula o esquema ideal online de acordo com um modelo de custo global. Porém, por ser em *cluster*, não existe a preocupação de configuração do ambiente e não analisam as restrições de custo monetário.

Shah e Mahadik (2009) discutem sobre algoritmos de escalonamento de recursos em grades computacionais, a partir de diferentes pontos de vista, como as políticas estáticas *versus* dinâmicas, QoS, estratégias que lidam com o comportamento dinâmico de recursos e assim por diante. E discutem também como a economia e a qualidade de serviço desempenham papel importante no agendamento de recursos. Por ser baseado em ambiente de grades, mais uma vez não existe a preocupação com o custo monetário, nem a preocupação com a configuração do ambiente.

Mani e Rao (2011) se concentram em explorar a natureza dinâmica dos preços de energia elétrica, de modo que uma redução de custos é obtida pela localização geográfica dos pedidos de servidores que operam a custos monetários mais baixos em determinadas épocas do ano. Decisões de escalonamento são feitas considerando as cargas e os custos operacionais dos servidores, ou seja, os pedidos são programados para rodar em servidores que operam a baixo custo, que também tem carga esperada baixa. Apesar de o escopo ser um pouco diferente do apresentado nessa dissertação, esse trabalho tem em comum as restrições de orçamento utilizadas em um contexto de configuração de ambiente. Porém, nessa dissertação, é utilizada a restrição de tempo de resposta no modelo de custo. E a configuração é de um ambiente de nuvem computacional.

Dias *et al.* (2011) propõem o Heracles, uma abordagem para execução de *workflows* baseados em técnicas *peer-to-peer (P2P)*. Esta abordagem propõe um mecanismo de tolerância a falhas e gerenciamento dinâmico de recursos inspirados em técnicas de *P2P*. O objetivo do Heracles é executar as atividades em paralelo, sem pedir aos cientistas para especificar o número de nós envolvidos na execução, e

automaticamente reagendar tarefas que falharam. Desta forma, os cientistas só precisam definir o prazo para o *workflow*. Semelhante a este trabalho, Dias *et al.* focam em atender restrições de cientistas, porém não tem como objetivo a configuração do ambiente, já que focam na execução em *clusters* que já são ambientes com configuração definida.

2.3 Algoritmos Genéticos

Dutta e Joshi (2011) discutem estratégias de agendamento de tarefas em ambiente de nuvens computacionais, satisfazendo requisitos de qualidade. Um algoritmo genético é proposto para calcular o melhor custo de agendamento de tarefas baseada em múltiplos QoS, em um tempo finito. Difere do trabalho apresentado nessa dissertação, pois o algoritmo genético é utilizado para agendar tarefas, ao invés de configurar o ambiente.

Raisanen e Whitaker (2005) apresentam um estudo sobre o problema de colocação da antena e algoritmos genéticos. O problema de colocação da antena, ou problema de planejamento celular, envolve a localização e configuração de infraestrutura para redes celulares sem fios. A partir da localização dos sítios candidatos, um conjunto precisa ser selecionado em função de objetivos relacionados com questões como custo financeiro e prestação de serviços. Apesar de ter em comum as restrições de custo monetário utilizadas em um contexto de configuração de ambiente, difere do trabalho apresentado nessa dissertação, pois não utilizam a restrição de tempo de resposta no modelo de custo. E a configuração de ambiente não é de um ambiente de nuvem computacional.

He *et al.* (2011) consideram o cenário onde vários *clusters* de máquinas virtuais (denominados como *clusters* virtuais) estão hospedados em um ambiente de nuvem que consiste de um cluster de nós físicos. Vários *clusters* virtuais convivem no *cluster* físico, com cada *cluster* virtual oferecendo um determinado tipo de serviço para os pedidos recebidos. Os autores utilizam algoritmo genético para adaptar o número mínimo de nós para acomodar todas as máquinas virtuais no sistema, o que desempenha um papel importante na economia de consumo de recursos. Como é baseado em um ambiente de *cluster*, esse trabalho diferente do trabalho apresentado

nessa dissertação, pois não tem como objetivo configurar o ambiente, nem utiliza as restrições de custo monetário.

Macías e Guitart (2011) lidam com o problema de oferecer preços competitivos na negociação de serviços nos mercados de nuvens computacionais. Uma abordagem baseada em algoritmos genéticos é proposta, no qual uma função de preço oferece os mais adequados em função do estado do sistema. Os autores demonstram a capacidade de adaptação de algoritmos genéticos para os preços nos mercados de nuvens computacionais. Esse trabalho foi citado por analisar ambientes de nuvens computacionais, e assim, considerar o custo monetário envolvidos na prestação de serviços. Porém tem um escopo diferente do apresentado nessa dissertação, pois não visam configuração de ambientes, e sim calcular um bom valor de mercado para o serviço.

Capítulo 3 - *Workflows* Científicos e Nuvens Computacionais

Esse capítulo tem como objetivo elucidar os conceitos relacionados ao tema dessa dissertação. Esse capítulo está organizado da seguinte forma: A Seção 3.1 define experimento científico, enquanto que a Seção 3.2 apresenta o conceito de ciclo de vida deste tipo de experimento. A Seção 3.3 apresenta o conceito de *workflows* científicos. A Seção 3.4 apresenta o conceito de computação em nuvem, que é primordial para o completo entendimento da dissertação e a Seção 3.5 discute sobre o ciclo de vida de um *workflow* neste tipo de ambiente. A Seção 3.6 discute sobre problemas de otimização. E a Seção 3.7 apresenta algoritmos genéticos, relacionando-os aos problemas de otimização.

3.1 Experimentos Científicos Baseados em Simulação

Experimentos científicos baseados em simulação, ou simplificadamente experimentos científicos, são aqueles em que os objetos em estudo são representados por modelos computacionais (G. H. Travassos e Barros 2003). Os experimentos científicos são a maior preocupação da comunidade científica (Mattoso *et al.* 2010).

Segundo Soanes e Stevenson (2003), 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". 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). Nesta definição, o termo “condições controladas” é utilizado para indicar que há esforços para eliminar, ou pelo menos reduzir, o máximo possível, os erros ocasionais durante uma observação planejada (Juristo e Moreno 2010). Baseado nessas definições, pode-se concluir que um experimento científico está estritamente associado a um conjunto de ações controladas. Estas ações controladas incluem variações de testes, e seus resultados são, na maioria das vezes, comparados entre si com o objetivo de aceitar ou refutar uma hipótese científica.

Estes tipos de experimentos são comuns em diversas áreas de pesquisa como, por exemplo, análises filogenéticas (Ocaña, Daniel Oliveira, Eduardo Ogasawara, *et al.* 2011), genômica comparativa (Ocaña, Daniel Oliveira, Dias, *et al.* 2011), 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; Ester C. Sabino *et al.* 2011), prospecção de petróleo em águas profundas (Carvalho 2009; W. Martinho *et al.* 2009; Eduardo Ogasawara *et al.* 2011; D. Oliveira *et al.* 2009), mapeamento dos corpos celestes (Hey, Tansley, e Tolle 2009), ecologia (Hartman *et al.* 2010), agricultura (Fileto *et al.* 2003), busca de genes ortólogos dos tripanosomas causadores de doenças tropicais negligenciadas (F. Coutinho *et al.* 2011; Dávila *et al.* 2008), dinâmica de fluidos computacional (G. M. Guerra e F. A. Rochinha 2009a, 2009b, 2010; G. Guerra *et al.* 2009, 2012; Lins *et al.* 2009), estudos fisiológicos (Porto *et al.* 2011), previsão de precipitação (Evsukoff, B. Lima, e N. Ebecken 2011), monitoramento aquático (G. C. Pereira e N. F. F. Ebecken 2011) e pesquisa sobre energia escura (Governato *et al.* 2010). Grande parte desses exemplos pode ser considerada de larga escala pelo amplo volume de dados consumidos e produzidos, que demanda muitos recursos computacionais.

Como experimentos em larga escala, como os citados anteriormente, necessitam de muitas horas de processamento em ambientes de processamento de alto desempenho (PAD), estes são normalmente executados em ambientes de *clusters* e supercomputadores, grades computacionais, ambientes de computação voluntária e mais recentemente as nuvens de computadores. Outra característica desses experimentos é a alta complexidade envolvida em seu gerenciamento devido à 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. Este gerenciamento torna-se ainda mais complicado quando há a preocupação de registro da execução para análise posterior e para tentar sua reprodutibilidade (Davidson e Freire 2008). Esse registro é uma característica fundamental para que um experimento seja considerado verdadeiramente "científico".

Em grande parte dos casos, esses experimentos são representados através do encadeamento de múltiplas combinações de programas que podem consumir grandes quantidades de dados, e são distribuídos em ambientes computacionais heterogêneos. Nesses experimentos, cada programa pode ser executado consumindo um grupo

específico de parâmetros e dados, cada qual com a sua própria semântica e sintaxe. A saída de um programa é normalmente utilizada como entrada para outro programa no encadeamento (M. C. Cavalcanti *et al.* 2005).

Os *workflows* científicos são uma alternativa interessante para representar os encadeamentos de programas, ao invés de usarmos uma abordagem *ad hoc* manual ou baseada em *scripts*. Os *workflows* científicos vêm sendo utilizados como uma abstração para modelar o fluxo de atividades e de dados. Nos *workflows* científicos, estas atividades são geralmente programas ou serviços que representam algum algoritmo ou método que deve ser aplicado ao longo do processo de experimentação (Barker e van Hemert 2008).

Estes *workflows* são, usualmente, controlados e executados pelos Sistemas de Gerência de *Workflows* Científicos (SGWfC), que são mecanismos complexos que têm como objetivo apoiar a configuração e execução dos *workflows*. Existem muitos SGWfC disponíveis, atualmente, como o Kepler (Altintas *et al.* 2004), o Taverna (Hull *et al.* 2006), o VisTrails (Callahan *et al.* 2006), entre outros.

A execução de um *workflow* pode ser vista como um conjunto de ações controladas do experimento. Cada execução pode representar uma das tentativas conduzidas no contexto do experimento científico para avaliar uma hipótese. O histórico do experimento é, então, representado por todas as execuções distintas que compõem esse experimento científico. Porém, executá-las é apenas uma das etapas do ciclo de vida do experimento científico em larga escala.

3.2 Ciclo de Vida do Experimento Científico

Ciclo de vida do experimento é caracterizado por um conjunto de tarefas que passam por um processo de iteração, durante um estudo experimental, realizado por cientistas (Deelman *et al.* 2009). Nessa seção, será apresentado 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, que também pode ser definido como múltiplos ciclos que são percorridos pelo cientista várias vezes no curso do experimento. De acordo com Marta Mattoso *et al.* (2010), o processo de experimentação científica, apresentado na Figura 1 - Ciclo de vida

do experimento científico , pode ser resumidamente representado por um ciclo formado por três fases: composição, execução e 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.

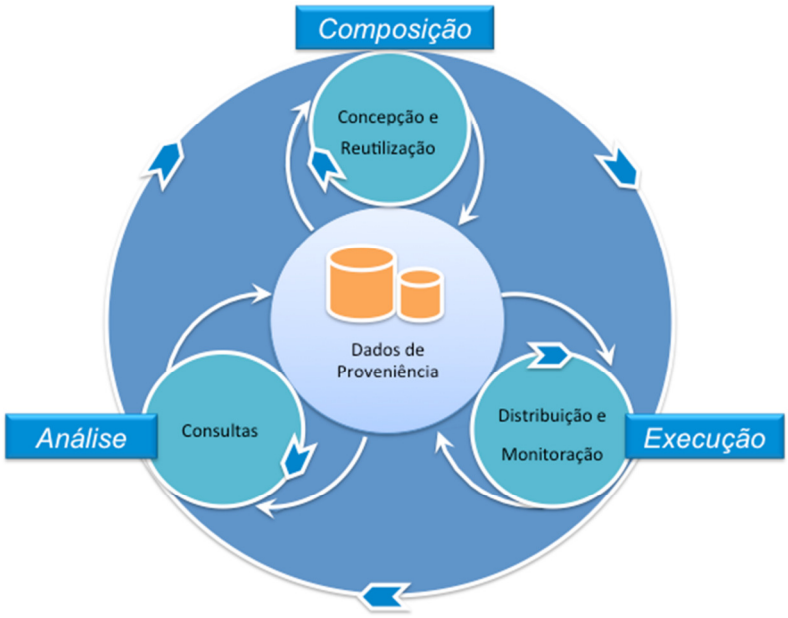

Figura 1 - Ciclo de vida do experimento científico (Mattoso *et al.* 2010)

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, o tipo 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.

A fase de execução é responsável pela materialização do experimento, obtida a partir do resultado da fase de composição, ou seja, por tornar concreta e executável uma especificação de *workflow* de forma que possa ser executada por um SGWfC. Sendo assim, nesta fase, são definidos os dados exatos que serão utilizados como entrada e como valores de parâmetro, através de um 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 engloba as ações relacionadas à necessidade da execução de atividades do *workflow* em

ambientes de PAD, principalmente devido a necessidades de desempenho. A subfase de monitoramento é a fase onde ocorre a verificação do estado atual da execução do *workflow*, uma vez que este pode executar durante um longo período de duração.

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 proveniência (Freire *et al.* 2008), também chamada de histórico do experimento, é essencial para se preservar os dados do experimento, determinar sua qualidade e forma de reprodução. As consultas aos dados de proveniência podem incluir informações tanto da fase de composição quanto da fase de execução do experimento. 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) A hipótese é refutada.

No entanto, esses resultados quase sempre precisam de uma nova execução do *workflow*, para 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 científico.

3.3 *Workflows* Científicos

A definição adotada, na maioria das vezes, para o conceito de *workflow* é a definição dada em *Workflow Management Coalition* (WfMC) (WfMC 2009): "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, nessa dissertação, o foco seja a visão científica do uso de *workflows*, o termo "*workflow*" surgiu, incialmente, 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 documentos em uma empresa, com o objetivo de reduzir custos com impressão e a manipulação (física) de documentos em papel (Mattos *et al.* 2008).

Apesar disso, nos últimos anos, esta tecnologia começou a ser aplicada em diferentes domínios da ciência, como a biologia e a astronomia, que não são comerciais. Áreas que anteriormente executavam seus experimentos manualmente ou através de *scripts* foram naturalmente migrando para a utilização de *workflows* científicos como abstração para especificação de seus experimentos (Hey, Tansley, e Tolle 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 (I. J. Taylor *et al.* 2007).

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

Desta forma, dado $a_i \mid (1 \leq i \leq n)$, consideremos como $I = \{i_1, i_2, .., i_m\}$ o conjunto de dados de entrada para a atividade a_i, então $Input(a_i) \supset I$. Além disso, consideremos O como sendo os dados de saída produzidos por a_i, então $Output(a_i) \supset O$. Uma atividade específica A é modelada como um $a(time)$ onde $time$ é o tempo total de execução de a. A dependência entre duas atividades é modelada como $dep(a_i, a_j, ds) \leftrightarrow \exists\, O_k \in Input(a_j) \mid O_k \in Output(a_i)$ e ds é o volume de dados transferidos entre essas duas atividades (independente da unidade de medida utilizada).

3.4 Computação em Nuvem

Nas últimas décadas, os sistemas distribuídos (Bal, Steiner, e Tanenbaum 1989; Tanenbaum e Renesse 1985) surgiram como uma solução para muitas exigências da indústria de computadores e da comunidade científica. No entanto, a construção de um sistema distribuído se apresenta como uma atividade complexa e na maioria dos casos pode ser considerada uma barreira para criar experimentos mais sofisticados.

A computação em nuvem (do inglês *Cloud Computing*) (W. Kim *et al.* 2009; Marinos e Briscoe 2009; Napper e Bientinesi 2009; Vaquero *et al.* 2009; Lizhe Wang *et al.* 2008) tem se apresentado como um ambiente alternativo de processamento de alto desempenho (PAD), onde serviços acessados através da *Web* permitem que diferentes tipos de usuários obtenham uma grande variedade de recursos virtualizados (baseados em máquinas virtuais – MV), como *software* e *hardware*. Desta forma, a computação, como costumávamos conhecer há alguns anos, mudou completamente. Os programas e os dados passaram dos *desktops* para a nuvem. Os usuários são capazes de acessar programas, documentos e dados de qualquer computador que esteja conectado na internet. Conceitos de elasticidade (Chen, X.-H. Sun, e Wu 2008), disponibilidade (Yu e Vahdat 2006) e segurança (Leusse *et al.* 2008) se tornaram questões chave a serem tratadas.

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

Usuários de centros de supercomputação podem utilizar as nuvens para reduzir o tempo de espera em filas de escalonadores (Armbrust *et al.* 2010). As nuvens permitem utilizar mais máquinas virtuais durante um horário de pico nos centros de dados, ou durante eventos especiais, como experimentos que requerem uma grande quantidade de recursos reservados durante uma janela de tempo maior. Além disso, grandes conjuntos de dados utilizados e gerados pelo experimento podem ser mais facilmente compartilhados para a análise, dentro ou fora da nuvem, democratizando o acesso aos resultados científicos, embora a transferência de dados em nuvens seja um problema em aberto.

Desta forma, as nuvens podem proporcionar um ambiente adequado para a execução paralela de *workflows* científicos que demandem PAD. Entretanto, muitos problemas continuam sem solução. Desta forma, novas propostas são necessárias para fornecer uma solução transparente para os cientistas executarem seus *workflows* em

paralelo em nuvens de forma sistemática. A seguir, de forma a explicar melhor o cenário atual da computação em nuvem, apresentamos uma taxonomia desenvolvida (D. Oliveira, F. Baião, e M. Mattoso 2010) que detalha as principais características das nuvens, além dos padrões utilizados neste ambiente.

Formalmente, um ambiente de nuvem pode ser representado por VM = {VM_0, VM_1, VM_2,…, VM_{n-1}} como o conjunto de *n* MVs disponíveis identificados por um componente de captura de infomações. Para cada máquina VM_i, também considere o conjunto P = {P_0, P_1, P_2, ..., P_{m-1}}, conjunto de *m* programas instalados e configurados por cada máquina VM_i. Outros parâmetros também são considerados e devem ser formalizados:

- Disponibilidade: A disponibilidade de uma MV, denotada por A(VM_i), corresponde à porcentagem de tempo que um tipo de MV está disponível para uso.
- Confiabilidade: definimos confiabilidade de uma MV R(VM_i) como a razão entre o número de solicitações atendidas pela VM_i e o número de pedidos recebidos pela VM_i.
- Custo de Inicialização: o custo de inicialização é estimado analisando o histórico dos tempos de inicialização de cada tipo de MV e usando a configuração da MV.
- Custo de Transferência: os custos necessários para transferir certa quantidade de dados a partir da máquina do cliente ao provedor de serviço.
- Custo de armazenamento: custo baseado no espaço necessário para o armazenamento de informações na nuvem, e
- Custo de Execução: custo de execução de um programa P_j em uma MV denominada VM_i. Este custo inclui o custo desde o início da execução do programa até o seu final, ignorando-se os custos de inicialização e transferência envolvidos.

3.5 Ciclo de Vida do *Workflow* em Nuvem

Um *workflow* científico que é executado em um ambiente de nuvens de computadores segue as mesmas etapas do ciclo de vida de um experimento, porém com

algumas fases adicionais. Podemos definir o ciclo de vida de um *workflow* científico executado em nuvens como sendo composto de sete etapas principais como mostrado na Figura 2.

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

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

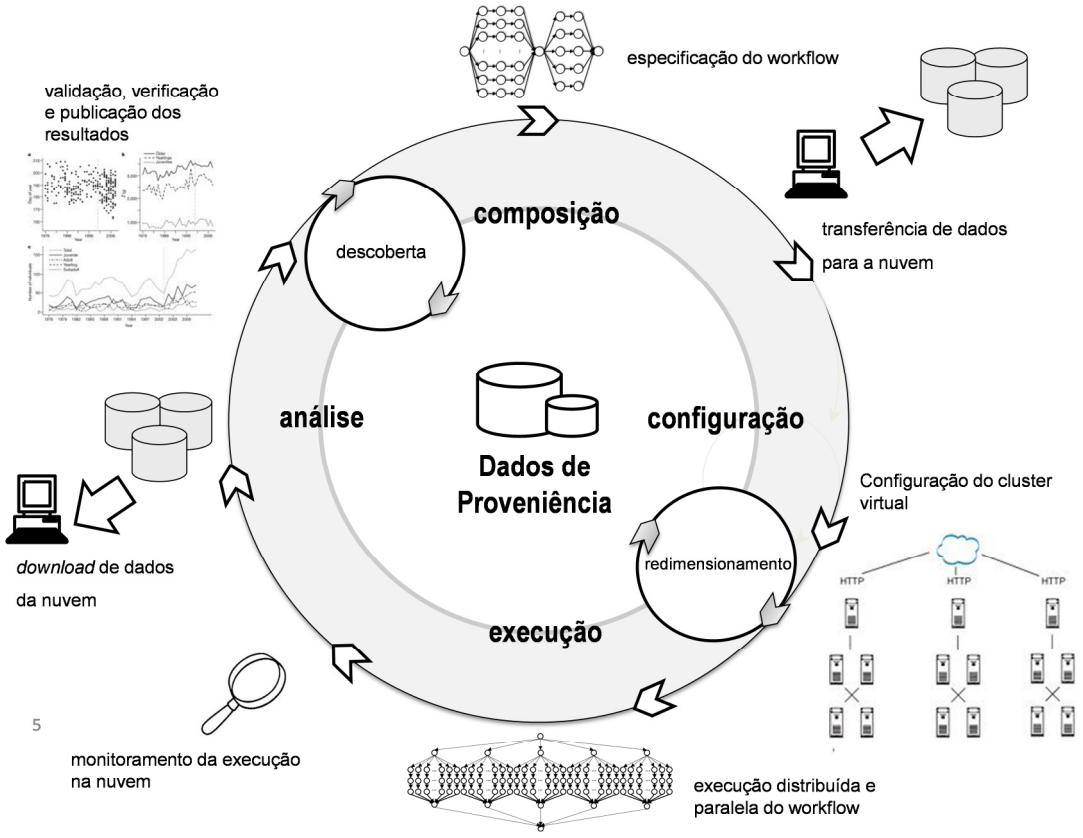

Figura 2 - Ciclo de vida de um *workflow* científico executado em nuvens de computadores (Daniel Oliveira 2012)

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.

3.6 Otimização de Recursos em Nuvens de Computadores

Uma vez que nesta dissertação tratamos de um problema de otimização de recursos, essa seção apresenta uma visão geral dos problemas de otimização. Um problema computacional é descrito informalmente por Garey e Johnson (1979) como uma questão genérica a ser respondida, geralmente possuindo vários parâmetros, ou variáveis livres, cujos valores não são especificados. Um problema é então descrito através de:

(1) uma descrição genérica de todos os seus parâmetros, e;

(2) uma declaração de quais propriedades a resposta ou solução, deve satisfazer.

Através da especificação de valores particulares para todos os parâmetros do problema obtém-se uma instância de um problema.

Para que um problema seja chamado de problema computável, deve-se existir um procedimento efetivo que o resolva em um número finito de passos, ou seja, se existe um algoritmo que leve à sua solução. Porém, um problema considerado "em princípio" computável pode não ser tratável na prática, devido às limitações dos recursos computacionais para executar o algoritmo implementado (Toscani e Veloso 2001). Se existe um algoritmo de tempo polinomial que resolve todas as instâncias de um problema, este problema é tratável, caso contrário diz-se que é intratável.

Um problema de otimização combinatória Π é definido por (Garey e Johnson 1979), como uma terna $\Pi = (D_\Pi, S_\Pi, m_\Pi)$, onde:

(i) D_Π é o conjunto das instâncias;

(ii) S_Π é a função que associa a cada instância $I \in D_\Pi$, um conjunto finito $S_\Pi(I)$ de candidatas a solução para I; e

(iii) m_Π é uma função que atribui a cada instância $I \in D_\Pi$ e a cada candidata $\sigma \in S_\Pi(I)$, um número racional positivo $m_\Pi(I, \sigma)$ chamado valor solução de σ.

Se Π é um problema de minimização [maximização], uma solução ótima para uma instância $I \in D_\Pi$ é uma candidata a solução $\sigma^* \in S_\Pi(I)$ tal que, para todo $\sigma \in S_\Pi(I)$, $m_\Pi(I, \sigma^*) \leq m_\Pi(I, \sigma)$ [$m_\Pi(I, \sigma^*) \geq m_\Pi(I, \sigma)$]. O valor $m_\Pi(I, \sigma^*)$ de uma solução ótima para I é denotado por $OPT_\Pi(I)$.

Os elementos básicos de um problema de otimização são os mesmos de um problema de decisão (aqueles que podem ser respondidos com sim ou não), acrescidos da função m cujo valor mede quão boa é uma solução admissível. O problema consiste em encontrar uma solução com uma medida máxima (no caso de um problema de maximização), ou uma medida mínima (no caso de um problema de minimização).

Para a resolução de problemas de otimização, utiliza-se algoritmos, que correspondem a uma descrição passo a passo de como um problema é solucionável. A descrição deve ser finita e os passos devem ser bem definidos, sem ambiguidades, além de executáveis computacionalmente. Diz-se que um algoritmo resolve um problema p se este algoritmo recebe qualquer instância I de p e sempre produz uma solução para esta instância I.

O objetivo principal dessa dissertação é informar quantas máquinas virtuais de cada tipo disponível devem ser instanciadas, no ambiente de nuvem, de maneira próxima à configuração ideal, respeitando restrições de custo financeiro e tempo de execução. Esse problema se assemelha em partes ao problema do caixeiro viajante (Applegate et al. 2007), onde existem diversos caminhos que devem ser percorridos com o menor custo total, assim como temos diversas configurações possíveis e deve ser escolhida a que produz o menor custo. Além disso, sua complexidade também se assemelha, sendo categorizada como um problema da classe NP (Cormen et al. 2009) (que será explicada a seguir) que o remete para um campo de complexidade exponencial, isto é, o esforço computacional necessário para a sua resolução cresce exponencialmente com o tamanho do problema.

3.6.1 Algoritmos Determinísticos

São algoritmos que apresentam comportamento previsível. Dada uma determinada entrada (instância do problema), o algoritmo apresenta sempre a mesma saída e o mesmo ponto de parada.

3.6.2 Algoritmos Não-Determinísticos

Algoritmos não-determinísticos resolvem o problema ao deduzir os melhores passos através de estimativas sob forma de heurísticas. Ou seja, apresentam comportamento guiado por uma certa distribuição estatística. Dada uma determinada entrada, o algoritmo apresenta uma saída com uma certa probabilidade.

3.6.3 Classes de Complexidade

A Teoria da Complexidade está estruturada matematicamente de modo a classificar problemas computacionais relativamente a sua dificuldade intrínseca.

Nessa seção, são definidas as principais classes de complexidade para problemas de decisão, relativamente à performance de seus algoritmos com respeito a complexidade de tempo. As definições, aqui presentes, foram retiradas de (Garey e Johnson 1979).

A classe P é definida como o conjunto de todos os problemas de decisão resolvíveis por um algoritmo determinístico em tempo polinomial.

A classe NP é definida como o conjunto de todos os problemas de decisão resolvíveis por um algoritmo não-determinístico em tempo polinomial.

Um problema de decisão Π é NP-completo se $\Pi \in NP$ e, para todo problema de decisão $\Pi' \in NP$, tem-se que $\Pi' \propto \Pi$. Portanto, os problemas NP-completos são identificados como os problemas mais difíceis em NP.

A classe NP-completo foi definida como aquela classe que contém os problemas mais difíceis em NP. As técnicas conhecidas para provar a NP-completude podem também serem usadas para provar a dificuldade de problemas não restritos à classe NP. Qualquer problema de decisão Π, pertencente ou não à classe NP, que seja redutível a um problema NP-completo, terá a propriedade de não ser resolvido em tempo polinomial, a menos que P = NP. Neste caso, tal problema Π é chamado NP-difícil, já que ele apresenta dificuldade não menor que qualquer problema NP-completo, definindo desta maneira uma nova classe de problemas: a classe NP-Difícil.

Qualquer problema aberto em NP, ou seja, todo problema que ainda não foi provado estar em P nem em NP-completo, pode ser visto como um candidato à classe NP-Intermediário.

3.7 Algoritmos Genéticos

Algoritmos genéticos (AGs) constituem uma técnica de busca e otimização elaborada a partir do princípio Darwiniano de seleção natural e reprodução genética (D. E. Goldberg 1989). Estes algoritmos pertencem a uma classe particular de algoritmos

evolutivos que usam técnicas inspiradas na biologia evolutiva, como hereditariedade, mutação, seleção natural e recombinação (ou *crossing over*).

Os algoritmos genéticos, geralmente, são aplicados em problemas complexos de otimização (L. D. Davis e Mitchell 1991), tais como problemas que lidam com diversos parâmetros, ou características que precisam ser combinadas em busca da melhor solução; problemas com muitas restrições ou condições; e problemas com grandes espaços de busca.

Em relação aos algoritmos de otimização tradicionais, os algoritmos genéticos tendem a ser menos propensos a convergir para ótimos locais, em detrimento aos ótimos globais. Isso acontece, pois os algoritmos genéticos são considerados altamente paralelos (D. E. Goldberg 1989). Ou seja, eles podem fazer a busca em diferentes áreas do espaço de solução, alocando um número de membros apropriado para a busca em várias regiões.

De forma simples, pode-se explicar o mecanismo dos algoritmos genéticos da seguinte maneira: Inicialmente, é gerada uma população formada por um conjunto aleatório de indivíduos que podem ser vistos como possíveis soluções do problema. Durante o processo evolutivo, esta população é avaliada: para cada indivíduo é dada uma nota, ou índice, refletindo sua habilidade de adaptação a determinado ambiente. Uma porcentagem dos mais adaptados é mantida, enquanto os outros são descartados. Os membros mantidos pela seleção podem sofrer modificações em suas características fundamentais através de mutações e recombinação (*crossover*) genética gerando descendentes para a próxima geração. Este processo é repetido até que uma solução satisfatória seja encontrada.

Os algoritmos genéticos possuem as seguintes operações básicas:

1. Reprodução: O ato de fazer uma cópia de uma possível solução.
2. *Crossover*: O ato de troca de genes entre duas possíveis soluções, simulando o "acasalamento" das duas soluções.
3. Mutação: O ato de aleatoriamente alterar o valor de um gene em uma potencial solução.

Um dos principais elementos dos algoritmos genéticos é a sua função de aptidão (do inglês *Fitness Function*). Para o correto funcionamento do algoritmo, é necessário ser capaz de avaliar o quão "boa" uma solução é, em relação a outras soluções possíveis.

A função de aptidão é responsável por essa avaliação, e retorna um valor de aptidão, geralmente inteiro e positivo. Quanto maior o número, melhor a solução é.

Os valores de aptidão são utilizados em um processo de seleção natural para escolher quais as possíveis soluções continuarão para a próxima geração, e quais não irão continuar. Deve-se notar, no entanto, que durante o processo de seleção natural, as soluções com os maiores valores de aptidão não são necessariamente escolhidas. As soluções são escolhidas estatisticamente, de maneira que é mais provável que uma solução com um valor de aptidão maior seja escolhida, porém a escolha não é garantida. Isto tende a corresponder ao mundo natural.

Capítulo 4 - O SciCumulus

Não há dúvidas de que os ambientes de nuvens computacionais representam um avanço tecnológico. Porém, o gerenciamento de execuções paralelas de *workflows* científicos, nestes ambientes, ainda é uma questão em aberto. Uma infraestrutura específica para a execução paralela destes *workflows*, nas nuvens, se faz necessária. Nesse contexto, uma das soluções propostas, recentemente, para oferecer essa infraestrutura foi o SciCumulus (Daniel Oliveira *et al.* 2010), que é apresentado neste capítulo.

A organização do capítulo segue da seguinte maneira: Na Seção 4.1, é apresentada uma breve descrição da arquitetura conceitual do SciCumulus. Na Seção 4.2, apresentamos os tipos de execução (estática e adaptativa) contempladas no SciCumulus.

4.1 Arquitetura

O SciCumulus é um motor de execução de *workflows* projetado para distribuir, controlar e monitorar execuções paralelas, em ambiente de nuvens computacionais, das atividades de *workflows* científicos oriundas de um Sistema de Gerência de *Workflow* Científico (SGWfC), com a utilização de técnicas de computação de muitas tarefas (CMT). Ele permite que os cientistas sejam isolados de toda a complexidade envolvida na configuração dos ambientes de alto desempenho. No caso específico, um ambiente de nuvem, como o Amazon EC2 (Amazon EC2 2010). Como o ambiente de nuvem é distribuído e cada máquina virtual é independente da outra, deve-se encapsular em uma única unidade de trabalho o programa a ser executado, os dados, os valores de parâmetros e a estratégia de paralelismo que será executada. Esta unidade de trabalho recebe o nome de *cloud activity*. O SciCumulus possui uma arquitetura baseada em serviços e que possui quatro camadas:

i. Camada Cliente: esta camada é responsável por realizar a interface entre o ambiente de nuvem e o SGWfC. Seus componentes são instalados nas máquinas dos cientistas e despacham as atividades do *workflow* para serem executadas em paralelo no ambiente de nuvem usando um SGWfC local, como o VisTrails (Callahan *et al.* 2006) ou o Kepler (Altintas *et al*. 2004)
ii. Camada de Distribuição: esta camada cria e gerencia a distribuição de *cloud activities* em uma ou mais máquinas virtuais instanciadas em um ambiente de

nuvem. Seus componentes podem ser instalados em qualquer ambiente, mas preferencialmente na nuvem para diminuir o impacto de comunicação com os componentes da camada de execução;

iii. Camada de Execução: esta camada executa efetivamente uma determinada *cloud activity*. É responsável pela execução de programas encapsulados em *cloud activities* e por coletar dados de proveniência. Seus componentes são instalados nas várias máquinas virtuais envolvidas na execução paralela das *cloud activities*;

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

A camada cliente é responsável por iniciar a execução paralela de atividades de *workflows* na nuvem. Os componentes da camada cliente podem ser instalados em qualquer SGWfC existente, tais como o VisTrails, o Kepler e o Taverna (Hull et al. 2006). Os componentes desta camada devem ser executados na fase de configuração do ciclo de vida do *workflow* executado em nuvem, uma vez que a transferência dos dados e o despacho da atividade para a nuvem são configurações prévias à execução paralela do *workflow*. Os componentes instalados no SGWfC iniciam a execução paralela do *workflow* na nuvem. Há três componentes principais nesta camada: o componente de carga, o componente de despacho, e o componente de *download*.

O componente de carga move os dados de entrada para a nuvem, enquanto que o componente de *download* reúne os resultados finais nas máquinas virtuais e os traz para a máquina do cientista. O componente de despacho cria um arquivo de configuração (que pode ser representado como um arquivo XML), a partir de especificações realizadas pelo cientista, que armazena a configuração para a execução distribuída e inicia o processo de execução paralela das atividades na nuvem se comunicando diretamente com os componentes da camada de distribuição.

A camada de distribuição controla a execução de atividades paralelas em nuvens. A camada é composta por cinco componentes: o intermediário de execução, o explorador de parâmetros, o fragmentador de dados, o encapsulador e o escalonador. Estes componentes são invocados na fase de execução do ciclo de vida do *workflow* executado em nuvens. O intermediário de execução faz a interface entre a camada cliente e a camada de distribuição. 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. O fragmentador de dados atua quando os dados de entrada devem ser fragmentados (de acordo com a definição do cientista), e gera subconjuntos dos dados que podem ser distribuídos ao longo das máquinas virtuais durante a execução juntamente com os valores de parâmetros. 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 gerar problemas de desempenho e, por isso, este modo de trabalho não é indicado. Este problema, no entanto, pode ser dirimido se pudermos estimar de maneira ótima *a priori,* a quantidade de máquinas virtuais a serem utilizadas, que é o objetivo desta dissertação. Por outro lado, o modo adaptativo visa analisar os recursos disponíveis para a nuvem durante o curso da execução do *workflow*, a fim de usar mais (ou menos) máquinas virtuais e adaptar o tamanho do grupo de atividades de acordo com a capacidade de processamento das máquinas virtuais disponíveis e de acordo com as restrições informadas pelo cientista.

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. O controlador de instância faz a interface entre a camada de distribuição e a camada de execução. O configurador do ambiente configura todo o ambiente, definindo as

variáveis de ambiente, e criando diretórios necessários para gerenciar a execução. O executor invoca a aplicação específica e coleta os dados de proveniência, armazenando-os em um repositório também localizado na nuvem (um banco de dados relacional).

Finalmente, a camada de dados contém todos os repositórios de dados utilizados pelo SciCumulus. Ela possui dois componentes principais: o sistema de arquivos compartilhados, e o repositório de proveniência. 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 autônomo que monitora o ambiente a procura de mudanças no mesmo (novas máquinas virtuais disponíveis ou máquinas virtuais destruídas). O sistema de arquivos compartilhado contém todos os dados de entrada consumidos por diversas *cloud activities* durante a execução paralela.

Os componentes supracitados fornecem o arcabouço conceitual mínimo para a execução paralela de *workflows* científicos em nuvens computacionais. Entretanto, existem otimizações que podem ser desenvolvidas e necessidades importantes dos cientistas que não são atendidas com esta arquitetura básica. Uma dessas otimizações é justamente o dimensionamento do ambiente que é tratado nesta dissertação. Embora o SciCumulus possua quatro camadas com diversos componentes, a abordagem proposta nesta dissertação está no escopo da camada de distribuição.

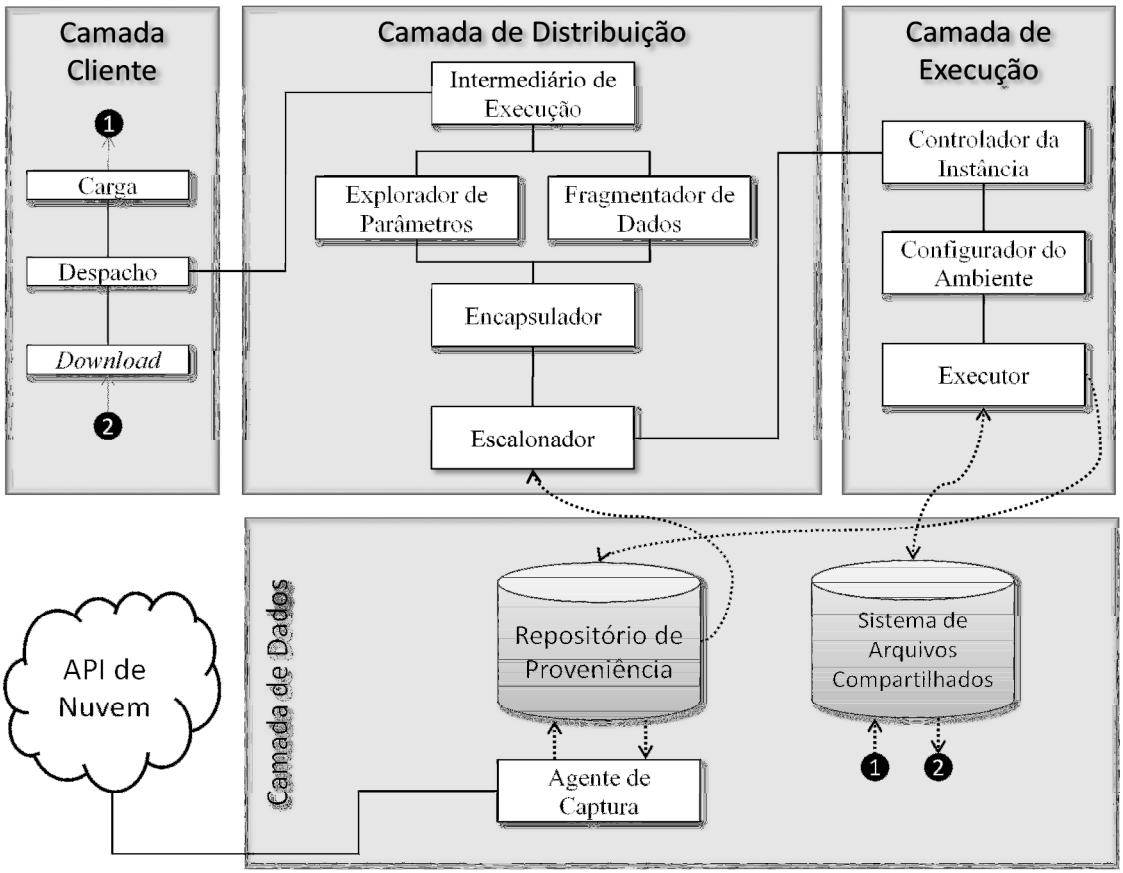

Figura 3 - Arquitetura conceitual SciCumulus

O SciCumulus apoia dois tipos de paralelismo para serem utilizados pelos cientistas: Varredura de parâmetros e paralelismo de dados (Samples *et al.* 2005).

Supondo que um *workflow* seja formado por quatro conjuntos: um conjunto *A* de atividades para serem executadas, um conjunto *D* de dados de entrada para serem consumidos, um conjunto *P* de parâmetros e um conjunto *S* de saída do *workflow*, pode-se caracterizar os tipos de paralelismo como a seguir.

Uma varredura de parâmetros é definida como a execução de uma determinada atividade do *workflow W*, com a utilização de um subconjunto dos parâmetros de *P*, e simultaneamente, a mesma atividade seria executada, por outra máquina virtual, com a utilização de outros valores de parâmetros contidos em *P*.

O paralelismo de dados pode ser definido como a execução de uma determinada atividade do *workflow W*, com a utilização de um subconjunto dos dados de entrada *D*, e simultaneamente, a mesma atividade seria executada, em outra MV, utilizando, desta vez, outros dados de entrada contidos em *D*.

4.2 Execução Estática x Execução adaptativa

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 gerar problemas de desempenho, como mencionado anteriormente (este problema pode ser dirimido se pudermos estimar *a priori*, a quantidade de máquinas virtuais a serem utilizadas). Entretanto, este modo de trabalho do escalonador não é indicado.

Por outro lado, o modo adaptativo visa analisar os recursos disponíveis para a nuvem durante o curso da execução do *workflow*, a fim de usar mais (ou menos) máquinas virtuais e adaptar o tamanho do grupo de atividades de acordo com a capacidade de processamento das máquinas virtuais disponíveis. Baseado no escalonamento escolhido, o escalonador transfere as *cloud activities* para as máquinas virtuais disponíveis e controla a sua execução. Além disso, pode também combinar os resultados parciais (se necessário) para organizar os resultados finais a serem transferidos para a camada cliente (que está instanciada no SGWfC). Mais detalhes sobre os modos de execução do SciCumulus se encontram em (Oliveira 2012).

Capítulo 5 - O SciCumulus-ECM

Conforme vimos na última seção, já existem soluções que apoiam a execução paralela de *workflows* em nuvens de computadores. Entretanto, as abordagens existentes ainda não conseguem dimensionar de fato o ambiente de nuvem antes da execução do *workflow*. Essa é de fato uma tarefa muito complicada, pois existem diversos tipos de recursos para utilização. Por exemplo, a Amazon EC2 (Amazon EC2 2010) fornece pelo menos 10 diferentes tipos de máquinas virtuais que podem ser instanciadas para utilização. A escolha do tipo de máquina virtual para ser instanciada na execução paralela de um *workflow* científico pode impactar severamente no desempenho do mesmo.

Para que esse tipo de dimensionamento seja possível, deve-se inicialmente capturar informações fundamentais do ambiente de nuvem, catalogar recursos com funcionalidades equivalentes que podem ser selecionados, e aplicar um modelo de custo a fim de determinar o dimensionamento ideal do ambiente. É importante ressaltar que este problema é inerente às nuvens, pois sua configuração é dinâmica. Esse capítulo apresenta o SciCumulus-ECM, a abordagem de dimensionamento de recursos de nuvem que é composta pelo componente de captura de informações da nuvem, o modelo de proveniência implementado, o modelo de custo de dois critérios, com suporte a QoS para nuvens, permitindo melhor uso dos recursos e o cumprimento dos requisitos impostos pelo cientista, e o componente otimizador.

Baseado nos requisitos descritos pelos cientistas acerca da execução do *workflow* (tempo total de execução e custo financeiro de execução, uma vez que os provedores de nuvem cobram por hora de utilização), está sob a responsabilidade de um componente otimizador (a ser acoplado à arquitetura do SciCumulus, ou de qualquer outro motor de execução de *workflows* em nuvens) determinar os recursos da nuvem que serão instanciados, para execução de um determinado *workflow*, em um determinado tempo. Em outras palavras, este componente é o centro da abordagem proposta e é responsável por aperfeiçoar a seleção e instanciação dinâmica de máquinas virtuais antes das atividades serem escalonadas.

As próximas subseções apresentam o modelo de custo, o componente de captura de informações autônomo, o esquema de proveniência, e o componente otimizador de dimensionamento do ambiente. O modelo de custo é utilizado pelo

otimizador, durante a simulação, para a obtenção da melhor configuração possível do ambiente.

5.1 O Componente de Captura de Informações da Nuvem e Modelo de Proveniência

Para que o modelo de custo seja capaz de decidir a melhor configuração do ambiente, levando em consideração os critérios do cientista, com relação ao custo e ao tempo de execução de uma atividade, são necessárias algumas informações sobre os provedores de serviços e sobre o ambiente de nuvem. No contexto dessa dissertação, estas informações são coletadas através de um componente autônomo capaz de fazer uma varredura entre os provedores de serviços disponíveis na nuvem, compatíveis com o componente, e de capturar dados sobre *hardware*, *software* e sobre o próprio provedor. Alguns desses dados são os tipos de máquinas virtuais disponíveis, o preço para utilização do serviço, a capacidade de armazenamento, os sistemas operacionais, as taxas de transferência, entre outras informações.

O trabalho do componente de captura começa a partir do momento que o cientista seleciona um ou mais provedores de serviços na nuvem como elegíveis para executar seu *workflow* científico. A partir desse momento, o componente, por meio de APIs disponibilizadas pelos provedores, consegue os dados necessários. Em sua versão inicial, o componente contempla o provedor Amazon EC2, que foi escolhido devido à estabilidade e confiabilidade. Porém, uma das metas traçadas é que, futuramente, ele esteja configurado para outros provedores, como o GoGrid (2012). O componente coleta os dados relativos ao ambiente, como tipo de máquinas virtuais, características de máquinas virtuais, contas de usuário, etc. e os armazena em um banco de dados relacional de proveniência instanciado no PostgreSQL que também se encontra em uma instância na nuvem. A estrutura do banco de dados utilizado pelo componente é apresentada na Figura 4.

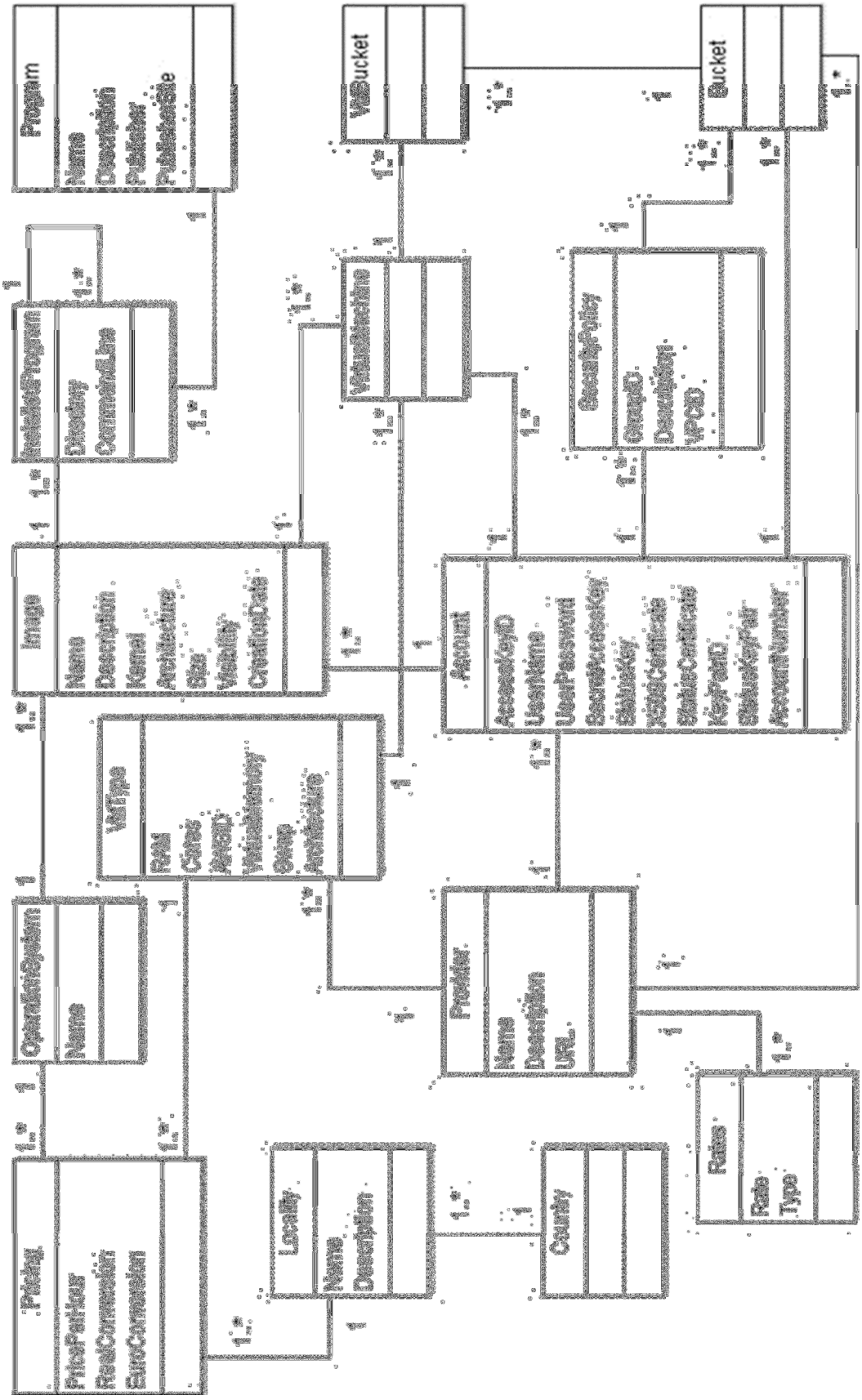

Figura 4 - Modelo de Proveniência de informações do ambiente utilizado pelo SciCumulus-ECM

Para o armazenamento dos dados coletados pelo componente de captura, foi desenvolvido um modelo de proveniência canônico de solução de nuvem capaz de relacionar os diversos tipos de dados de provedores de nuvem e dos ambientes. Dessa forma pode-se obter informações importantes para o dimensionamento do ambiente e o escalonamento de atividades *a posteriori*. De posse dessas informações, fica a cargo do otimizador analisá-las e decidir, baseado no modelo de custo descrito a seguir, qual a melhor configuração de máquinas virtuais dentre os critérios escolhidos pelo cientista. As principais classes do modelo são *Image*, *VMType*, *Provider* e *Pricing*. A classe *Image* armazena os dados sobre as imagens disponíveis no provedor de nuvem, tais como nome, descrição, arquitetura, etc. A partir destas imagens, pode-se instanciar as máquinas virtuais necessárias. A classe *VMType* armazena os dados referentes aos tipos existentes de máquinas virtuais e suas características, como memória RAM, número de *cores* do processador, entre outras características. A classe *Provider* é onde ficam as informações sobre o provedor de serviços na nuvem, no caso unicamente a Amazon. E a classe *Pricing* é onde são registrados os dados sobre os custos financeiros para a utilização dos serviços do provedor.

5.2 O Modelo de Custo Proposto

Um modelo de custo consiste em um conjunto de fórmulas que estimam os custos envolvidos em um ambiente computacional particular (Elmasri e Navathe 2006). No nosso caso, o ambiente de nuvem. O modelo de custo apresentado a seguir, foi incorporado no otimizador/dimensionador de recursos do SciCumulus-ECM. O otimizador utiliza o modelo de custo para simular diversas execuções e decidir quando escolher determinado tipo e a quantidade de máquinas virtuais, baseado em um destes dois critérios: tempo de execução e custo financeiro, com a prioridade escolhida pelo cientista. Por se tratar de ambientes distribuídos, o otimizador precisa ser eficiente, uma vez que pode haver elevados custos de comunicação envolvidos na obtenção de informações sobre o ambiente. A seguir, detalhamos cada um dos parâmetros envolvidos no modelo de custo proposto.

Como a nuvem tem seu foco na cobrança, sob demanda, o custo financeiro será o primeiro a ser explicado. O custo financeiro para executar um programa P_j de uma máquina virtual VM_i, denotado por $C_{Ti,j}$ é definido na Equação 1:

$$C_{Ti,j} = C_{Ii} + C_{Ei,j} + C_{Rj} \quad (1)$$

onde C_{Ii} é o custo de inicialização da máquina virtual VM_i, $C_{Ei,j}$ é o custo de execução do programa P_j na máquina virtual VM_i e C_{Rj} é o custo de transferência de dados envolvido na execução do programa P_j. Podemos estender a Equação 1, para que ela abranja também os critérios de qualidade "Disponibilidade" e "Confiabilidade", como definido anteriormente. Assim, a Equação 2, define o custo total da execução do programa P_j na máquina virtual VM_i.

$$C_{Ti,j} = \frac{C_{Ii}+C_{Ei,j}+C_{Rj}}{A(VM_i)\times R(VM_i)} \quad (2)$$

Os valores de $A(VM_i)$ e $R(VM_i)$ são números inteiros entre 0 e 1, e dividindo o tempo total de execução por estes critérios de qualidade, estaremos aumentando o custo estimado para máquinas virtuais menos confiáveis ou constantemente indisponíveis. Esta informação deverá ser fornecida pelo componente de captura de informações ou pelo banco de dados de proveniência para as aplicações clientes que desejam utilizar seus recursos.

Entretanto temos ainda que definir os custos totais referentes à execução do *workflow* como um todo, pois os custos definidos até agora são referentes à execução de uma atividade em uma determinada máquina virtual. O custo total de inicialização, representado pelo somatório dos diversos custos de inicialização de cada máquina virtual. Assim, definimos por $I(VM_i) = C_{Ii}$. Desta forma, podemos simplesmente definir o custo total de inicialização como na Equação 3:

$$C_I = \sum_{i=0}^{n-1} I(VM_i) \quad (3)$$

O custo total de execução é a soma dos custos de execução de cada programa P_j de uma MV específica VM_i, ou seja, $C_{Ei,j}$.

$$C_E = \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} c_{Ei,j} \quad (4)$$

O custo total de transferência de dados pode ser desmembrado em três novos custos e é calculado conforme a Equação 5:

$$C_R = C_U + C_D + C_S \qquad (5)$$

onde C_U é o custo de envio (*upload*), o C_D é o custo de transferência (*download*) e C_S é o custo de armazenamento de dados no provedor.

Todos os custos de transferência de dados, representados na Equação 5, são calculados com base no volume de dados e no preço cobrado pelo provedor para a realização do serviço. Assim, o custo de envio é definido como na Equação 6, sendo v_u o volume de dados enviados para o servidor, para a execução de uma determinada atividade, e ∂_u a taxa cobrada pelo provedor para receber dados.

$$C_U = \sum_{j=0}^{m-1} v_{u_j} \times \partial_u \qquad (6)$$

O custo financeiro de transferência é representado na Equação 7, onde v_d é o volume de dados gerados pela atividade e que deverão ser recuperados do provedor, e ∂_d é a taxa cobrada pelo provedor para enviar os dados:

$$C_D = \sum_{j=0}^{m-1} v_{d_j} \times \partial_d \qquad (7)$$

Para o cálculo do custo financeiro de armazenamento, deve-se somar o volume de dados enviados para o servidor e o volume de dados gerados pela atividade, e por fim, multiplicá-los pela taxa de armazenamento (∂_s) cobrada pelo provedor do serviço. O que resulta na Equação 8:

$$C_S = \sum_{j=0}^{m-1} (v_{u_j} + v_{d_j}) \times \partial_s \qquad (8)$$

Além do custo financeiro, temos que modelar o tempo de execução do *workflow* na nuvem. O tempo total de execução para executar todos os programas no conjunto de máquinas virtuais, denotado por T_T é definido na Equação 9, onde T_{Imax} é o maior tempo de inicialização das máquinas virtuais envolvidas na execução das atividades do *workflow*, T_E é o somatório do tempo de execução de cada programa P_j

em cada VM_i e T_R é o tempo de transferência de dados envolvido na execução de cada programa P_j nas máquinas virtuais.

$$T_T = T_{Imax} + T_E + T_R \qquad (9)$$

O tempo de inicialização deve ser calculado previamente através da análise dos diversos tempos de inicialização T_{Ii} de cada VM_i e, selecionando o maior deles, porque todas as máquinas virtuais são instanciadas em paralelo, tal como definido pela Equação 10.

$$T_I = \max_{0 \le x \le n-1} T_{Ii} \qquad (10)$$

O tempo de execução do *workflow* é a soma dos tempos de execução de cada programa P_j em uma determinada máquina virtual VM_i variando o número de parâmetros da atividade. A Equação 11 define o tempo de execução.

$$T_E = \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} T_{Ei,j} \qquad (11)$$

O tempo de transferência é calculado com base no volume de dados que são capturados e enviados para o servidor e a largura de banda disponível. Ele é definido pela seguinte fórmula:

$$T_R = \beta \times \left(\sum_{j=0}^{m-1} T_{Dj} + \sum_{j=0}^{m-1} T_{Uj}\right) \qquad (12)$$

Onde T_{Uj} é o tempo de upload, T_{Dj} é o tempo para fazer o download dos dados necessário para execução de um programa P_j específico e β é a largura de banda.

O modelo de custo visto, nessa seção, é utilizado pelo otimizador/dimensionador de recursos do SciCumulus-ECM para o cálculo de uma configuração de máquinas virtuais próxima a ideal, para a execução de um *workflow* científico, em um ambiente de nuvem. É importante ressaltar que o cientista deverá escolher qual critério priorizar no momento da otimização: se deseja que o *workflow* execute mais rápido ou que a execução seja mais barata.

5.3 O Otimizador

Conforme mencionado anteriormente, o objetivo principal do SciCumulus-ECM é informar quais tipos de máquinas virtuais e quantas máquinas virtuais de cada tipo devem ser instanciadas no ambiente, de maneira próxima à configuração ideal. Em outras palavras, este serviço é responsável por aperfeiçoar a seleção e instanciação dinâmica de máquinas virtuais antes de escalonar as atividades do *workflow* de fato. A escolha dos tipos de máquinas virtuais a serem utilizadas no escalonamento se assemelha em partes ao problema do caixeiro viajante (Applegate et al. 2007), onde existem diversos caminhos que devem ser percorridos com o menor custo total, assim como temos diversas configurações possíveis e deve ser escolhida a que produz o menor custo. Além disso, sua complexidade também se assemelha, sendo categorizada como um problema NP (Cormen et al. 2009), que o remete para um campo de complexidade exponencial, isto é, o esforço computacional necessário para a sua resolução cresce exponencialmente com o tamanho do problema.

Dessa forma, dado que é difícil (se não impossível), determinar a solução ótima deste problema em tempo hábil, optou-se por utilizar um método baseado em heurísticas que, do ponto de vista matemático, não asseguram a obtenção de uma solução ótima, porém apresentam grande chance de gerar uma solução aceitável em tempo viável (Bäck 1996).

Para o desenvolvimento do otimizador do SciCumulus-ECM optou-se por utilizar Algoritmos Genéticos (D. E. Goldberg 1989) para a escolha do conjunto de máquinas virtuais a serem utilizadas. A escolha pelos algoritmos genéticos se deve ao fato de serem algoritmos relativamente simples, além de serem indicados para casos onde o espaço a ser pesquisado é grande, ao mesmo tempo em que o problema não requer uma solução ótima. Segundo Ochi (1998), a ideia principal por trás dos algoritmos genéticos é tentar imitar algumas etapas do processo de evolução das espécies. A estrutura básica de um algoritmo genético é formada pelas seguintes etapas:

i. Representação de uma solução na estrutura de cromossomo;
ii. Construção de uma população inicial de cromossomos;
iii. Um mecanismo de avaliar os cromossomos segundo suas aptidões;

iv. Um mecanismo de reprodução de cromossomos - Operador Genético e;
v. Definição de parâmetros de uma AG.

Como infraestrutura-base para o otimizador foi utilizado o JGAP (Meffert 2012), um *framework* que fornece mecanismos genéticos básicos que podem ser facilmente usados para aplicar princípios evolutivos para soluções de problemas. Optou-se pela utilização do JGAP por ser de fácil adaptação para o problema que estamos tratando, como se pode ver a seguir.

Inicialmente, precisamos configurar um objeto de configuração descrevendo como queremos que nossos cromossomos sejam configurados. O cromossomo representa uma solução potencial e é dividida em múltiplos genes. Genes, no JGAP representam aspectos distintos da solução como um todo, assim como os genes humanos representam aspectos distintos de indivíduos, como sexo ou cor dos olhos. Portanto, o primeiro passo é decidir sobre a composição dos cromossomos, ou seja, quantos genes serão necessários e o que os genes representam. Nesta dissertação, os cromossomos foram modelados para que cada um tenha quatro genes, um para cada um dos tipos de máquinas virtuais que foram consideradas na execução do SciCumulus (*micro*, *large*, *extra-large* e *extra-large* de alta capacidade). Os valores (*i.e*. alelos) dos genes devem ser do tipo inteiro, uma vez que representam a quantidade de máquinas virtuais desse tipo que serão instanciadas. Também deve-se especificar um limite inferior e superior, que definem os valores para cada tipo de máquina virtual (comumente o valor mínimo é 1 e o máximo é 20, uma vez que utilizamos o Amazon EC2 e esta é uma limitação do provedor – não podemos instanciar mais de 20 máquinas virtuais na mesma conta).

```
Gene[] virtualMachines = new Gene[4];
virtualMachines[0] = new IntegerGene(conf, 0, 20); // Micro

virtualMachines[1] = new IntegerGene(conf, 0, 20); // Large

virtualMachines[2] = new IntegerGene(conf, 0, 20); // XLarge

virtualMachines[3] = new IntegerGene(conf, 0, 20); // XlargeHC
```

Figura 5 - Instanciação dos Genes e Cromossomos

O segundo passo é elaboração de uma função de "aptidão". O JGAP é projetado para fazer quase todo o trabalho evolutivo de uma forma relativamente simples e genérica. No entanto, ele não tem conhecimento do problema específico que você está realmente tentando resolver, e, portanto, não tem como decidir se uma possível solução é melhor do que outra solução potencial para um problema específico. A função de "aptidão" (chamada pelos desenvolvedores do JGAP de *fitness function*) é um método único que deve ser desenvolvido pelo usuário do JGAP, que retorna um valor inteiro que indica o quão boa é a solução em relação a outras soluções possíveis. Quanto mais alto o valor, melhor a solução. Quanto menor o valor, mais pobre a solução. Durante o processo de evolução, cromossomos estão expostos a vários operadores genéticos que representam o acasalamento, mutação, etc, e, em seguida, são escolhidas para a próxima geração, durante a fase de seleção natural, com base no valor de *fitness*. A função de *fitness* elaborada nessa dissertação, representada na Figura 6, é baseada nas restrições indicadas pelo cientista. A função foi escrita de modo que, independente da prioridade informada, nenhum dos dois critérios (tempo de execução, ou custo financeiro) é completamente descartado. Foi utilizado o peso de 80% para a prioridade informada pelo cientista, ao passo que o outro critério, que não é prioridade, recebe um peso de 20%. A cada geração, realizamos então uma simulação da execução do *workflow* baseada em um simulador (Vitor Viana et al. 2011) que implementa o modelo de custo proposto.

```
if (m_priority.equalsIgnoreCase("Cost")){
    fitness = (m_costConstraint - totalCost) * 0.8
        + (m_timeConstraint - totalTime) * 0.2;
}
else{
    fitness = (m_costConstraint - totalCost) * 0.2
        + (m_timeConstraint - totalTime) * 0.8;
}
```

Figura 6 - Função de aptidão

O terceiro passo é indicar quantos cromossomos existirão na população durante a execução do algoritmo genético. Como o objetivo principal da utilização desse algoritmo, nesse trabalho, é fazer a validação do modelo de custo para o problema de dimensionamento de máquinas virtuais, não foi feito nenhum estudo mais aprofundado sobre os melhores valores enviados como parâmetro para o algoritmo genético. Dessa forma, foram utilizados valores padrões e amplamente utilizados na resolução de diversos problemas. A população inicial de cromossomos é gerada aleatoriamente, por um método do JGAP, com 100 unidades. E o algoritmo foi configurado para que ocorram no máximo 200 gerações, uma vez que não podemos consumir muito tempo na configuração do ambiente ao invés de utilizar esse tempo na execução do *workflow*. Além de ter uma taxa de cruzamento de 35% da população e uma taxa de 8% de mutação do indivíduos.

Após essas adaptações, o algoritmo está pronto para começar a evolução da população e retornar a melhor configuração encontrada após as evoluções.

Capítulo 6 - Resultados Experimentais

Nesse capítulo, é apresentada a avaliação do SciCumulus-ECM. Para avaliar a viabilidade da abordagem proposta, utilizamos o *workflow* SciPhy de análise filogenética do domínio de bioinformática (Ocaña *et al*. 2011). Esse *workflow* será detalhado ao longo do capítulo. A partir da execução deste *workflow*, por meio do SciCumulus, com a utilização do SciCumulus-ECM, foram obtidos resultados que também serão apresentados a seguir.

O objetivo principal desse capítulo é analisar o tempo total de execução do *workflow* SciPhy, com a utilização do SciCumulus-ECM, em comparação com outras execuções do mesmo *workflow*, nas quais o escalonamento de suas atividades é feito sem uma configuração otimizada do ambiente, antes do início do experimento. Diversos testes foram executados de forma a avaliar o SciCumulus-ECM. Estes testes foram baseados em execuções prévias de *workflows* nas quais já se conhecia qual era a melhor configuração do ambiente para a execução dos experimentos.

Esse capítulo respeita a seguinte estrutura: A Seção 5.1 apresenta a configuração do ambiente utilizado para a execução do experimento. A Seção 5.2 apresenta o experimento base utilizado. A Seção 5.3 nos traz a análise comparativa entre desempenho estimado pelo modelo de custo do SciCumulus-ECM e o desempenho real. A seção 5.4 apresenta a análise do *overhead* de execução da otimização. A Seção 5.5 avalia o tempo de otimização do algoritmo ótimo, comparado ao tempo de execução do algoritmo genético. E, por fim, a seção 5.6 mostra a análise comparativa entre a execução normal do *workflow* e a execução utilizando o SciCumulus-ECM.

6.1 Ambiente Utilizado no Experimento

Para os experimentos executados nessa dissertação, implantamos o SciCumulus-ECM acoplado ao SciCumulus no ambiente de nuvem Amazon EC2. O Amazon EC2 oferece vários tipos diferentes de máquinas virtuais para serem utilizados pelos cientistas. 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), e *extra-large de alta capacidade* (EC2 ID: c1.xlarge - 7.5 GB RAM, 850 GB de armazenamento, 8 núcleos). A Tabela 1 resume os dados de desempenho e precificação dos tipos de máquinas virtuais passíveis de uso pelo SciCumulus-ECM.

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

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	1690 GB	4 UEC2	4	64 Bits	0,68
Extra-large de alta capacidade	7 GB	1690 GB	20 UEC2	8	64 Bits	1,16

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.

6.2 Experimento Base: O *Workflow* SciPhy

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

[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

i. A construção do alinhamento genético (MSA);
ii. A conversão de formato do alinhamento;
iii. A pesquisa sobre o melhor modelo evolutivo a ser usado;
iv. A construção da árvore filogenética;
v. A concatenação dos alinhamentos produzidos de forma a gerar um super-alinhamento que servirá para inferir a relação do genoma completo;
vi. A escolha do melhor modelo evolutivo para o super-alinhamento e;
vii. A construção da árvore filogenômica que contém informações sobre todos os organismos associados.

Estas atividades, respectivamente, executam as seguintes aplicações de bioinformática: programas de alinhamento genético (permitindo ao cientista a escolher entre o MAFFT, o Kalign, o ClustalW, o Muscle, ou o ProbCons), o ReadSeq (Gilbert 2003), o ModelGenerator (Keane *et al.* 2006), o RAxML (Stamatakis 2006), um *script* Perl, o ModelGenerator e o RAxML. A Figura 7 apresenta a visão conceitual do SciPhy.

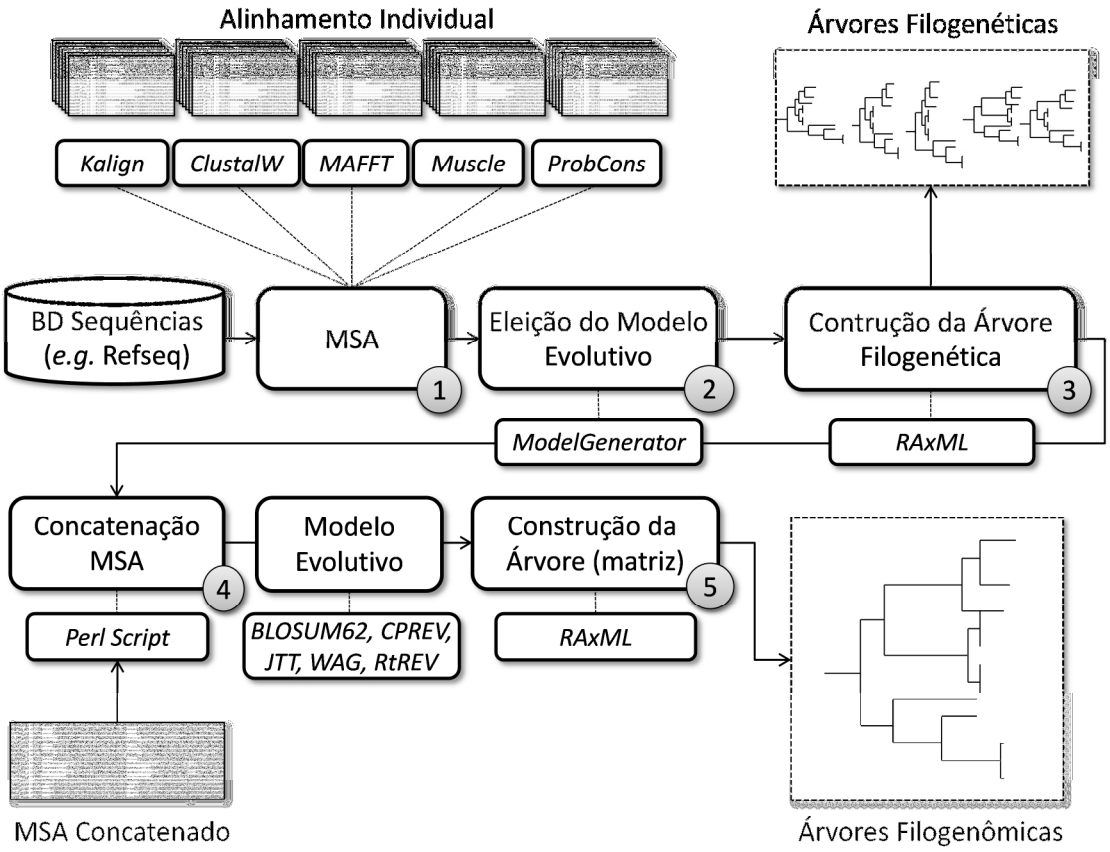

Figura 7 - Especificação conceitual do *workflow* SciPhy

A primeira atividade do SciPhy (MSA) constrói alinhamentos individuais utilizando um dos cinco programas disponíveis para alinhamento genético: o ClustalW, o Kalign, o MAFFT, o Muscle, ou o ProbCons. Cada programa de alinhamento recebe um arquivo multi-fasta (que pode representar um aminoácido ou não) como entrada (a partir de um conjunto de arquivos multi-fasta existentes), produzindo como saída um alinhamento (MSA). Neste ponto, o SciPhy obtém o MSA individual. Na segunda atividade, o alinhamento é convertido para o formato PHYLIP (Felsenstein J 1989). Formato este que é alcançado através da utilização do programa ReadSeq.

Cada MSA é testado na terceira atividade (Eleição do Modelo Evolutivo) para encontrar o melhor modelo evolutivo utilizando o ModelGenerator e ambos (MSA individual e modelo evolutivo) são utilizados na quarta atividade (Geração da Árvore Filogenética) para gerar árvores filogenéticas utilizando o RAxML. Consequentemente, várias árvores diferentes podem ser obtidas para cada um dos programas de MSA que foram eleitos. Uma vez que os cientistas não conhecem *a priori* qual o método de

alinhamento que produz o melhor resultado final (a melhor árvore), eles têm que realizar uma análise filogenética exploratória (varredura de parâmetros), ou seja, executar o SciPhy várias vezes (uma vez para cada método MSA).

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

Apesar de o SciPhy ser conceitualmente simples, 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. Um experimento típico de filogenia pode analisar centenas ou milhares de arquivos multi-fasta, cada um contendo centenas ou milhares de sequências biológicas. Como o processamento destes arquivos e das combinações de parâmetros é independente, o SciPhy se torna um candidato interessante para explorarmos a execução paralela em experimentos de bioinformática. No contexto do SciCumulus, cada combinação de parâmetros mais dados de entrada no SciPhy gerará uma atividade diferente, que poderá ser processada em paralelo, aumentando assim a eficiência na execução do workflow.

Este *workflow* é representativo devido a sua característica de necessitar de grande poder de processamento e de produzir um grande volume de dados. Por exemplo, com uma configuração de 200 arquivos multi-fasta de entrada (este valor pode ser muito superior), a execução do SciPhy em uma máquina com dois núcleos, apenas variando os métodos de alinhamento (5 execuções) necessita em média 9 dias para terminar. O que torna a configuração inicial das máquinas virtuais instanciadas no ambiente importantíssima. Além disso, sem a aplicação de técnicas de paralelismo e com o aumento da combinação de parâmetros e dados de entrada a serem explorados, esse tempo pode se tornar proibitivo para a execução do experimento. Por esta razão, foi escolhido como estudo de caso do quarto *Provenance Challenge* (twiki.ipaw.info/bin/view/Challenge/).

6.3 Calibração do Modelo

Antes de avaliar o SciCumulus-ECM com o SciPhy optamos por realizar uma calibragem do modelo de custo e do otimizador com um *workflow* diferente para que o mesmo não se tornasse tendencioso ao SciPhy. Como caso real de execução de *workflows* científicos para calibrar o modelo, foram utilizadas 524 execuções do *workflow* de cristalografia (Oliveira *et al.* 2011), que foi executado em paralelo na nuvem da Amazon EC2.

A configuração de execução utilizada nessa seção se refere ao processamento paralelo de 2.000 imagens de entrada separadas em grupos de 3 imagens, o que gerou um conjunto de 667 atividades a serem executadas na nuvem. Dado o volume de dados a ser processado, este *workflow*, em uma máquina com um núcleo virtual, executa em aproximadamente 45.876 segundos (12,8 horas). A partir do cenário inicial (nenhuma máquina virtual criada), o otimizador vai executar o algoritmo genético de forma a determinar o número de máquinas virtuais envolvidas na execução e verificando se o tempo total de execução reduziu proporcionalmente até alcançar as restrições impostas pelo cientista. (dependendo da prioridade de critério fornecida: tempo ou custo financeiro).

Nesta primeira calibragem, as simulações se deram sempre em relação a quantidade de máquinas sem variação de tipo. Ou seja, o otimizador analisou os cenários em que todas as máquinas eram do mesmo tipo e apenas determinou a quantidade "ideal". A Figura 8 apresenta os resultados de simulação de tempo para as configurações escolhidas pelo otimizador, a Figura 9 apresenta os resultados reais para o tempo de execução do *workflow* com o SciCumulus, enquanto que a Figura 10 exibe a comparação entre o custo financeiro real e o custo financeiro simulado. Como podemos perceber, existe uma diferença pequena de tempo entre a simulação e o real. Entretanto, esta execução foi útil para melhorarmos nossa implementação de forma a melhor representar o cenário de escalonamento.

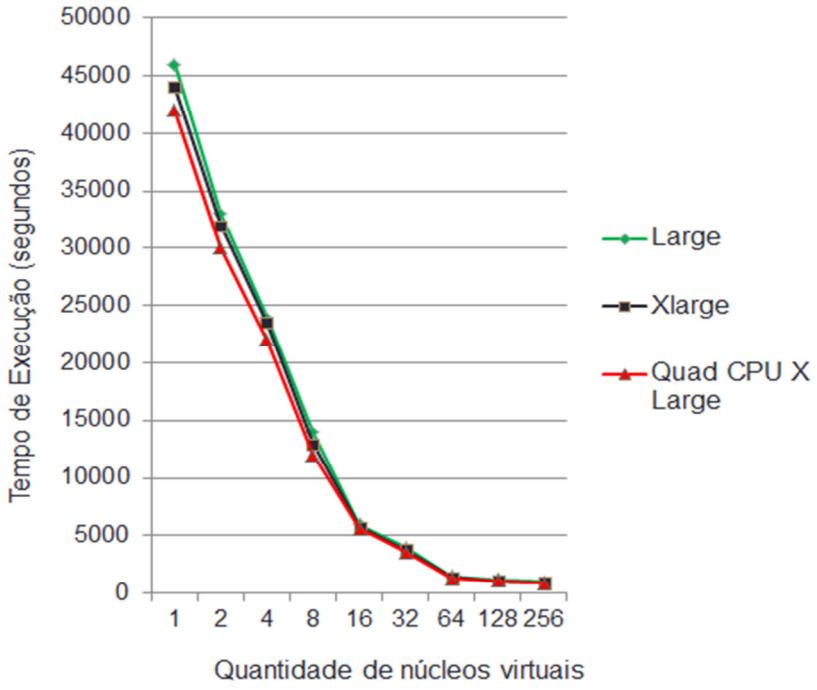

Figura 8 - Resultados simulados para os tempos de execução

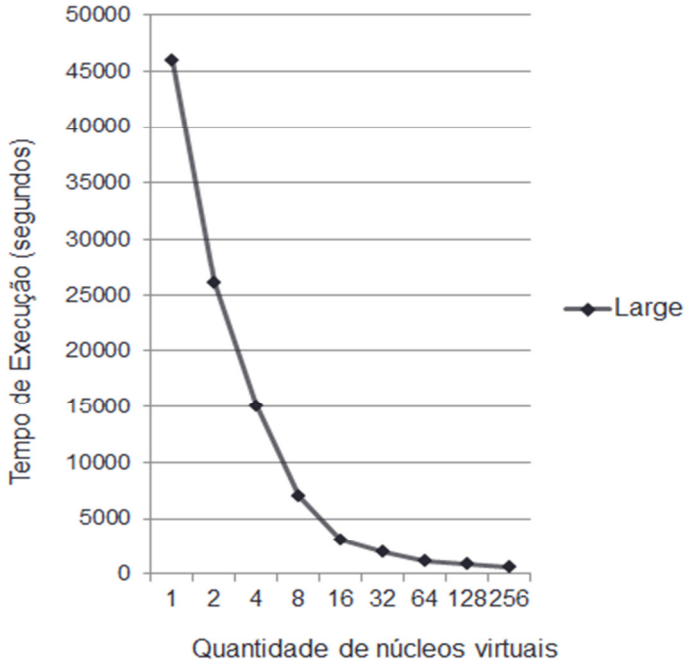

Figura 9 - Resultados reais para os tempos de execução

Esta análise foi executada para cada tipo de máquina virtual existente analisando o custo monetário e o tempo decorrido de execução. Para a execução mostrada nessa seção, as restrições foram de US$ 10,00 de custo financeiro e 1 hora de tempo de execução máximo. Uma vez que problemas de exploração de parâmetros são em sua maioria embaraçosamente paralelos (atividades independentes), a mudança do tipo de máquina não impactou tanto no desempenho simulado do workflow. Este comportamento pode ser explicado uma vez que *workflows* embaraçosamente paralelos são considerados um padrão altamente escalável para nuvens (Jackson *et al*. 2010). Nesta simulação, a restrição de tempo imposta foi alcançada utilizando entre 16 e 256 cores virtuais, como apresentado na Figura 8. Embora o desempenho tenha sido equivalente utilizando os diferentes tipos de máquinas virtuais, o custo financeiro foi significantemente diferente. Por exemplo, o preço pago nas simulações variou de US$ 2,44 (16 cores) até US$ 39,16 (256 cores). Neste experimento foi definido como prioridade o custo monetário ao invés do tempo de execução. Desta forma, o otimizador fixou o número de máquinas virtuais em 16 (uma vez que utilizando esta quantidade de cores a restrição de tempo era satisfeita) e mudou os tipos de máquinas virtuais possíveis a fim de reduzir os custos. Como resultado a instancia *Large* da Amazon foi considerada a mais adequada do ponto de vista de custo monetário.

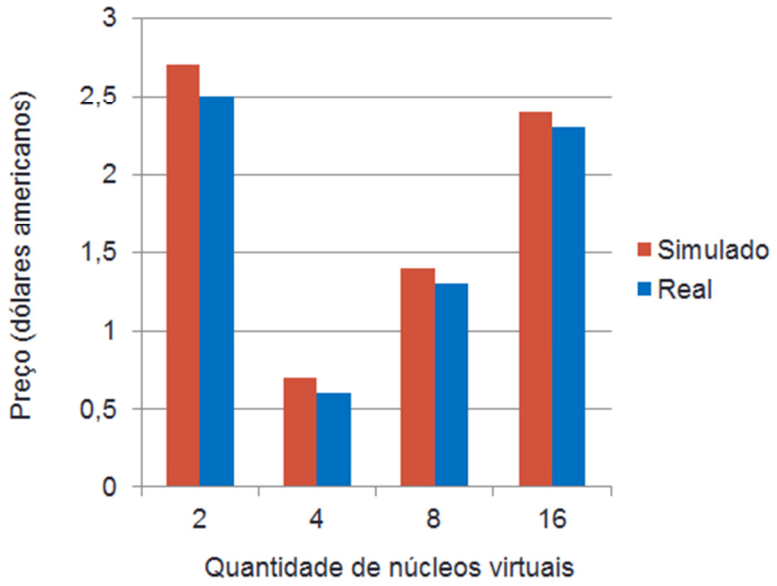

Figura 10 - Comparação entre custo simulado e custo real

6.4 Tempo de Otimização x Tempo de Execução do *Workflow*

Dado que o modelo e o otimizador já estavam calibrados, deveríamos avaliar se o tempo de otimização era aceitável frente ao tempo para executar o *workflow*. Para avaliar o impacto da otimização realizada pelo SciCumulus-ECM, frente ao tempo total de execução de um *workflow*, executamos o *workflow* SciPhy com prioridade para o tempo de execução total e processando 400 arquivos multi-fasta de entrada e limitando o orçamento em US$ 75,00.

Este *workflow* levou, em média, 204 horas para ser executado nas execuções prévias realizadas utilizando apenas o SciCumulus sem otimização e dimensionamento do ambiente. Em todos os testes realizados, para processar as 200 evoluções no algoritmo genético, com uma população de 100 cromossomos, o SciCumulus-ECM levou em média 5,5 minutos, tempo este que pode ser considerado desprezível frente ao tempo total de execução do *workflow* e frente ao ganho na execução proporcionado pelo SciCumulus-ECM, como será apresentado a seguir.

6.5 Força bruta x Algoritmo Genético

Para avaliar a eficiência da abordagem que utiliza o algoritmo genético, frente à abordagem que verifica todos os casos possíveis de combinações de máquina virtuais, executamos o *workflow* SciPhy com prioridade para o custo financeiro e processando 400 arquivos multi-fasta de entrada e limitando o orçamento em US$ 40,00.

Foram analisadas várias possibilidades de configuração do ambiente, ou seja, foram impostas limitações de instanciação de máquinas virtuais (2MVs, 4 MVs, 8 MVs, 16 MVs, 32 MVs e 64 MVs). Lembrando que a Amazon só permite a instanciação de 20 máquinas, porém os testes foram realizados sem levar essa limitação em consideração, pois este cenário não é constante em todos os provedores desse tipo de serviço. No algoritmo genético, foram executadas 200 evoluções, com uma população inicial de 100 cromossomos.

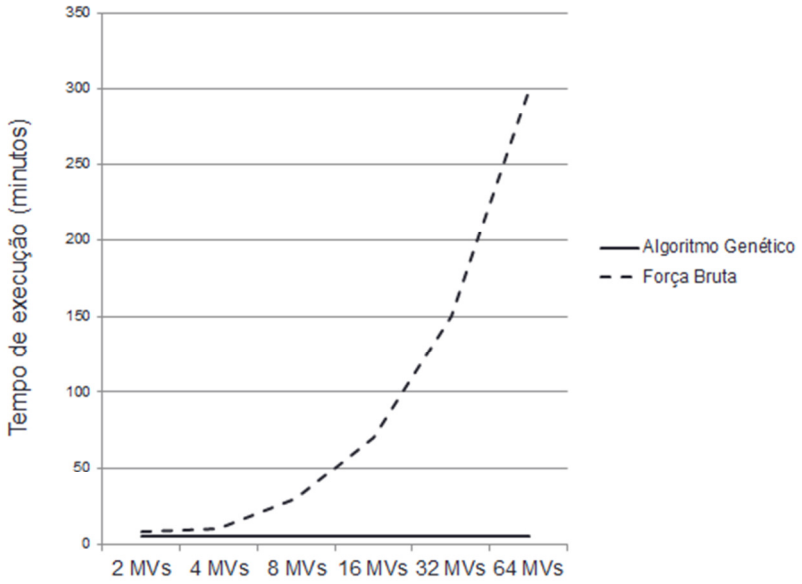

Figura 11 - Algoritmo Genético x Força Bruta

Conforme esperado, o tempo para se encontrar a configuração ótima se torna proibitivo conforme a quantidade de máquinas aumenta. Como se pode ver, na Figura 11, o algoritmo genético mantém quase o mesmo tempo para otimizar a configuração do ambiente, independente do número de máquinas virtuais que devem ser avaliadas. Isto acontece, pois esse tempo é baseado apenas no tamanho da população e no número de evoluções que o algoritmo está programado para realizar. Vale ressaltar que quanto maior o número de evoluções, maior será a qualidade da otimização, porém o tempo para otimizar será um pouco maior. Enquanto que no algoritmo que avalia todos os casos possíveis (força bruta), o tempo de otimização, à medida que o número de máquinas virtuais aumenta, cresce exponencialmente.

6.6 Análise de Desempenho do *Workflow* com SciCumulus-ECM

Para avaliar se o SciCumulus-ECM conseguiu melhorar o desempenho do SciPhy executando com o SciCumulus realizamos uma execução completa do SciPhy com o método adaptativo do SciCumulus, que foi explicado em 4.2 Execução Estática x Execução adaptativa. Apesar da abordagem adaptativa do SciCumulus conseguir distribuir melhor as atividades de acordo com os recursos existentes e dimensionar a

quantidade de recursos envolvidos na execução (durante a execução), ela ainda pode ser otimizada em vários sentidos. Um dos pontos de melhoria se refere à otimização da coleção inicial de máquinas virtuais. Nos exemplos da Figura 12 iniciamos a execução com nenhuma máquina virtual disponível.

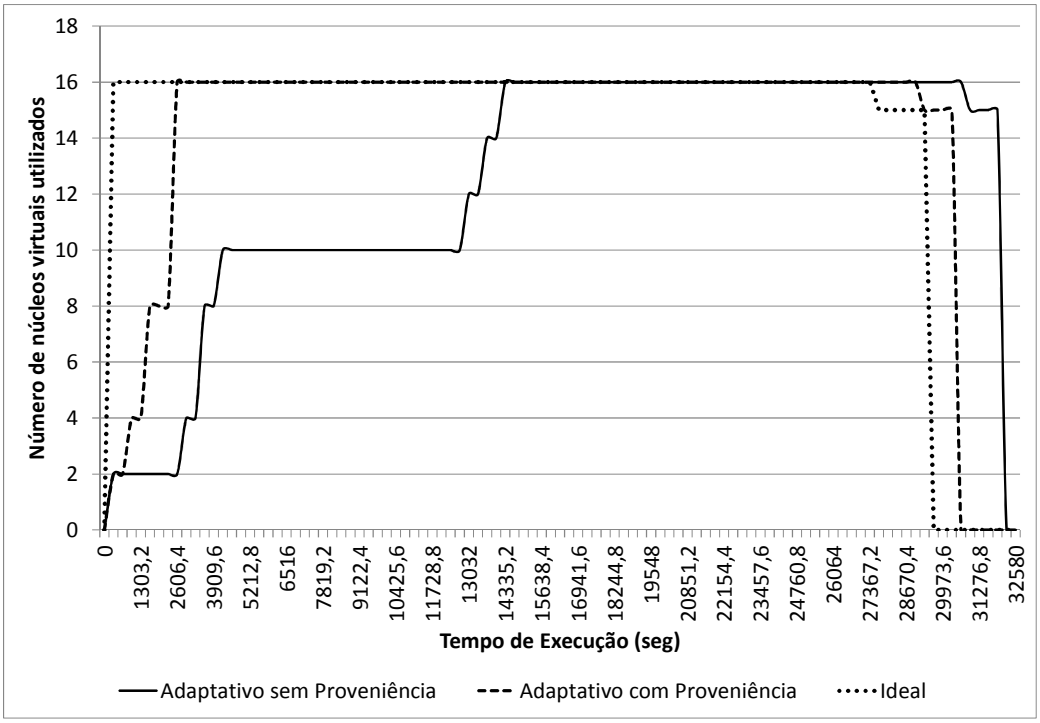

Figura 12 - Execução adaptativa

O SciCumulus então vai aumentando a quantidade de recursos gradativamente de acordo com as restrições impostas pelos cientistas e de acordo com a previsão de demanda baseada nas consultas de proveniência. Entretanto, esse aumento gradual tem uma sobrecarga de processamento embutida, pois perdemos tempo aumentando (ou diminuindo) a quantidade de máquinas envolvidas (uma vez que as máquinas tem um custo de inicialização que não pode ser negligenciado).

Essa sobrecarga é diminuída caso o conjunto inicial de máquinas virtuais seja suficientemente perto do ideal para o experimento. Para dimensionar este conjunto de máquinas utilizamos o SciCumulus-ECM. Um exemplo da utilização do SciCumulus-ECM, com o SciCumulus, é apresentado na Figura 13.

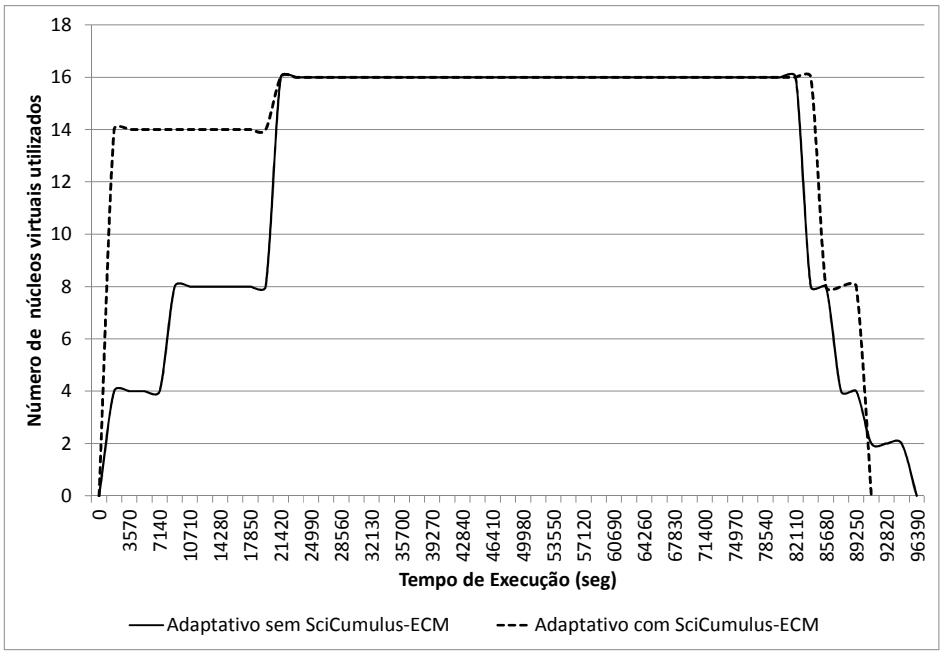

Figura 13 - Execução com SciCumulus-ECM

Neste exemplo, executamos o *workflow* SciPhy processando 400 arquivos multi-fasta de entrada e limitando o orçamento em US$ 75,00. Sem a utilização do SciCumulus-ECM partimos do zero e o SciCumulus escala a quantidade de máquinas conforme analisa os dados de proveniência que são gerados em tempo de execução. Os aumentos de quantidade de máquinas ocorrem na análise de desempenho das atividades de alinhamento, de conversão e de geração do modelo. Entretanto, esse aumento do número de máquinas virtuais depende de análise de dados de proveniência e leva tempo, fazendo com que o *workflow* demora mais a terminar.

Com a utilização do SciCumulus-ECM já iniciamos a execução com 14 máquinas virtuais e este valor só é aumentado para 16 no momento do término da primeira execução de atividade do *workflow* associada à atividade modelgenerator do SciCumulus (que é uma atividade computacionalmente intensiva). Podemos perceber que existe uma diferença de 6% entre a execução com o SciCumulus-ECM (91.035,00 segundos) e sem a utilização do SciCumulus-ECM (96.390,00 segundos). Apesar de percentualmente a diferença não ser grande, ela tende a escalar de acordo com o aumento da complexidade do *workflow*, *i.e.* quanto maior for o tempo de execução do *workflow*, maior será a diferença de tempo absoluta. É importante ressaltar que o tempo de otimização gasto com o algoritmo genético do SciCumulus-ECM não se encontra contabilizado neste gráfico. Porém, em todos os testes realizados, para

processar as 200 evoluções no algoritmo genético, com uma população de 100 cromossomos, o SciCumulus-ECM levou em média 5,5 minutos, tempo este que pode ser considerado desprezível frente ao tempo total de execução do *workflow* e frente ao ganho na execução proporcionado pelo SciCumulus-ECM.

Capítulo 7 - Conclusões e Trabalhos Futuros

Esse capítulo apresenta as conclusões gerais desse trabalho, englobando as principais contribuições e as principais perspectivas de trabalhos futuros que podem ser desenvolvidos como consequência direta dessa dissertação.

A abordagem apresenta alguns serviços referentes à configuração de ambientes distribuídos, mais especificamente, nuvens de computadores.

No contexto de *workflows* científicos, nuvens de computadores são utilizadas como ambiente de alto desempenho para processamento de grandes volumes de dados. Todavia, estes ambientes, são geralmente geridos no modelo *pague pelo uso*. Desta maneira, se não houver um estudo sobre a melhor configuração que se deve contratar nesse tipo de cenário, o custo financeiro, além do tempo total de execução, podem inviabilizar o experimento. A hipótese presente nesse trabalho, é que se for adotado um modelo de custo e uma abordagem de otimização baseada em metaheurísticas (*e.g.* algoritmos genéticos), então pode-se reduzir tempo e custo financeiro da execução de um *workflow* em nuvens de computadores.

Este trabalho apresenta as seguintes contribuições:

i. um componente de captura de informações sobre ambientes de nuvens computacionais.

ii. um modelo de custo, que foca na otimização de dois critérios: tempo de execução total do *workflow* e custo monetário de execução.

iii. um otimizador, que visa determinar a melhor configuração possível do ambiente de nuvem antes de efetivamente escalonar as atividades do *workflow* neste ambiente.

Essa dissertação é a união de algumas publicações associadas à configuração de ambientes de nuvens de computadores, para execução de experimentos científicos. As contribuições mencionadas podem ser sumarizadas na seleção das seguintes publicações realizadas durante o mestrado:

i. V. Viana, D. de Oliveira, e M. Mattoso (2011), "Towards a Cost Model for Scheduling Scientific *Workflows Activities* in Cloud Environments". In: 2011

IEEE World Congress on Services (SERVICES) 2011 IEEE World Congress on Services (SERVICES), p. 216-219

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

Essa dissertação apresentou um modelo de custo, baseado em nuvem, utilizado para ajudar a dimensionar os recursos do ambiente antes de programar as atividades para esses ambientes. Este modelo de custo visa otimizar dois critérios: tempo de execução total e custo monetário do *workflow*. Pode-se verificar que o modelo de custo foi capaz de avaliar satisfatoriamente o custo envolvido na execução do experimento nas diferentes configurações do ambiente de nuvem. Todos os resultados de simulação no que se referiu ao custo foram na prática equivalentes aos custos reais (3% de diferença). Isto indica que uma sintonia fina do modelo ainda é necessária, entretanto os resultados atuais são motivadores e significativos.

Em conjunto com o modelo de custo, o otimizador apresentado nessa dissertação era responsável por determinar a melhor configuração do ambiente. Ou seja, a quantidade de cada tipo de máquinas virtuais disponível no provedor, no caso, a Amazon, que deveria ser instanciada para que a execução do *workflow* atendesse aos critérios dos cientistas. Pode-se observar nos experimentos (todos executados na nuvem da Amazon), que ao adotarmos um modelo de custo focado na distribuição das atividades de um *workflow* na nuvem e em uma abordagem de otimização baseada em metaheurísticas (*e.g.* algoritmos genéticos), então podemos melhor dimensionar o ambiente de nuvem de forma a reduzir tempo e custo financeiro da execução de um *workflow* quando comparados a ambientes que não tiveram a quantidade de recursos dimensionada. O otimizador conseguiu reduzir o custo envolvido na execução do *workflow* com um ganho de 6% em relação à execução sem a utilização do nosso otimizador. Como dito anteriormente, apesar de percentualmente a diferença não ser grande, ela tende a escalar de acordo com o aumento da complexidade do *workflow*, *i.e.* quanto maior for o tempo de execução do *workflow*, maior será a diferença de tempo absoluta.

Por questões de simplificação na implementação, o componente que captura informações dos ambientes de nuvem está, hoje, com a implementação dependente da API da Amazon EC2. Idealmente, deve ser independente de ambiente. Entretanto, o ambiente da Amazon EC2 foi mais estável (senão o único disponível de fato) para que pudéssemos realizar os experimentos. Como trabalho futuro, pode-se citar a evolução do componente que captura informações do ambiente para que contemple outros provedores.

Outro ponto importante seria a realização de novos experimentos utilizando este modelo de custo para instanciar o ambiente, a fim de avaliar sua eficiência. Além dos já citados, outros trabalhos futuros seriam a variação de parâmetros utilizados pelo algoritmo genético, na tentativa de obter melhor resultados, e também a proposição de novas heurísticas de simulação que demandem menos tempo de processamento.

Referências

Aalst, Wil van der, e Kees van Hee. 2002. *Workflow Management: Models, Methods, and Systems*. The MIT Press.

de Almeida-Neto, Cesar et al. 2011. "Demographic characteristics and prevalence of serologic markers among blood donors who use confidential unit exclusion (CUE) in São Paulo, Brazil: implications for modification of CUE polices in Brazil". *Transfusion* 51(1): 191–197.

Altintas, I. et al. 2004. "Kepler: an extensible system for design and execution of scientific workflows". In *Scientific and Statistical Database Management*, Greece, p. 423-424.

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

Applegate, David L. et al. 2007. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

Armbrust, Michael et al. 2010. "A view of cloud computing". *Commun. ACM* 53(4): 50-58.

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

Bal, Henri E., Jennifer G. Steiner, e Andrew S. Tanenbaum. 1989. "Programming languages for distributed computing systems". *ACM Comput. Surv.* 21(3): 261-322.

Barham, Paul et al. 2003. "Xen and the art of virtualization". In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, Bolton Landing, NY, USA: ACM, p. 164-177. http://portal.acm.org/citation.cfm?id=945445.945462&coll=GUIDE&dl=GUIDE&CFID=95425877&CFTOKEN=56736045 (Acessado junho 30, 2010).

Barker, Adam, e Jano van Hemert. 2008. "Scientific Workflow: A Survey and Research Directions". In *Parallel Processing and Applied Mathematics*, , p. 746-753. http://dx.doi.org/10.1007/978-3-540-68111-3_78 (Acessado novembro 8, 2009).

Callahan, Steven P. et al. 2006. "VisTrails: visualization meets data management". In *SIGMOD International Conference on Management of Data*, Chicago, Illinois, USA: ACM, p. 745-747. http://portal.acm.org/citation.cfm?id=1142473.1142574 (Acessado julho 9, 2008).

de Carvalho Costa, Rogério Luís, e Pedro Furtado. 2008. “A QoS-oriented external scheduler”. In *Proceedings of the 2008 ACM symposium on Applied computing*, SAC ’08, New York, NY, USA: ACM, p. 1029–1033. http://doi.acm.org/10.1145/1363686.1363923 (Acessado julho 16, 2012).

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

Cavalcanti, Maria Cláudia et al. 2005. “Managing structural genomic workflows using web services”. *Data & Knowledge Engineering* 53(1): 45-74.

Chen, Yong, Xian-He Sun, e Ming Wu. 2008. “Algorithm-system scalability of heterogeneous computing”. *J. Parallel Distrib. Comput.* 68(11): 1403-1412.

Cormen, Thomas H. et al. 2009. *Introduction to Algorithms*. third edition. The MIT Press.

Costa, Rogério Luís de Carvalho, e Pedro Furtado. 2009. “Runtime Estimations, Reputation and Elections for Top Performing Distributed Query Scheduling”. In *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, Shanghai, China, p. 28-35. http://ieeexplore.ieee.org/Xplore/login.jsp?url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F5071832%2F5071833%2F05071851.pdf%3Farnumber%3D5071851&authDecision=-203 (Acessado fevereiro 17, 2011).

Coutinho, Fábio et al. 2011. “Many task computing for orthologous genes identification in protozoan genomes using Hydra”. *Concurrency and Computation: Practice and Experience* 23(17): 2326-2337.

L. D. Davis, e Melanie Mitchell. 1991. *Handbook of Genetic Algorithms*. CiteSeerX. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.87.3586 (Acessado setembro 11, 2012).

Davidson, Susan B., e Juliana Freire. 2008. “Provenance and scientific workflows: challenges and opportunities”. In *ACM SIGMOD international conference on Management of data*, Vancouver, Canada: ACM, p. 1345-1350. http://portal.acm.org/citation.cfm?id=1376772 (Acessado maio 1, 2009).

Dávila, Alberto M. R. et al. 2008. “ProtozoaDB: dynamic visualization and exploration of protozoan genomes”. *Nucleic Acids Research* 36(Database issue): D547-D552.

Deelman, Ewa et al. 2009. “Workflows and e-Science: An overview of workflow system features and capabilities”. *Future Generation Computer Systems* 25(5): 528-540.

Dias, Jonas et al. 2011. “Poster: scientific data parallelism using P2P technique”. In *Proceedings of the 2011 companion on High Performance Computing Networking, Storage and Analysis Companion*, SC ’11 Companion, New York,

NY, USA: ACM, p. 27–28. http://doi.acm.org/10.1145/2148600.2148615 (Acessado julho 16, 2012).

Dutta, D., e R. C. Joshi. 2011. "A genetic: algorithm approach to cost-based multi-QoS job scheduling in cloud computing environment". In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, ICWET '11, New York, NY, USA: ACM, p. 422–427. http://doi.acm.org/10.1145/1980022.1980111 (Acessado julho 16, 2012).

Elmasri, Ramez E., e Shamkant B. Navathe. 2005. *Sistemas de Banco de Dados*. 4a ed. Addison Wesley.

Evsukoff, Alexandre, Beatriz Lima, e Nelson Ebecken. 2011. "Long-Term Runoff Modeling Using Rainfall Forecasts with Application to the Iguaçu River Basin". *Water Resources Management* 25(3): 963-985.

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

Fileto, Renato et al. 2003. "POESIA: An ontological workflow approach for composing Web services in agriculture". *The VLDB Journal* 12(4): 352–367.

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

Freire, Juliana et al. 2008. "Provenance for Computational Tasks: A Survey". *Computing in Science and Engineering, v.10* (3): 11-21.

Garey, Michael R., e David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co.

Garg, Saurabh Kumar, Rajkumar Buyya, e H. J Siegel. 2009. "Scheduling Parallel Applications on Utility Grids: Time and Cost Trade-off Management". In Wellington, New Zealand. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.144.6566 (Acessado agosto 27, 2011).

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

GoGrid. 2012. *Cloud Hosting, Cloud Servers, Hybrid Hosting, Cloud Infrastructure from GoGrid*. http://www.gogrid.com/ (Acessado setembro 8, 2012).

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

Goncalez, Thelma T. et al. 2011. "Blood transfusion utilization and recipient survival at Hospital das Clinicas in São Paulo, Brazil". *Transfusion*: no–no.

Gorder, Pam Frost. 2008. “Coming Soon: Research in a Cloud”. *Computing in Science and Engg.* 10(6): 6-10.

Gounaris, Anastasios et al. 2004. “Resource Scheduling for Parallel Query Processing on Computational Grids”. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID ’04, Washington, DC, USA: IEEE Computer Society, p. 396–401. http://dx.doi.org/10.1109/GRID.2004.55 (Acessado julho 16, 2012).

Governato, F. et al. 2010. “Bulgeless dwarf galaxies and dark matter cores from supernova-driven outflows”. *Nature* 463(7278): 203-206.

Guerra, Gabriel et al. 2009. “Scientific Workflow Management System Applied to Uncertainty Quantification in Large Eddy Simulation”. In *Congresso Ibero Americano de Métodos Computacionais em Engenharia*, Búzios, Rio de Janeiro, Brazil: CILAMCE, p. 1-13.

Guerra et al. 2012. “Uncertainty Quantification in Computational Predictive Models for Fluid Dynamics Using Workflow Management Engine”. *International Journal for Uncertainty Quantification* 2(1): 53-71.

Guerra, Gabriel. M, e Fernando A Rochinha. 2009a. “A Sparse Grid Method Applied to Stochastic Fluid-Structure Interaction”. In *COBEM, 2009, Gramado, Brazil*,.

Guerra e Rochinha. 2010. “Stochastic modeling of Flow-Structure Interaction using a Sparse Grid Stochastic Collocation Method”. In *IV European Congress on Computational Mechanics, 2010, Paris*,.

Guerra e Rochinha. 2009. “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*,.

Hartman, Amber L et al. 2010. “Introducing W.A.T.E.R.S.: a Workflow for the Alignment, Taxonomy, and Ecology of Ribosomal Sequences”. *BMC Bioinformatics* 11(1): 317.

He, Ligang et al. 2011. “Optimizing Resource Consumptions in Clouds”. In *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, GRID ’11, Washington, DC, USA: IEEE Computer Society, p. 42–49. http://dx.doi.org/10.1109/Grid.2011.15 (Acessado julho 16, 2012).

Hey, Tony, Stewart Tansley, e Kristin Tolle. 2009. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research.

Hull, Duncan et al. 2006. “Taverna: a tool for building and running workflows of services”. *Nucleic Acids Research* 34(2): 729-732.

Isard, Michael et al. 2009. “Quincy: fair scheduling for distributed computing clusters”. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP ’09, New York, NY, USA: ACM, p. 261–276. http://doi.acm.org/10.1145/1629575.1629601 (Acessado julho 16, 2012).

Jackson, Keith R et al. 2010. “Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud”. In *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, CLOUDCOM ’10, Washington, DC, USA: IEEE Computer Society, p. 159–168. http://dx.doi.org/10.1109/CloudCom.2010.69 (Acessado fevereiro 21, 2011).

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

Juristo, Natalia, e Ana M. Moreno. 2010. *Basics of Software Engineering Experimentation*. Softcover reprint of hardcover 1st ed. 2001. Springer.

Keane, Thomas M et al. 2006. “Assessment of methods for amino acid matrix selection and their use on empirical data shows that ad hoc assumptions for choice of matrix are not justified”. *BMC Evolutionary Biology* 6: 29.

Kim, Won et al. 2009. “Adoption issues for cloud computing”. In *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, Kuala Lumpur, Malaysia: ACM, p. 3-6. http://portal.acm.org/citation.cfm?id=1806338.1806341&coll=GUIDE&dl=GUIDE&CFID=95425877&CFTOKEN=56736045 (Acessado junho 30, 2010).

Lemos, Melissa et al. 2004. “Ontology-Driven Workflow Management for Biosequence Processing Systems”. In *Database and Expert Systems Applications*, orgs. Fernando Galindo, Makoto Takizawa, e Roland Traunmüller. Berlin, Heidelberg: Springer Berlin Heidelberg, p. 781-790. http://www.springerlink.com/content/rx24ttq8239ymcjr/ (Acessado novembro 3, 2011).

Leusse, Pierre de et al. 2008. “Secure & Rapid Composition of Infrastructure Services in the Cloud”. In *Proceedings of the 2008 Second International Conference on Sensor Technologies and Applications*, IEEE Computer Society, p. 770-775. http://portal.acm.org/citation.cfm?id=1446297.1446607&coll=GUIDE&dl=GUIDE&CFID=95425877&CFTOKEN=56736045 (Acessado junho 30, 2010).

Lins, Erb F et al. 2009. “Edge-based finite element implementation of the residual-based variational multiscale method”. *International Journal for Numerical Methods in Fluids* 61(1): 1–22.

Macías, Mario, e Jordi Guitart. 2011. “A genetic model for pricing in cloud computing markets”. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC ’11, New York, NY, USA: ACM, p. 113–118. http://doi.acm.org/10.1145/1982185.1982216 (Acessado julho 16, 2012).

Mani, Sudha, e Shrisha Rao. 2011. “Operating cost aware scheduling model for distributed servers based on global power pricing policies”. In *Proceedings of the Fourth Annual ACM Bangalore Conference*, COMPUTE ’11, New York, NY, USA: ACM, p. 12:1–12:8. http://doi.acm.org/10.1145/1980422.1980434 (Acessado julho 16, 2012).

Marinos, Alexandros, e Gerard Briscoe. 2009. “Community Cloud Computing”. In *Proceedings of the 1st International Conference on Cloud Computing*, Beijing, China: Springer-Verlag, p. 472-484. http://portal.acm.org/citation.cfm?id=1695659.1695704&coll=GUIDE&dl=GUIDE&CFID=95425877&CFTOKEN=56736045 (Acessado junho 30, 2010).

Martinho, W. et al. 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. et al. 2008. “Gerência de Workflows Científicos: Uma Análise Crítica no Contexto da Bioinformática”. *COPPE/UFRJ* (Relatório técnico).

Mattoso, M. et al. 2008. “Gerenciando Experimentos Científicos em Larga Escala”. In *SEMISH - CSBC*, Belém, Pará - Brasil, p. 121-135.

Mattoso, Marta et al. 2010. “Towards Supporting the Life Cycle of Large-scale Scientific Experiments”. *International Journal of Business Process Integration and Management* 5(1): 79–92.

Meffert, Klaus et al. 2012. “JGAP - Java Genetic Algorithms and Genetic Programming Package”. http://jgap.sf.net (Acessado setembro 9, 2012).

Napper, Jeffrey, e Paolo Bientinesi. 2009. “Can cloud computing reach the top500?” In *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, Ischia, Italy: ACM, p. 17-20. http://portal.acm.org/citation.cfm?id=1531666.1531671&coll=GUIDE&dl=GUIDE&CFID=95425877&CFTOKEN=56736045 (Acessado junho 30, 2010).

Nguyen, Thi-Van-Anh et al. 2012. “Cost models for view materialization in the cloud”. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT ’12, New York, NY, USA: ACM, p. 47–54. http://doi.acm.org/10.1145/2320765.2320788 (Acessado julho 16, 2012).

Ocaña, Kary A. C. S., Daniel Oliveira, Jonas Dias, et al. 2011. “Optimizing Phylogenetic Analysis Using SciHmm Cloud-based Scientific Workflow”. In *2011 IEEE Seventh International Conference on e-Science (e-Science)*, Stockholm, Sweden: IEEE, p. 190-197.

Ocaña, Kary A. C. S., Daniel Oliveira, Eduardo Ogasawara, et al. 2011. “SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes”. In *Advances in Bioinformatics and Computational Biology*, orgs. Osmar Norberto de Souza, Guilherme P. Telles, e Mathew Palakal. Berlin, Heidelberg: Springer Berlin Heidelberg, p. 66-70. http://www.springerlink.com/index/10.1007/978-3-642-22825-4_9 (Acessado agosto 25, 2011).

Ochi, Luiz Satoru. 1998. *Algoritmos genéticos: origem e evolução*. 1º ed. Sao Paulo: Sociedade Brasileira de Matemática Aplicada e Computacional.

Ogasawara, Eduardo et al. 2011. “An Algebraic Approach for Data-Centric Scientific Workflows”. *Proc. of VLDB Endowment* 4(12): 1328-1339.

Ogasawara et al. 2009. “Exploring many task computing in scientific workflows”. In *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, Portland, Oregon, USA: ACM, p. 1-10. http://portal.acm.org/citation.cfm?id=1646468.1646470&coll=Portal&dl=ACM&type=series&idx=SERIES371&part=series&WantType=Proceedings&title=SC&CFID=://www.google.com/search?hl=en&CFTOKEN=www.google.com/search?hl=en (Acessado janeiro 28, 2010).

Oliveira, D. et al. 2011. “A Performance Evaluation of X-Ray Crystallography Scientific Workflow Using SciCumulus”. In *IEEE International Conference on Cloud Computing (CLOUD)*, Washington, D.C., USA: IEEE, p. 708-715.

Oliveira et al. 2009. “Using Ontologies to Support Deep Water Oil Exploration Scientific Workflows”. In *IEEE International Workshop on Scientific Workflows*, Los Angeles, California, United States, p. 364-367.

Oliveira, D., F. Baião, e M. Mattoso. 2010. “Towards a Taxonomy for Cloud Computing from an e-Science Perspective”. In *Cloud Computing: Principles, Systems and Applications (to be published)*, Computer Communications and Networks, Heidelberg: Springer-Verlag.

Oliveira, Daniel et al. 2010. “SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows”. In *3rd International Conference on Cloud Computing*, CLOUD ’10, Washington, DC, USA: IEEE Computer Society, p. 378–385. http://dx.doi.org/10.1109/CLOUD.2010.64 (Acessado novembro 2, 2011).

Oliveira, Daniel. 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.

Owonibi, Michael, e Peter Baumann. 2010. “A cost model for distributed coverage processing services”. In *Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems*, HPDGIS ’10, New York, NY, USA: ACM, p. 19–26. http://doi.acm.org/10.1145/1869692.1869696 (Acessado julho 16, 2012).

Patavino, Giuseppina Maria et al. 2012. “Number of recent sexual partners among blood donors in Brazil: associations with donor demographics, donation characteristics, and infectious disease markers”. *Transfusion* 52(1): 151–159.

Pereira, G.C., e N.F.F. Ebecken. 2011. “Combining in situ flow cytometry and artificial neural networks for aquatic systems monitoring”. *Expert Systems with Applications* 38(8): 9626 - 9632.

Porto, Fabio et al. 2011. “A metaphoric trajectory data warehouse for Olympic athlete follow-up”. *Concurrency and Computation: Practice and Experience*.

http://onlinelibrary.wiley.com/doi/10.1002/cpe.1869/abstract (Acessado janeiro 12, 2012).

Raicu, I., I.T. Foster, e Yong Zhao. 2008. “Many-task computing for grids and supercomputers”. In *Proceedings of the Workshop on Many-Task Computing on Grids and Supercomputers*, Austin, Texas, USA, p. 1-11.

Raisanen, Larry, e Roger M. Whitaker. 2005. “Comparison and evaluation of multiple objective genetic algorithms for the antenna placement problem”. *Mob. Netw. Appl.* 10(1-2): 79–88.

Sabino, Ester C. et al. 2011. “Human immunodeficiency virus prevalence, incidence, and residual risk of transmission by transfusions at Retrovirus Epidemiology Donor Study-II blood centers in Brazil”. *Transfusion*: no–no.

Samples, Michael E. et al. 2005. “Parameter sweeps for exploring GP parameters”. In *2005 workshops on Genetic and evolutionary computation*, Washington, D.C., USA: ACM, p. 212-219. http://portal.acm.org/citation.cfm?id=1102308 (Acessado setembro 1, 2009).

Shah, Darshana, e Swapnali Mahadik. 2009. “QoS oriented failure rate-cost and time algorithm for compute grid”. In *Proceedings of the International Conference on Advances in Computing, Communication and Control*, ICAC3 ’09, New York, NY, USA: ACM, p. 264–267. http://doi.acm.org/10.1145/1523103.1523158 (Acessado julho 16, 2012).

Simmhan, Y. et al. 2010. “Bridging the Gap between Desktop and the Cloud for eScience Applications”. In *3rd IEEE Conference of Cloud Computing*, Miami: IEEE Computer Society, p. 474-481.

Singh, Gurmeet, Carl Kesselman, e Ewa Deelman. 2007. “Adaptive pricing for resource reservations in Shared environments”. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, GRID ’07, Washington, DC, USA: IEEE Computer Society, p. 74–80. http://dx.doi.org/10.1109/GRID.2007.4354118 (Acessado julho 16, 2012).

Soanes, Catherine, e Angus Stevenson. 2003. *Oxford Dictionary of English*. 2nd Revised edition. Oxford University Press.

Stamatakis, Alexandros. 2006. “RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models”. *Bioinformatics (Oxford, England)* 22(21): 2688-2690.

Sullivan, Francis. 2009. “Guest Editor’s Introduction: Cloud Computing for the Sciences”. *Computing in Science and Engineering* 11(4): 10-11.

Tanenbaum, Andrew S., e Robbert Van Renesse. 1985. “Distributed operating systems”. *ACM Comput. Surv.* 17(4): 419-470.

Taylor, Ian et al. 2007. “The Triana Workflow Environment: Architecture and Applications”. In *Workflows for e-Science*, Springer, p. 320-339. http://dx.doi.org/10.1007/978-1-84628-757-2_20 (Acessado julho 14, 2008).

Taylor, Ian J. et al. 2007. *Workflows for e-Science: Scientific Workflows for Grids*. 1º ed. Springer.

Toscani, L. V., e P. A. S. Veloso. 2001. *Complexidade de Algoritmos: análise, projetos e métodos*. Porto Alegre, Brazil, Brazil: Sagra Luzzatto.

Travassos, G. H., e M. O. Barros. 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*, Rome, Italy, p. 117-130.

Vaquero, Luis M. et al. 2009. "A break in the clouds: towards a cloud definition". *SIGCOMM Comput. Commun. Rev.* 39(1): 50-55.

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

Viana, Vitor et al. 2011. "SciCumulus-ECM: Um Serviço de Custos para a Execução de Workflows Científicos em Nuvens". In *XXVI SBBD*, Florianópolis, SC, Brasil.

Wall, Larry, Tom Christiansen, e Jon Orwant. 2000. *Programming Perl*. 3rd ed. O'Reilly Media.

Wang, L. et al. 2006. "A Scientific Workflow Framework Integrated with Object Deputy Model for Data Provenance". In *Advances in Web-Age Information Management*, , p. 569-580. http://dx.doi.org/10.1007/11775300_48 (Acessado julho 6, 2008).

Wang, Lizhe et al. 2008. "Scientific Cloud Computing: Early Definition and Experience". In *10th IEEE HPCC*, Los Alamitos, CA, USA: IEEE Computer Society, p. 825-830.

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

Yu, Haifeng, e Amin Vahdat. 2006. "The costs and limits of availability for replicated services". *ACM Trans. Comput. Syst.* 24(1): 70-113.

Zhao, Yong et al. 2007. "Swift: Fast, Reliable, Loosely Coupled Parallel Computation". In *3rd IEEE World Congress on Services*, Salt Lake City, USA, p. 206, 199. http://dx.doi.org/10.1109/SERVICES.2007.63 (Acessado fevereiro 18, 2010).