Universidade Federal
do Rio de Janeiro
Escola Politécnica

AVALIAÇÃO DE CONSULTAS EXECUTADAS SOBRE BASES DE DADOS DE PROVENIENCIA DISTRIBUÍDAS

Edimar Babilon dos Santos

Vanessa Marques de Assis

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores: Marta Lima de Queirós Mattoso
Flavio da Silva Costa

Rio de Janeiro
Abril de 2013

AVALIAÇÃO DE CONSULTAS EXECUTADAS SOBRE BASES DE DADOS DE PROVENIÊNCIA DISTRIBUÍDAS

Edimar Babilon dos Santos
Vanessa Marques de Assis

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO.

Examinada por:

_____

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

_____

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

_____

Prof. Flavio da Silva Costa, M.Sc

_____

Prof. Alvaro Luiz Gayoso Azevedo Coutinho, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
ABRIL de 2013

Babilon, Edimar
Marques, Vanessa

Avaliação de consultas executadas sobre bases de dados de proveniência distribuídas / Edimar Babilon dos Santos, Vanessa Marques de Assis. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2013.

X, 60 p.: il.; 29,7 cm.

Orientadores: Marta Lima de Queirós Mattoso e Flavio da Silva Costa Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Computação e Informação, 2013.

Referências Bibliográficas: p. 56-60.

1. Fragmentação de Dados 2. Workflows Científicos 3. Dados de Proveniência4. Nuvem Computacional I. Mattoso, Marta Lima de Queirós *et al.* II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Computação e Informação. III. Título.

A nossos pais, irmãos e amigos,

Por tudo que representam em nossas vidas.

.

AGRADECIMENTOS
VANESSA MARQUES DE ASSIS

Primeiramente e acima de tudo agradeço às pessoas mais importantes da minha vida: meus pais Leila e Francisco. À minha mãe por ser a pessoa mais inspiradora que tive o prazer de conhecer, que é meu maior orgulho e exemplo a ser seguido e que sempre me apoiou em todos os momentos da minha vida. Ao meu pai, por ter sido a primeira pessoa a me parabenizar por entrar na universidade e ter me apoiado em cada obstáculo e cada conquista, sempre me lembrando do meu potencial. Embora ele não esteja mais aqui para me parabenizar, sei que sempre estará presente nas boas lembranças, nas lições ensinadas e no amor que cultivou.

Aos meus familiares: meus irmãos, meus tios, sobretudo minha amada tia Rosana que sempre esteve ao meu lado, meus primos e meus sobrinhos por sempre terem apostado suas fichas em mim.

Aos meus amigos maravilhosos, sobretudo aqueles que não desistiram de mim mesmo após longos períodos de desaparecimento em épocas de provas.

Ao meu "muito mais que amigo e namorado" Diego, por enxergar em mim um potencial maior até do que eu mesma enxergo e pela paciência e apoio mesmo nos momentos mais estressantes.

À minha querida ECI 2008, e agregados que tanto amo, não só pelas sessões de estudo na biblioteca, MSN e Skype, sem as quais eu dificilmente estaria me formando agora, mas também pelas grandes amizades que tornaram essa jornada muito mais divertida e prazerosa. Um agradecimento especial ao Edimar pela ótima parceria na execução desse projeto.

Agradeço também aos meus professores, sobretudo à profª Marta Mattoso por me inspirar com suas aulas para a escolha do tema deste projeto e por me orientar ao longo dele, ao prof. Rezende pela paciência em tirar dúvidas e sempre ajudar da melhor forma possível e ao prof. Geraldo Xexéo pelas oportunidades oferecidas que me permitiram crescer como profissional. Agradeço também ao Flavio Costa que tornou esse projeto possível sendo sempre muito solícito, revisando os textos e respondendo a centenas de perguntas mesmo num domingo à noite.

E a todos que participaram direta ou indiretamente dessa jornada e de alguma forma contribuíram para que chegasse até aqui, meu muito obrigada.

AGRADECIMENTOS
EDIMAR BABILON DOS SANTOS

Após cinco anos de graduação, pude passar por momentos difíceis e muitos outros momentos felizes. Ao concluir esse ciclo é importante passar por um momento de reflexão e lembrar-se das pessoas que estiveram ao seu lado durante essa caminhada e tornaram possível a concretização desse feito.

Primeiramente gostaria de agradecer a Deus por permitir que essa jornada se completasse. À minha mãe Vera Lucia Babilon e irmã Ane Caroline Babilon dos Santos por serem exemplos de mulheres e participarem tão intensamente na formação do meu caráter, ao meu pai Edemar Natalino dos Santos por seu carinho e atenção e por ser um pai sempre presente na minha vida.

À minha sobrinha Valentina Babilon Guedes de Meira, que em tão pouco tempo já conseguiu preencher um enorme espaço no meu coração. Um agradecimento especial a duas pessoas que certamente foram muito importantes na minha vida minha segunda mãe Maria de Lourdes Caser, e minha avó Maria Sodalina Souza Babilon, que infelizmente não puderam ver esse momento se tornar realidade.

À minha orientadora Marta Lima de Queirós Mattoso, por ser um exemplo de professora e por ter participado de grande parte da minha graduação.

Ao Flavio da Silva Costa por sua valiosa contribuição para a execução desse trabalho

Aos meus amigos, Bernardo Costa, Erika Hoyer, Felipe Horta, Fernando Pinheiro, Lívia Ribeiro, Luiz Alves, Pedro Lemos, Pedro Figueiredo, Silvia Benza, Thiago Ferraz e Thiago Souza pelas longas noites mal dormidas na tentativa de realizar o melhor trabalho, em especial à Vanessa Marques por sua parceria na vida e na execução desse projeto final.

Resumo do Projeto de Graduação apresentação à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

Avaliação de Consultas Executadas sobre Bases de Dados de Proveniência Distribuídas

Edimar Babilon dos Santos
Vanessa Marques de Assis
Abril/2013

Orientadores: Marta Lima de Queirós Mattoso e Flavio da Silva Costa
Curso: Engenharia de Computação e Informação

Devido à cooperação entre centros de pesquisas e à necessidade de alto desempenho computacional, a execução de *workflows* científicos vem sendo cada vez mais realizada em ambientes distribuídos. Sendo assim, uma grande quantidade de dados de proveniência da execução dos workflows é gerada de maneira, também, cada vez mais distribuída. Para conseguirmos manter características fundamentais dos experimentos científicos, tais como sua reprodutibilidade, a informação de proveniência deve ser capturada mesmo tendo o experimento sido executado em ambientes distribuídos. Os dados de proveniência gerados sob a execução de um experimento serão úteis para a sua gerência e também poderão ser usados em futuras execuções, para análises de desempenho ou de confiança dessas novas execuções, onde, por meio de análises de informação histórica, somos capazes de inferir uma série de conclusões. Sendo assim, é importante avaliar como tais dados deverão estar organizados, de maneira a otimizar os tempos das consultas frequentemente realizadas pelos cientistas. Para isso, este projeto propõe uma fragmentação dos dados de proveniência de *workflows* e realiza uma análise de desempenho a fim de avaliar se a fragmentação proposta traz ou não benefícios.
Palavras-chave: *Workflows* Científicos, Fragmentação de Dados, Dados de Proveniência, Nuvem Computacional.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Computer and Information Engineer.

Evaluation of Queries Executed on Distributed Provenance Databases

Edimar Babilon dos Santos

Vanessa Marques de Assis

April/2013

Advisors: Marta Lima de Queirós Mattoso

Flavio da Silva Costa

Major: Computer and Information Engineering

Due to the cooperation between research institutes and to heterogeneous characteristics, the execution of scientific workflow is being increasingly performed on distributed environment. So an increasingly amount of provenance data is being generated on a distributed environment as well. In order to keep fundamental properties of the scientific experiments, like reproducibility, all the provenance data has to be captured by scientists, even when the experiment is executed in a distributed environment. Provenance data generated by an experiment will be useful to manage the experiment and to be used as resource for future executions on performance or trust analysis by using historical data. With that being said, it's important to evaluate how that data should be organized in a way that optimizes the time needed to run queries often used by scientists. In order to do that, this project proposes a fragmentation project of provenance data generated by workflows and analysis the results to evaluate if the goal was reached.

Keywords: Scientific Workflows, Data Fragmentation, Provenance Data, Cloud Computing.

SUMÁRIO

Índice de figuras

Capítulo 1

Introdução

Nos experimentos baseados em simulação, tanto o ambiente quanto os participantes do experimento são executados em ambientes computacionais. Nestes experimentos, simulações, que são normalmente realizadas por múltiplas combinações de programas onde cada um deles possui um conjunto de parâmetros de entrada, são realizadas para validar modelos e hipóteses levantadas pelos cientistas, o que muitas vezes torna necessária a utilização de meios computacionais complexos e de alta capacidade de desempenho.

Para representar o encadeamento dos programas que constituem uma simulação são utilizados *workflows* científicos, que são uma abstração para modelar o fluxo de dados e atividades de um experimento. Em um *workflow* científico, essas atividades são normalmente programas ou serviços que representam algoritmos computacionais sólidos (Barker e van Hemert, 2008).

Devido à crescente complexidade dos experimentos, surgiram os Sistemas de Gerência de *Workflow*s Científicos (SGWfC), que possuem um motor de execução acoplado, responsável por invocar cada um dos programas do fluxo. A utilização desses sistemas torna possível a execução paralela das atividades da cadeia do *workflow*, o que permite reduzir o tempo necessário para a produção do resultado final, melhorando o dia a dia dos pesquisadores.

O volume de dados científicos hoje processados atingiu níveis nunca antes vistos na história (Wallis *et al.* 2010). Os experimentos científicos em larga escala necessitam de recursos computacionais cada vez mais potentes para sua execução e são geralmente realizados por equipes que estão geograficamente dispersas (Gordon *etal.*2008). Para suprir essa necessidade por poder de processamento, a nuvem vem se mostrando uma solução muito atrativa, pois oferece serviços sob demanda e elimina os investimentos iniciais com a aquisição de máquinas de alto desempenho, que podem ser extremamente altos.

A execução de *workflows* científicos vem sendo cada vez mais realizada em ambientes distribuídos. Sendo assim, uma grande quantidade de dados de proveniência é gerada de maneira, também, cada vez mais distribuída. Dados estes que precisam ser capturados e gerenciados pelos cientistas. Por proveniência podemos entender todos os dados que são armazenados com o objetivo de descrever determinado experimento (Costa, 2012). Os dados de proveniência são de extrema importância, pois permitem a reprodução e avaliação de resultados.

Dados de proveniência referentes a um conjunto de experimentos são frequentemente distribuídos fisicamente e, assim como os bancos de dados distribuídos, repositórios de proveniência também podem ser replicados ou fragmentados. Para conseguirmos manter características fundamentais dos experimentos científicos, tais como sua reprodutibilidade, toda informação de proveniência deve ser capturada mesmo tendo o experimento sido executado em um ambiente distribuído. Quando os dados de proveniência estão replicados, é possível encontrar mais de uma réplica idêntica do mesmo repositório em vários locais. Quando fragmentados, existem diversos fragmentos de um repositório distribuídos que quando reunidos reconstroem o

repositório original. Embora a distribuição de repositórios de proveniência possa melhorar os tempos de consulta, a fragmentação dos mesmos gera um grande desafio que é o de encontrar e recuperar dados quando os repositórios estão distribuídos.

Em termos de custos financeiros e tempo de execução é de grande importância ter dados históricos de experimentos que possam fornecer informações úteis acerca da estimativa de recursos a serem alocados na nuvem.

Sendo assim, os dados de proveniência gerados sob a execução de um experimento serão úteis para a sua gerência e também poderão ser usados em futuras execuções, para análises de desempenho ou de confiança dessas novas execuções, onde por meio de análises de informação histórica, somos capazes de inferir uma série de conclusões.

Como dados de proveniência também podem ser utilizados para tomar decisões em tempo real é importante ter consultas com um tempo de resposta pequeno, para que essas não afetem o desempenho da execução do *workflow* como um todo. Dessa forma, é importante avaliar como os dados deverão estar organizados, de maneira a otimizar os tempos das consultas e diminuir a probabilidade de que os dados não estejam disponíveis para consulta. Para isso, este projeto propõe uma fragmentação dos dados de proveniência de *workflows* e realiza uma análise de desempenho a fim de avaliar se a fragmentação proposta traz ou não benefícios.

O projeto está dividido em cinco capítulos, além dessa introdução. O Capítulo 2 tem o objetivo de apresentar os conceitos preliminares para o completo entendimento dos temas abordados. O Capítulo 3 trata do SciCumulus, um mediador para a execução paralela de *workflows* científicos em nuvens computacionais. O Capítulo 4 é

responsável pelas abordagens de fragmentação dos nós, bem como análise dos resultados obtidos. E, por fim, o Capítulo 5 conclui o trabalho.

Capítulo 2

Conceitos Preliminares

Nesse capítulo serão apresentados os principais conceitos utilizados nos demais capítulos do documento. Sendo assim, a Seção 2.1 define o que caracteriza um experimento científico computacional. A Seção 2.2 descreve o ciclo de vida desse tipo de experimento. Os conceitos acerca de *workflows* científicos são apresentados na Seção 2.3. A Seção 2.4 apresenta o conceito de proveniência de dados no contexto de *workflows* científicos. A Seção 2.5 trata do conceito de nuvem computacional. O ciclo de vida de um *workflow* científico na nuvem é tratado na Seção 2.6. O conceito de *cloud activity* é apresentado na Seção 2.7. E, por fim, a seção 2.8 mostra conceitos de fragmentação em banco de dados.

2.1. Experimentos Científicos Baseados em Simulação

Um experimento científico consiste na adoção de uma estratégia concreta a partir do qual os cientistas podem observar resultados com a intenção de provar ou refutar uma hipótese, ou seja, "um teste executado sob condições controladas, que é realizado para demonstrar uma verdade conhecida, examinar a validade de uma hipótese, ou determinar a eficácia de algo previamente não explorado" (Soanes e Stevenson 2003). Nos últimos anos, a utilização de sistemas de *workflows* para a execução de experimentos científicos baseados em simulações computacionais vem aumentando consideravelmente (Mattoso et al. 2010).

Em muitos casos, os experimentos científicos são formados por um conjunto de programas que devem ser executados de forma encadeada, produzindo e consumindo uma grande quantidade de dados (Costa, 2012). A execução desses experimentos frequentemente pode demandar muito tempo e recursos, sendo necessária a utilização de técnicas de paralelismo e ambientes de processamento de alto desempenho (PAD).

O conceito de experimento científico engloba o conceito de *workflow*, e não podem ser tratados como um sinônimo. A execução de um *workflow* pode ser vista como um conjunto de ações controladas do experimento (Oliveira, 2012).

2.2. Ciclo de Vida do Experimento Científico

Nesta seção será apresentado um modelo de ciclo de vida para um experimento cientifico proposto em (Mattoso et al. 2010c). A figura 1 apresenta o ciclo de vida de uma modelo científico em larga escala.

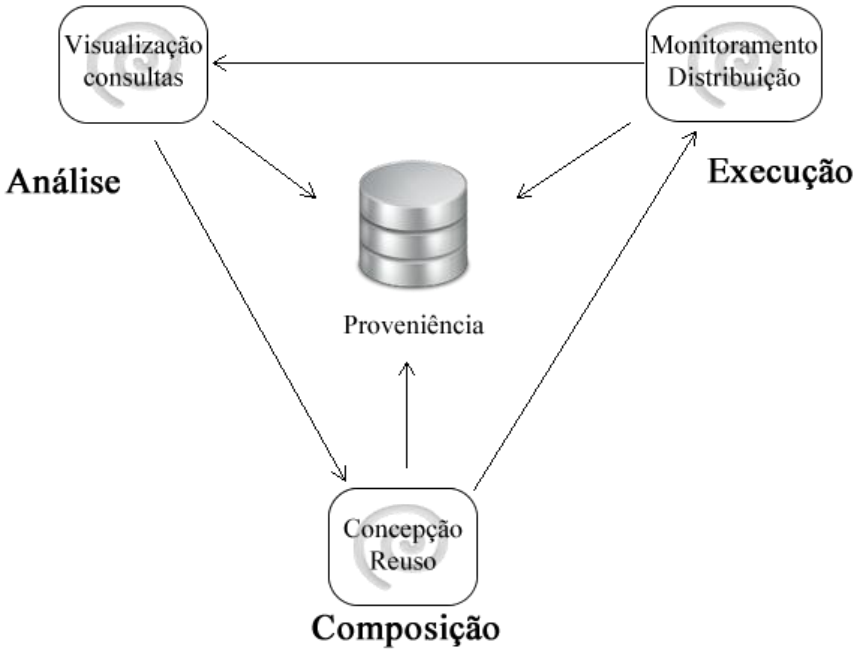

Figura 1 Ciclo de vida de um experimento científico adaptado de (Mattoso *et al.*2010c)

É possível observar na figura a existência de algumas etapas que podem ser percorridas pelos cientistas ao longo do seu experimento científico. As fases percorridas são: execução, composição e análise. Cada fase possui um subciclo que pode ser percorrido em momentos distintos.

A fase de composição tem por intuito criar e estruturar o experimento científico, ela também se responsabiliza por gerar uma sequência lógica de atividades. Essa etapa pode se decompor em outras duas subfases: concepção e reuso. Enquanto a concepção tem por finalidade a criação do experimento o reuso tem por propósito recuperar e adaptar o experimento para uma nova finalidade.

Já na fase de execução, que possui uma maior dependência computacional e por isso é a fase em que os esforços devem ser concentrados, as especificações do *workflow* se tornam concretas podendo então ser executado por um SGWfC. Essa fase também se

subdivide em outras duas: distribuição e monitoramento. A subfase de distribuição está relacionada com a necessidade da execução de atividades do *workflow* em ambientes de PAD, principalmente devido a necessidades de desempenho. A subfase de monitoramento está relacionada à necessidade de verificar periodicamente o estado atual da execução do *workflow*, uma vez que este pode executar por um longo tempo (Oliveira, 2012).

A fase de análise tem como foco principal estudar os dados gerados nas etapas de composição e execução. Esta fase assim como as outras pode ser decomposta em duas subfases: visualização e consulta. Ao analisar os dados de proveniência, os cientistas podem se deparar com a situação em que a hipótese analisada é validada ou a que a hipótese é refutada.

2.3. *Workflow* Científico

Um *workflow* científico pode ser definido como a especificação formal de um processo científico que representa os passos a serem executados em um determinado experimento científico (Deelman *et al*. 2009). Os *workflows* fornecem a abstração necessária para a especificação dos experimentos científicos de maneira estruturada, isso permite representar um *workflow* através de um artefato, um programa ou script executável, para assim poder ser gerenciado por um Sistema Gerenciador de *Workflow* científico (SGWfC).

Fazem parte de um *workflow* científico os seguintes componentes: atividade, porta, relacionamento e *sub-workflow*. A atividade é responsável por consumir e produzir dados, as portas descrevem os dados que são consumidos e produzidos em cada

atividade. Ainda na especificação das portas é interessante avaliar se uma porta comporta-se apenas para consumo de dados (porta de entrada), ou produção de dados (porta de saída), ou ainda para os dois.

Um relacionamento é a parte do *workflow* que permite o encadeamento lógico das atividades através da definição das portas de entrada e saída, estabelecendo assim um fluxo de dados. A figura 2 apresenta um esquema de um *workflow* científico proposto por (Sousa, 2011).Analisando a figura é possível observar a existência de duas atividades(atividades *A* e *B*). A atividade *A* apresenta três portas de entrada (portas *x*, *y* e *z*) e duas portas de saídas (portas *a* e *b*). Já a atividade *B* apresenta duas portas de entrada (portas *W* e *d*) e duas portas de saída (portas *c* e *d*). É importante salientar o fato de duas portas terem o nome igual (porta *d*), isso permite enfatiza a presença de uma porta do tipo entrada e saída. Ainda é possível observar os relacionamentos presentes entre as atividades *A* e *B* (relacionamentos *rel1* e *rel2*).

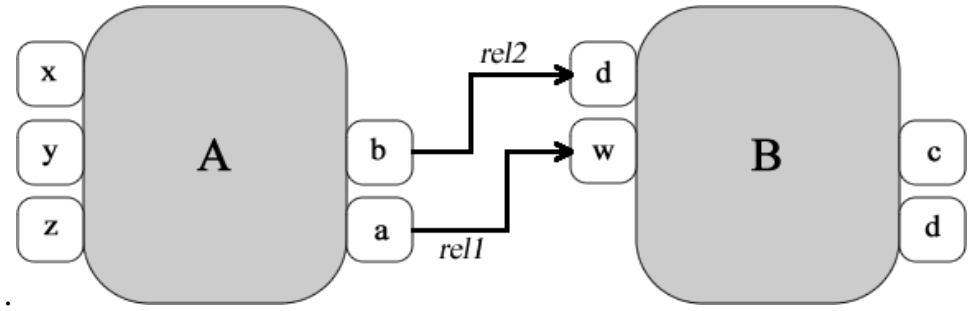

Figura 2. Esquema geral de um *workflow* científico. Adaptado de (Sousa, 2011).

2.4. Proveniência de Dados

No dicionário Michaelis da língua portuguesa o termo proveniência é definido como "*s.f.* (*lat provenientia*, de *provenire*) 1. Lugar de onde alguma coisa provém, emana ou se deriva. 2. Fonte, origem, procedência.".

E é a partir desta definição que caracterizaremos o conceito de proveniência: como a fonte, origem ou procedência de uma informação em um experimento científico. Essa origem consiste em todo o histórico que envolve um determinado dado do experimento científico, como quem criou aquele dado, quando aquele dado foi criado, por que processos aquele dado passou, que resultados aquele dado gerou, etc.

E é a partir da análise desse histórico, ou seja, da sequência de passos que levou a um determinado resultado, é que se pode avaliar os procedimentos utilizados, identificar as entradas de parâmetros utilizadas e, em alguns casos, reproduzir os resultados obtidos.

Goble *et al.* (2003) resumem as diversas funcionalidades para as informações de proveniência da seguinte maneira:

- Garantia de qualidade dos dados: informações de proveniência podem ser utilizadas para estimar a qualidade e a confiabilidade dos dados baseando-se na origem dos dados e suas transformações.
- Auditoria dos caminhos: os dados de proveniência podem traçar rotas dos dados, determinar a utilização de recursos e detectar erros na geração de dados.
- Verificação de atribuição: mantém controle sobre as informações do dono do experimento e seus dados.
- Informacional: permite realizar consultas baseadas nos descritores de origem para a descoberta de dados, além de prover o contexto necessário para interpretar os mesmos.

De maneira geral, a proveniência pode ser caracterizada em duas formas: prospectiva e retrospectiva (Cruz *et al.* 2009, Freire *et al.* 2008b). A proveniência prospectiva captura a especificação do que levou à geração de um determinado produto. No contexto de *workflows* científicos, a proveniência está interessada em capturar os dados relativos à estrutura do *workflow*, bem como as configurações de ambiente utilizadas para executá-lo.

Por outro lado, a proveniência retrospectiva foca em capturar dados, juntamente com os descritores gerados a partir da execução de um determinado processo, ou seja, de um determinado *workflow* científico. Entre os dados de proveniência que englobam a proveniência retrospectiva estão: tempos de início e fim de execução, arquivos produzidos, erros que ocorreram, informações de desempenho de atividades, entre outros.

Outro conceito importante é o conceito de granularidade dos dados de proveniência capturados. Buneman e Tan (2007) classificam os níveis de detalhe da proveniência capturada em dois tipos: "grão fino" (do inglês *fine grain*) e "grão grosso" (do inglês *coursegrain*). A proveniência de "grão grosso" diz respeito ao registro histórico de um conjunto de dados, já a proveniência de "grão fino" se refere ao registro da linguagem de um item especifico de dados.

Simmhan *et al.* (2005) classificam sistemas de gerencia de proveniência de acordo com:

- Aplicação: a finalidade e utilização das informações de proveniência, como reprodutibilidade, avaliação da qualidade de dados, trilhas de auditoria e proteção de propriedade intelectual.

- Conteúdo: os aspectos da proveniência que são coletados, como histórico de derivação de dados (proveniência retrospectiva) ou a especificação do *workflow* (proveniência prospectiva).
- Representação: o modelo de dados utilizados para descrever a proveniência.
- Armazenamento: modo de armazenamento físico das informações, como bancos de dados relacionais ou arquivos de texto;
- Disseminação: formas de acesso às informações de proveniência, como linguagens de consulta.

Para armazenar a proveniência retrospectiva devem ser utilizados, preferencialmente, modelos de dados que se baseiem na mais recente recomendação do *Open Provenance Model* (OPM) (Mareau *et al.* 2008), ou de sua evolução: o modelo PROV (Moreau e Missier 2011, Moreau *et al*. 2011), o qual também propõe uma representação genérica de proveniência. A principal diferença entre os dois é a representação. Enquanto o OPM utiliza representação por grados, o PROV utiliza a representação do modelo ER. Ambos expressam as relações casuais entre Processos, Agentes, Artefatos e Papéis existentes em *workflows*.

A grande vantagem da utilização desses tipos de recomendações é a interoperabilidade de descritores de proveniência vindos de ambientes heterogêneos, independentemente da tecnologia e dos Sistemas de Gerenciamento de *Workflows* Científicos utilizados.

2.5. Nuvem Computacional

A computação em nuvem (do inglês *Cloud Computing*) (Kim *et al.* 2009, Marinos e Briscoe 2009, Napper e Bientinesi 2009, Vaquero *et al.* 2009, Wang *et al.* 2008) surgiu como um novo paradigma de computação distribuída, onde serviços baseados na *Web* têm como objetivo permitir que diferentes tipos de usuário obtenham uma grande variedade de recursos de *software* e *hardware*.

Vaquero *et al.* (2009) define nuvem como uma grande gama de recursos virtualizados que são facilmente usáveis e acessíveis (como hardware, plataformas de desenvolvimento e/ou serviços). Esses recursos podem ser reconfigurados para se ajustar a uma demanda variável, permitindo uma utilização ótima de recursos. Esse conjunto de recursos normalmente é disponibilizado por um provedor em um modelo pague-pelo-uso (do inglês *pay-per-use model*).

É possível observar que, graças ao conceito de nuvem, a computação mudou completamente em relação ao que se conhecia há alguns anos. Isso porque os programas e dados passaram de computadores *desktop* para a nuvem, o que permite com que usuários acessem programas, documentos e dados de qualquer computador que tenha uma conexão com a internet.

Como a própria definição apresentada indica, a computação em nuvem representa um grande potencial para apoiar experimentos científicos que necessitem de ambientes de processamento de alto desempenho. 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 com consequente manutenção dos mesmos, além de questões como atualizações constantes de *software* e *hardware*.

Além disso, as nuvens tornam possível que uma grande quantidade de dados utilizados e gerados pelo experimento seja mais facilmente compartilhada para a análise, dentro e fora da nuvem, democratizando o acesso a resultados científicos, embora a transferência de dados em nuvens seja um problema em aberto.

Em Oliveira *et al.* (2010) é proposta uma taxonomia para o campo de computação em nuvem a partir de uma perspectiva de e-Science. Essa taxonomia classifica as características do domínio de computação em nuvem baseado em diferentes aspectos: características arquiteturais, de modelo de negócio, de infraestrutura tecnológica, de privacidade, de normas, de precificação, de orientação e de acesso.

Em relação ao Modelo de Negócio, as abordagens são geralmente classificadas em três categorias (NIST 2010): Software como Serviço (do inglês *Software as a Service* ou SaaS), Plataforma como Serviço (do inglês *Platform as a Service* ou PaaS) e Infraestrutura como Serviço (do inglês *Infrastructure as a Service* ou IaaS), criando um modelo chamado SPI (do inglês *Service-Platform Infrastructure*) (Youseff *et al.* 2008, Zhu e Wang 2008).

Em SaaS o *software* é disponibilizado através da *Web* para uso comercial ou livre como um serviço sob demanda. Em IaaS o provedor oferece uma infraestrutura, como um *cluster*, para o usuário final através da *Web*. Em IaaS, o usuário final normalmente é responsável por configurar o ambiente para usar. Já PaaS é a entrega de uma plataforma de programação como um serviço, o que facilita a implantação de programas em nuvem.

Já em relação ao aspecto da privacidade, de acordo com o modelo, classificam-se ambientes em nuvem em três categorias: privados, públicos e mistos. Em nuvens públicas, os recursos são dinamicamente disponibilizados na Internet para qualquer usuário. Nas nuvens privadas, os dados são acessíveis apenas dentro de uma determinada corporação ou uma instituição cientifica. Já uma nuvem mista é composta por múltiplas nuvens, sejam elas públicas ou privadas.

Em relação à precificação, existem ambientes gratuitos e os baseados no modelo pague-pelo-uso, mencionado anteriormente. No modelo gratuito, os recursos estão disponibilizados gratuitamente para usuários autorizados a utilizá-los. Já o modelo pague-pelo-uso aplica-se em ambientes de nuvem comerciais e científicos. Os cientistas pagam pelo uso da nuvem da mesma forma que usuários comerciais o fazem.

Sobre as características arquiteturais, é importante ressaltar aquelas que são desejáveis em um ambiente de nuvem. Algumas dessas características são: a heterogeneidade, pois uma nuvem deve apoiar a agregação de *hardware* heterogêneo e vários tipos de recursos de *software*, como acontece com *workflows* científicos; a virtualização, que permite que usuários possam se beneficiar da mesma infraestrutura física usando instâncias independentes; e a elasticidade, que permite o aumento ou a diminuição, sob demanda do número de máquinas virtuais em um ambiente de nuvem.

Os métodos de acesso são classificados de acordo com a forma com que o usuário acessa os recursos, seja por *browser* ou por unidades desenvolvidas especialmente para esse fim. Os padrões tratam das normas preestabelecidas as quais um ambiente de nuvem segue. Já a parte de orientação pode ser centrada em tarefa, como quando um *software* é disponibilizado na nuvem como um serviço, pois este está orientado à tarefa

a qual deve executar, ou em usuário, como quando a infraestrutura é fornecida como um serviço ao cientista, onde ele pode ter o controle dos processos e escolher os programas, aplicativos e dados a serem utilizados.

2.6. Ciclo de Vida de *Workflows* na 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. A Figura 3 representa as sete principais etapas que compõem o ciclo de vida de um *workflow* cientifico executado em nuvens.

Figura 3. Ciclo de vida de um *workflow* científico executado em nuvens de computadores adaptado de (Oliveira, 2012)

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. Isso acontece porque os *workflows* dependem do provimento de infraestrutura por parte do provedor de nuvem e, consequentemente, este ambiente não está completamente configurado para a execução do *workflow*. Além disso, existem também as questões como transferência dos dados e o posterior download dos mesmos para fins de análise. É importante ressaltar que todas as ações estão associadas com os dados de proveniência, ou seja, ou produzem dados ou os consomem de alguma maneira, ou ambos.

Os programas necessários para a execução do *workflow*, bem como as dependências de dados e em qual área das nuvens os mesmos se encontram são informados na fase de composição de *workflow*, onde os cientistas elaboram tais especificações. Esta especificação impacta diretamente na configuração do ambiente. 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*.

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.

2.7. O Conceito de Unidades de Trabalho na Nuvem

Quando se deseja executar uma atividade em paralelo na nuvem, uma abordagem de paralelismo que pode ser utilizada e que não impacta em reescrever código do programa invocado é fazer com que cada valor do parâmetro combinado com um dado a ser consumido seja executado em paralelo em uma máquina virtual diferente. Como o ambiente de nuvem é distribuído e cada máquina virtual é independente da outra, 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 utilizada.

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

As *cloud activities* são regidas por uma Máquina de Estados Finita específica (do inglês *Finite State Machine* ou FSM). Esta FSM possui cinco principais estados: Criado, Pronto, Ativado, Bloqueado e Terminado. Quando uma *cloud activity* é criada, ela recebe o estado de "Criado". Automaticamente, se não houver erro, recebe um estado "Pronto". Caso contrário, se, por alguma razão, uma *cloud activity* não pode imediatamente iniciar a execução, ele recebe um estado "Bloqueado". Um exemplo é

quando uma *cloud activity* tem uma dependência de dados com outra *cloud activity* que não tenha sido executada ainda. A *cloud activity* pode continuar para o estado "Ativado" se ela pode ser executada normalmente. Uma vez que termina a execução de uma *cloud activity*, ela recebe o estado "Terminado". Quando a *cloud activity* tem um problema como um programa corrompido ou dados corrompidos ou inexistentes, e não pode ser executada, então vai diretamente do estado "Ativado" para o estado "Terminado".

2.8. Fragmentação em Bancos de Dados

Esta seção trata da fragmentação em bancos de dados, pois quando se fala em bancos de dados distribuídos (livro Ozsu e Valduriez), é importante entender como essa distribuição é dada. Como a base de dados de proveniência de execução de workflows científicos é frequentemente armazenada em bancos de dados relacionais, o uso de técnicas de projeto de distribuição nesses bancos se mostra adequada para gerenciar grandes volumes de dados. Em geral, um banco de dados distribuído se dá por um conjunto de fragmentos da base de dados que estão espalhados fisicamente e que, quando reunidos, tem a capacidade de reconstruir o banco de dados. Dessa forma, é possível aumentar o nível de paralelismo, permitindo o acesso simultâneo a diferentes fragmentos. Além disso, uma mesma consulta pode ser dividida em partes, onde essas partes são executadas simultaneamente em nós diferentes da rede para então retornar os resultados e formar o resultado final. Quando se trata de um volume de dados muito grande, sua distribuição pode tornar as consultas mais rápidas, pois possivelmente serão executadas em paralelo e sobre partes menores, que constituem um menor volume de dados a ser acessado. Existem duas estratégias fundamentais de fragmentação sobre

bases de dados que seguem o modelo relacional: a fragmentação horizontal e a fragmentação vertical. É possível também combinar essas duas estratégias fundamentais, gerando uma fragmentação híbrida.

Independentemente da estratégia utilizada, é necessário garantir que a fragmentação mantenha a correção da base de dados, por meio de três regras, a saber, a completude, a reconstrução e a disjunção dos fragmentos.

A completude garante que se uma relação R é decomposta entre os fragmentos F_R= {R_1, R_2, R_3, ..., R_n}, então cada item encontrado em R, deve necessariamente ser encontrado em um R_i.

A reconstrução uma relação R garante que se R é decomposta entre os fragmentos F_R= {R_1, R_2, R_3, ..., R_n}, deve existir um operador relacional que consiga reconstruir R a partir de suas partes R_i, para todo $R_i$em F_R.

A disjunção garante que se uma relação R é decomposta horizontalmente entre os fragmentos F_R= {R_1, R_2, R_3, ..., R_n}, então qualquer item d_i encontrado em R_j, não pode ser encontrado em nenhum outro fragmento R_k, para $k \neq j$. Esse critério garante que os fragmentos horizontais são disjuntos. Já na fragmentação vertical, como a chave primária normalmente é repetida em cada um dos fragmentos, para que se mantenha a reconstrução de R, então a disjunção na fragmentação vertical deve ser garantida apenas para atributos que não são chaves primárias. Dessa forma, um atributo presente em um fragmento não deve estar presente em nenhum outro fragmento, a menos que o mesmo seja uma chave primária da tabela.

A seguir, serão apresentadas as estratégias fundamentais de fragmentação.

2.8.1. Fragmentação Horizontal

A fragmentação horizontal divide uma relação em relação às suas tuplas, fazendo com que cada fragmento possua um subconjunto de tuplas da relação completa.

Cada fragmento resultante de uma fragmentação horizontal é representado por uma operação de seleção realizada na relação original. Sendo assim, cada fragmento Ri de uma relação R é representado por $Ri = \sigma Hi(R)$, $1 \leq i \leq w$ onde Hi representa uma fórmula utilizada na operação de seleção para obter o fragmento Ri, também chamada de predicado de fragmentação.

Existem diversas técnicas para determinar como será realizada a fragmentação dos dados. Em geral, analisam-se os tipos de consulta e a origem das mesmas, bem como a frequência de cada uma delas.

Supondo que se deseja fragmentar a seguinte tabela abaixo:

Projeto

Cod_proj	Orçamento	Localização
P01	100.000	Rio de Janeiro
P02	200.000	Nova York
P03	150.000	Rio de Janeiro
P04	600.000	Tóquio
P05	30.000	Nova York
P06	150.000	Tóquio

Figura 4. Exemplo de tabela a ser fragmentada horizontalmente

A forma como a tabela será fragmentada dependerá diretamente de como os dados são acessados. Em geral, não é possível analisar cada um dos possíveis predicados utilizados pelos usuários para acessar os dados, pois pode existir uma grande variedade dos mesmos. No entanto, é importante analisar pelo menos os mais importantes. Em

geral, assume-se a que é verdadeira a afirmação de que as 20% consultas mais realizadas, correspondem a 80% do total dos acessos (Wiederhold, 1982).

Assumindo então que a maior parte dos acessos à tabela Projeto é feita através da localização dos mesmos, uma possível proposta de fragmentação é representada pela figura abaixo.

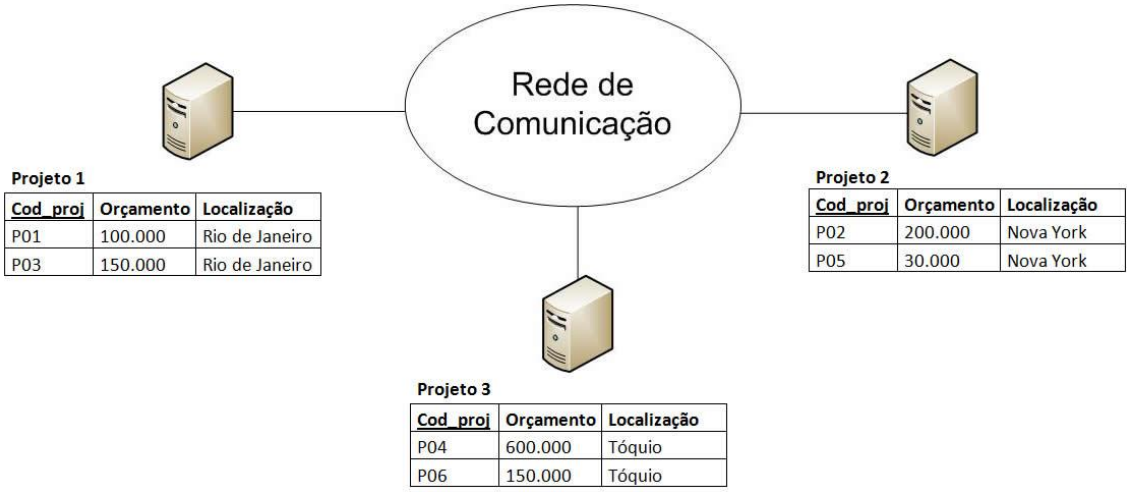

Figura 5. Exemplo de fragmentação horizontal

2.8.2. Fragmentação Vertical

A fragmentação vertical de uma relação R produz fragmentos R1, R2, ..., Rn, onde cada um deles contém um subconjunto dos atributos de R, juntamente com a chave primária de R. O objetivo da fragmentação vertical é fragmentar uma tabela em pedaços menores, para que assim as consultas realizadas pelos usuários sejam executadas em apenas um fragmento, otimizando o uso do espaço em memória. Um projeto de fragmentação ideal é aquele que minimiza o tempo de execução das solicitações feitas pelos usuários em cada fragmento.

A fragmentação vertical é muito utilizada também em bancos de dados centralizados com o objetivo de permitir que as consultas sejam executadas em pedaços menores das tabelas, necessitando assim de um número menor de acessos ao disco e à memória (Navathe et al., 1984).

Em geral, construir uma proposta de fragmentação vertical é uma tarefa potencialmente mais complexa do que a de construir uma proposta de fragmentação horizontal. Isso é dado pelo grande número de possibilidades de escolha disponíveis no caso da fragmentação vertical.

Um conceito importante no projeto de uma fragmentação vertical é o de afinidade entre os atributos, pois um fragmento terá um conjunto de atributos que são frequentemente acessados em conjunto.

Para isso, analisam-se todas as consultas mais frequentes, lembrando-se da regra de que 80% dos acessos correspondem a 20% das consultas, e verificam-se quais atributos são utilizados por cada uma delas. A partir dessa verificação, é possível montar uma matriz chamada de Matriz de Uso de Atributos que indica a utilização de cada um dos atributos. E é a partir desta matriz, juntamente com o número de acessos aos atributos por uma determinada aplicação e com a frequência de acessos daquela aplicação que se define o grau de afinidade entre dois atributos. Para um exemplo de fragmentação vertical, considera-se a tabela abaixo:

Funcionário

Cod_func	**Nome**	**Data_Nascimento**	**Função**	**Salário**
F01	José	30/12/1985	Vendedor	1500
F02	João	24/03/1990	Atendente	1400
F03	Ana	12/04/1981	Gerente	1700
F04	Maria	13/06/1989	Publicitário	1600

Figura 6 - Exemplo de tabela a ser fragmentada verticalmente

Supondo que através da análise das consultas mais frequentes, verificou-se que os atributos Nome e Data_Nascimento possuem um alto grau de afinidade, assim como os atributos Função e Salário. Sendo assim, uma possível proposta de fragmentação é representada pela figura abaixo.

Funcionário 1

Cod_func	Nome	Data_Nascimento
F01	José	30/12/1985
F02	João	24/03/1990
F03	Ana	12/04/1981
F04	Maria	13/06/1989

Funcionário 2

Cod_func	Função	Salário
F01	Vendedor	1500
F02	Atendente	1400
F03	Gerente	1700
F04	Publicitário	1600

Figura 7. Exemplo de fragmentação vertical

2.8.3. Fragmentação Híbrida

Uma fragmentação chamada de híbrida, mista ou aninhada, é aquela em que as estratégias da fragmentação vertical e da fragmentação horizontal se combinam. Nesse caso, uma fragmentação vertical pode ser seguida por uma horizontal, ou vice versa. A figura abaixo ilustra o processo de fragmentação horizontal, seguido pelo de fragmentação vertical.

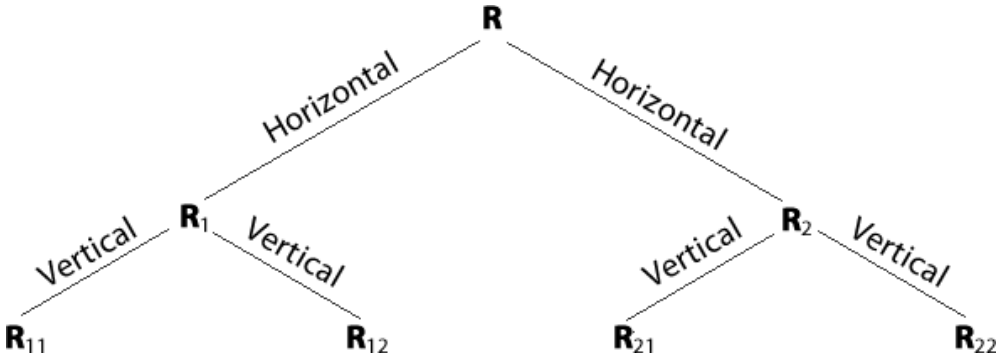

Figura 8. Árvore que representa um exemplo de fragmentação híbrida

Capítulo 3

SciCumulus

A principal função do SciCumulus é possibilitar que os SGWfC consigam gerenciar a execução paralela de *cloud activities* de *workflow*s científicos utilizando a infraestrutura da nuvem. Ele permite a distribuição das *cloud activities* em diferentes máquinas virtuais possibilitando assim a execução em paralelo, além de fornecer um controle dos recursos durante a execução do *workflow* e mecanismos de tolerância a falhas.

3.1. Arquitetura

O SciCumulus segue uma arquitetura de quatro camadas (Oliveira et al. 2010b, 2010c, 2011c, 2011a):

Camada Cliente: Responsável por intermediar a relação entra a nuvem e o SGWfC. Os componentes dessa camada ficam na máquina do cliente, fornecendo assim uma interface para os cientistas submeterem as atividades do *workflow* para serem executadas em paralelo.

Camada de Distribuição: Tem como principal função gerenciar a execução paralela das *cloud activities* em uma ou mais maquinas virtuais instanciadas.

Camada de Execução: Essa camada se responsabiliza pela execução dos programas encapsulados nas *cloud activitie*s também por capturar os dados de proveniência

Camada de Dados: Armazena os dados de entrada, assim como os dados de proveniência gerados e produzidos durante a execução paralela das cloud activities. Essa camada também se responsabiliza por coletar informações do ambiente.

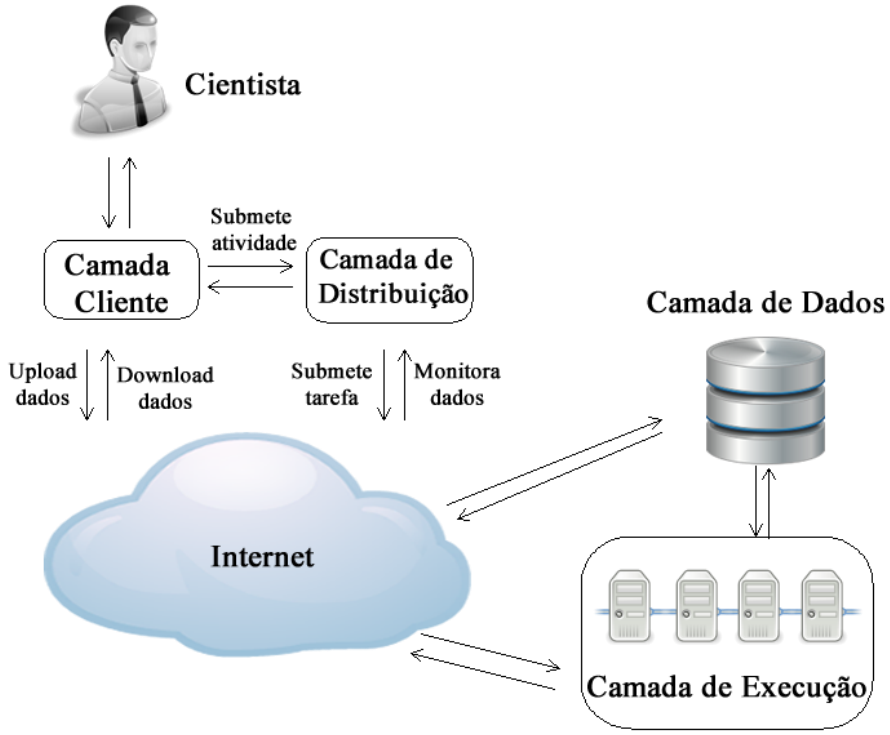

Figura 9. Arquitetura do SciCumulus– adaptado de (Costa et al.2013)

A ideia desse modelo é blindar os cientistas da complexidade envolvida no gerenciamento das cloud activities em paralelo. É necessária apenas uma configuração inicial e a partir daí os cientistas podem contar com uma poderosa ferramenta para execução de seus experimentos e análise dos dados de proveniência.

A arquitetura do SciCumulus pode ser teoricamente aplicada em qualquer ambiente de nuvem e acoplada a qualquer SGWfC (Oliveira, 2012). A figura abaixo fornece um

detalhamento melhor da arquitetura do SciCumulus, mostrando os componentes envolvidos em cada camada.

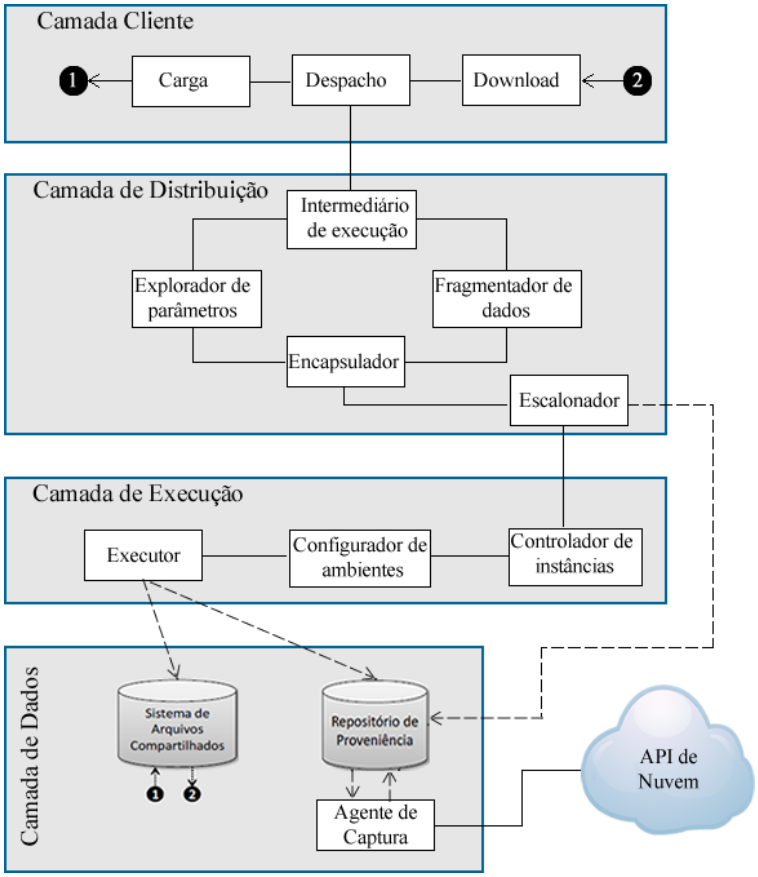

Figura 10. Arquitetura detalhada do SciCumulus. Adaptado de (Oliveira, 2012)

A camada cliente possui três componentes principais: o componente de carga, componente de despacho e o componente de download. Esses componentes são executados na fase de configuração do ciclo de vida do *workflow* executado na nuvem. O componente de carga tem a responsabilidade de levar os dados para a nuvem, enquanto que o de download retira os dados das maquinas virtuais, esses dois componentes são preferencialmente utilizados quando os dados de entrada possuem tamanho pequeno ou médio e produzem saída de tamanho pequeno ou médio, pois no caso de possuir uma entrada grande ou uma saída grande haverá um grande consumo de

banda de transferência de dados. O componente de despacho cria um arquivo, que pode ser um XML, com as informações da configuração da execução distribuída, esse componente é responsável por se comunicar com a camada de distribuição iniciando assim o processo de execução das tarefas na nuvem.

A camada de distribuição recebe o arquivo de configuração da camada cliente e inicia um gerenciamento das tarefas que contêm os programas a serem executados, a estratégia paralela que deve ser seguida e os valores de parâmetros e dados de entrada que servem de insumo para ela. Essa camada possui cinco componentes: O intermediário de execução, o explorador de parâmetros, o fragmentador de dados, o encapsulador e o escalonador. O intermediário de execução recebe os dados fornecidos pelo componente despacho da camada cliente e inicia a execução paralela do *workflow*, esse componente também é responsável por invocar o componente explorador de parâmetros e fragmentador de dados, isso depende da estratégia adotada pelo cientista. O explorador de parâmetros faz uma manipulação nos parâmetros recebidos na camada cliente e o fragmentador de dados atua na fragmentação dos dados de entrada podendo gerar subconjuntos que são distribuídos por varias máquinas virtuais. O encapsulador gera as tarefas a serem transferidas para as máquinas virtuais e o escalonador defina quais máquinas virtuais receberão uma tarefa a ser executada.

A camada de execução possui três componentes: O controlador de instâncias, o configurador de ambientes e o executor. O controlador de instâncias faz a comunicação entre a camada de distribuição e a de execução além de ser responsável por controlar o fluxo nas máquinas instanciadas. O configurador de ambiente cria o ambiente para a execução do *workflow*, definindo as variáveis e criando diretórios para controlar a

execução, já o executor chama a aplicação a ser executada coletando os dados de proveniência e os armazenando em um repositório na nuvem.

Por fim a camada de dados que possui dois componentes: o sistema de arquivos compartilhados e o repositório de proveniência. O sistema de arquivos compartilhados contém os dados de entrada que foram utilizados por diferentes tarefas durante a execução paralela e o repositório de proveniência possui os dados de proveniência (Freire et al.,2008) coletados durante a execução do experimento.

3.2. Modelo de Proveniência

O SciCumulus assim como o seu modelo de proveniência foi projetado para operar na nuvem, é possível obter informações do experimento científico que esta sendo executado e do ambiente de execução através da compreensão do modelo de proveniência do SciCumulus. Esse modelo de proveniência é baseado no Open Provenance Model (OPM) (Moreau et al. 2008a, 2008b). O OPM foca na interoperabilidade entre os dados de proveniência de diversos SGWfC, e representa as relações entre processos, agentes, artefatos e papéis envolvidos na execução de um *workflow* científico.

AFigura 11apresenta um diagrama de classes da Unified Modeling Language (UML) do modelo de proveniência do SciCumulus. É importante observar a existência de três partes principais

- Elementos que representam os *workflow*s executados na nuvem e os artefatos consumidos e produzidos

- Elementos que representam informações sobre a nuvem
- Elementos que representam informações sobre o monitoramento do SciCumulus

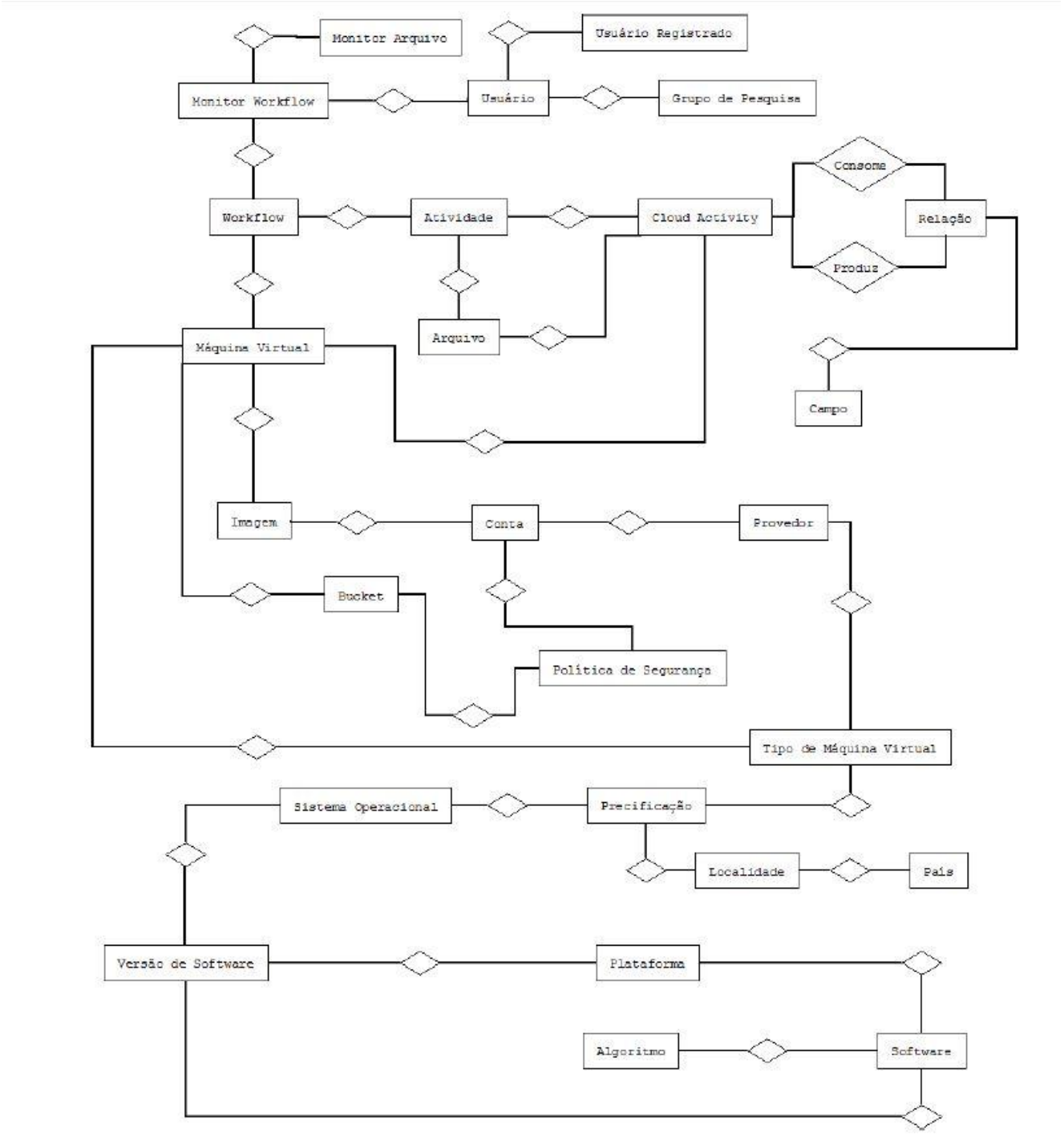

Figura 11.Modelo conceitual do repositório de proveniência do SciCumulus (Oliveira, 2012)

3.3. Detalhes de Implementação

O SciCumulus, utilizado para realização deste projeto final, foi implantado no ambiente Amazon EC2. A linguagem utilizada no desenvolvimento do SciCumulus foi o Java Versão 6 Update 15. O SciCumulus utiliza o MPJ para criação de um cluster virtual na nuvem, o MPJ adota a técnica de paralelismo aninhado que mistura a distribuição entre processos pela utilização de troca de mensagens em MPI e execução em vários núcleos das máquinas virtuais através da utilização de threads, ao utilizar o MPJ se torna necessário definir o nó principal, ou seja, a máquina virtual marcada como rank0 , esse número é usado para identificar um processo particular em um conjunto de processos em execução.

O SciCumulus inicialmente consulta o repositório de proveniência para descobrir as imagens, que contém os metadados do sistema de arquivos e os códigos de inicialização do sistema operacional. Para gerenciar o cluster na nuvem, é necessária a utilização de um pacote de software chamado Vappio. Esse pacote de software fornece um serviço Web para iniciar (vp-add-instances), buscar (vp-search-instances), descrever (vp-describe-instances) e encerrar (vp-terminate-instances) máquinas virtuais na nuvem. O Vappio possui uma lista de IPs das maquinas virtuais instanciadas e essa lista serva para atualizar o arquivo machines.conf sobre quais maquinas virtuais estão disponíveis para utilização.

A conexão desktop com o ambiente de nuvem é feita através da utilização do protocolo SSH, permitindo assim enviar e receber os arquivos de dados diretamente para um sistema de arquivos na nuvem. É interessante saber que os componentes de carga e download da camada cliente se conectam com a máquina virtual em que os

componentes de distribuição e execução foram instalados. Os componentes da camada cliente podem ser implantados em diferentes tipos de SGWfC, um exemplo de SGWfC que pode ser citado é o VisTrails.

Capítulo 4

Fragmentação de Nós

Neste capítulo, apresentaremos os detalhes relevantes da base de dados de proveniência do SciCumulus, a fim de entender o contexto em que ela está inserida e desenvolver uma estratégia de fragmentação que favoreça o tempo de execução de consultas no cenário apresentado e esteja mais próxima a realidade distribuída dos experimentos científicos de um modo geral.

4.1. Base de Dados

A Base de dados de proveniência do Scicumulus é constituída por 25 tabelas, dentre elas: ativação, atividade, *workflow*, arquivo, grupo, campo, relação, usuários etc. Como tratar de cada uma das tabelas tornaria este texto excessivamente extenso e, considerando que nem todas essas tabelas são relevantes no estudo realizado por este projeto, serão consideradas apenas as tabelas de maior relevância para os cientistas no contexto desta análise.

Sendo assim, primeiramente será analisada a tabela referente a um *workflow*, tendo em vista que todos os dados inseridos e todas as buscas realizadas giram em torno de um determinado *workflow* ou de um conjunto de *workflows* que interessa ao cientista de

alguma forma. A Figura 12representa a tabela, juntamente com os atributos que a constituem.

Figura 12 - Tabela *Workflow*

A tabela de nome *hworkflow* representa um *workflow* na base de dados. Esta tabela possui cinco atributos. O atributo *wkfid* é a chave primária da tabela, representada por um número inteiro. O atributo localidade contém o local onde o workflow está sendo executado e esse atributo será de extrema importância na estratégia de fragmentação, graças ao caráter disperso do cenário em que os experimentos estão inseridos. O atributo *tag* define um rótulo ou uma classificação para o experimento, um exemplo é SciPhy, um *workflow* de análise filogenética. O atributo *tagexec* identifica unicamente uma tentativa de execução do *workflow*. Caso o cientista solicite a execução de uma tentativa já existente, o SciCumulus identificará as atividades ainda não executadas para dar prosseguimento a execução solicitada. O atributo *expDir* aponta para o diretório onde se encontram os dados do experimento. Esse diretório se encontra compartilhado na nuvem. O atributo *description* permite que uma descrição referente ao experimento executado seja inserida.

A próxima tabela a ser analisada é a tabela *hactivity*, que diz respeito a uma atividade de um *workflow* e é representada pela Figura 13.

Figura 13 - Tabela de Atividade

Nessa tabela, o atributo *actid* é um inteiro que representa um identificador. O atributo *wkfid* é uma chave estrangeira que indica a qual *workflow* a atividade pertence, lembrando que um *workflow* é constituído por um conjunto de atividades. *Tag* é um rótulo para a atividade, que indica qual atividade está sendo representada. *Atype* indica o tipo de operação que a atividade realiza. O atributo *description* permite que o cientista insira um texto com uma descrição mais detalhada da atividade. *Templatedir* indica em que diretório está o *template* referente à atividade representada, que pode conter informações em relação à chamada da atividade bem como recursos necessários. *Status* contém o estado em que a atividade se encontra, considerando que uma atividade pode estar enfileirada, executando, esperando ou executada.*Activation* contém o caminho para o arquivo com a chamada da atividade na nuvem. *Startime* e *endtime* indicam os tempos de início e de fim da execução da atividade, respectivamente, sendo o tempo de início indicado pelo momento em que a primeira ativação da atividade foi iniciada e o tempo de fim indicado pelo momento em que a última ativação da atividade foi

finalizada. *Extractor* indica o arquivo usado para executar o programa que extrai os dados do resultado da execução da ativação para transformá-los em tupla de saída.

A seguir, outra tabela de extrema importância nas consultas mais realizadas pelos cientistas: a tabela de ativação, representada pela Figura 14. Aqui devemos destacar que uma determinada atividade do *workflow* pode ser executada milhares de vezes, variando-se os parâmetros de entrada. A tabela abaixo irá guardar exatamente isso. Para cada parâmetro consumido por uma determinada atividade, um novo registro será então armazenado nessa relação.

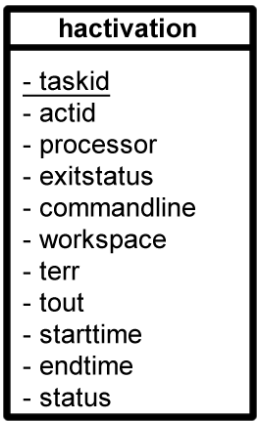

Figura 14 - Tabela de Ativação

O atributo *taskid* representa o identificador da ativação. O atributo *actid* faz referência à atividade a qual a ativação pertence, lembrando que uma atividade é formada por um conjunto de ativações. *Exitstatus* é um número inteiro que representa o status de saída, indicando caso tenha ocorrido algum problema na execução. *Commandline* indica o caminho para o arquivo que contém o comando para a execução da ativação. *Workspace* indica o local em que a ativação produziu e consumiu seus dados. O atributo *terr* contém possíveis mensagens de erro que tenham sido geradas da execução da ativação. *Startime* e *endtime* indicam o tempo de início e o tempo de fim da

execução da ativação, respectivamente. Por fim, o *status* indica em que estado a ativação se encontra, podendo estar enfileirada, em execução, esperando ou já executada.

As tabelas apresentadas nesta seção contemplam grande parte das consultas feitas por cientistas, levando em consideração que a enorme parte das consultas gira em torno dos *workflows* executados ou em execução para que se possa obter informações para o experimento. E como um *workflow* é constituído por um conjunto de atividades a serem executadas que, respectivamente, são constituídas por conjuntos de ativações, é fundamental levar em consideração os atributos de cada uma dessas tabelas. Na próxima seção, características importantes da base de dados também serão apresentadas a fim de justificar a estratégia de fragmentação utiliza.

4.2. Estratégias de Fragmentação

Primeiramente, é importante ressaltar por que razão a fragmentação parece ser uma alternativa interessante o suficiente para justificar a análise realizada neste projeto. Para isso, precisamos entender melhor o cenário em que os experimentos científicos estão inseridos.

Existem diversos centros de pesquisa e universidades que realizam experimentos com a utilização de *workflows* científicos. No entanto, estes diversos centros de pesquisa estão espalhados ao redor do mundo inteiro e dados de proveniência referentes a experimentos executados estão sendo gerados constantemente. Esses dados de

proveniência gerados podem ser de extrema utilidade para um outro centro de pesquisa que pode estar ou não localizado em um outro país.

Um exemplo que ilustra a utilidade desses dados em um contexto realista é o seguinte: Pedro é um cientista localizado em São Paulo, no Brasil, e ele já executa há algum tempo uma série de experimentos utilizando um *workflow* chamado SciPhy, que realiza análise filogenética. Em sua base de dados de proveniência, já existe uma grande quantidade de informações referentes a experimentos realizados no passado, juntamente com informações referentes às atividades executadas e as ativações referentes a cada uma dessas atividades. Um cientista na Irlanda, chamado Robert, está se familiarizando agora com o *workflow* SciPhy e pretende utilizá-lo para a execução de experimentos de análise filogenética. Ele nunca executou o *workflow* antes, então informações referentes a parâmetros utilizados, atividades envolvidas, tempo de execução, dentre outras são desconhecidas por ele e seriam de grande utilidade para que ele possa realizar a configuração de seu ambiente de execução de maneira eficiente a fim de executar o *workflow*. Sendo assim, as informações geradas por Pedro seriam de grande utilidade para Robert. Esse é um cenário comum e o objetivo deste projeto é permitir que Robert possa obter esses dados de proveniência da maneira mais eficiente possível, tendo em vista que Robert tenderá a fazer consultas frequentes nos dados gerados por Pedro. Além disso, mesmo no futuro, onde ambos já estarão experientes com o uso do *workflow*, será interessante para o Robert consultar os dados gerados pelas execuções de experimentos realizadas por Pedro e vice-versa, pois os resultados obtidos podem ser, e provavelmente são, úteis para ambos os cientistas. Sendo assim, pode-se perceber que a redução do tempo necessário para a realização de tais consultas tende a trazer benefícios para ambos os cientistas, no cenário exemplificado, e para qualquer outro cientista que

venha a realizar buscas nesses conjuntos de dados. Tendo isso em vista, existem duas formas de se alocar os dados para que todos possam ter acesso: eles podem se encontrar centralizados em um único nó, e todos os cientistas acessam aquele nó para fazer suas buscas, bem como para inserir novos registros; ou eles podem se encontrar de maneira distribuída, de forma que as consultas serão realizadas apenas nos nós que dizem respeito a ela. Os dois cenários serão analisados a fim de entender qual é a melhor forma de alocar os dados.

Sabe-se que os cientistas estão naturalmente distribuídos ao redor do mundo e que, consequentemente, os dados também tendem a estar distribuídos, conforme as máquinas onde os *workflows* são executados. Sendo assim, no cenário distribuído realizamos uma fragmentação por *workflow*, pois essa é a distribuição mais natural, intuitiva e compatível com a realidade descrita: os fragmentos representam os conjuntos de *workflows* gerados. Nesse caso, voltando ao exemplo apresentado, Robert faria sua consulta apenas no fragmento que contém informações referentes ao *workflow* SciPhy, que é seu *workflow* de interesse. Ao partir para um cenário centralizado estaríamos indo contra a natureza distribuída do cenário descrito. Perderíamos inclusive a alta disponibilidade que o ambiente distribuído proporciona, pois na falha do suposto nó centralizado, não haveria a possibilidade de recuperar a informação pesquisada em um outro nó disponível, como é possível no ambiente distribuído.

Se um mesmo *workflow* pode ser executado tanto na Irlanda quanto no Brasil, como no exemplo apresentado, a próxima decisão a ser tomada seria onde inserir o *workflow*. As consultas feitas por um grupo de pesquisa aos dados gerados por um outro grupo que trabalha na mesma linha de pesquisa tendem a ser frequentes. No entanto, a frequência com que se acessa os dados pelo seu próprio grupo de pesquisas tende a ser

muito mais frequente. Além disso, quando se trata de seu grupo o cientista não só realiza consultas a base como também insere novos dados. Levando isso em consideração, no exemplo dado o ideal seria que Pedro, que está no Brasil, tivesse acesso rápido ao fragmento onde se encontram as execuções geradas por ele, e o mesmo vale para Robert, que está na Irlanda.

Com isso em mente, o projeto de fragmentação aqui proposto é baseado não só nos *workflows* executados, mas também na localização dos mesmos. Ou seja, um experimento SciPhy executado por cientistas do Brasil estará na máquina presente no Brasil, enquanto o experimento executado por cientistas na Irlanda se encontrarão na máquina presente na Irlanda. Desse modo, os dados estarão disponíveis e podem ser consultados por qualquer cientista presente na rede, mas leva-se em consideração o caráter distribuído de suas execuções.

A Figura 15 mostra onde estão localizadas cada uma das máquinas utilizadas no experimento, bem como quais são as máquinas que contém os dados fragmentados e qual contém os dados de forma centralizada.

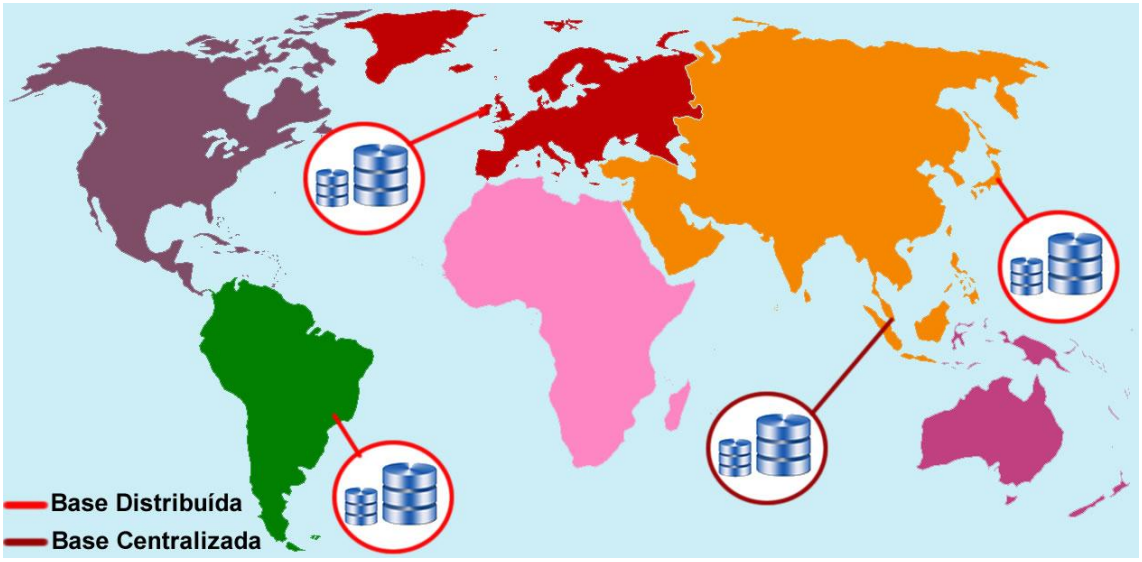

Figura 15 - Mapa com localização das máquinas utilizadas nas análises

No cenário proposto neste projeto, partimos de uma configuração centralizada, atualmente localizada em Singapura. Ela possui todo o conjunto de *workflows* executados. A partir dos dados presentes no nó centralizado, iniciamos o projeto de fragmentação baseado no *workflow* executado e na sua localização. O resultados foram colocados em três máquinas localizadas em São Paulo, na Irlanda e em Tóquio. Os dados encontrados no nó centralizado são os mesmos dos encontrados no conjunto de nós distribuídos, ou seja: se uma determinada informação pode ser encontrada no nó centralizado então aquela informação pode também ser encontrada no conjunto de nós distribuídos e vice-versa. Além disso, se uma informação não pode ser encontrada no nó centralizado, ela também não poderá ser encontrada no conjunto de nós distribuídos.

As mesmas consultas serão realizadas em ambos os cenários e os tempos serão medidos a fim de obter os desempenhos e descobrir se a estratégia proposta é viável em termos de desempenho. Vale ressaltar que os algoritmos utilizados para descobrir em qual fragmento uma informação se encontra estão fora do escopo dessa análise. É importante também considerar que os tempos de acesso a cada uma das máquinas

podem oscilar, mas nos testes realizados não foram observados grandes oscilações ou diferenças nos tempos de acesso que possam impactar os resultados de maneira significativa e por isso, não foram levados em consideração. O nó centralizado se localiza em Singapura e, por ser distante, pode tornar o tempo de acesso um pouco mais lento. No entanto isso reflete o cenário real, onde os centros de pesquisa podem estar em diversos locais espalhados pelo mundo. Além disso, existem nós distribuídos que também estão muito distantes, como Tóquio e Irlanda, o que torna a análise bastante realista.

Como apresentado no capitulo referente aos conceitos preliminares, existem duas estratégias de fragmentação fundamentais: a fragmentação vertical e a fragmentação horizontal. Para escolher entre uma e outra é importante avaliar as particularidades do cenário. A fragmentação vertical se mostra vantajosa em casos onde um determinado atributo é acessado com muita freqüência. Por exemplo, caso os atributos referentes aos tempos de início e fim de um atividade fossem acessados com muita mais freqüência que os outros atributos, uma estratégia interessante seria a de realizar uma fragmentação vertical a partir desses atributos.

No entanto, não se observa no cenário descrito um número maior de acessos a um atributo específico. O que se observa é um interesse maior por determinados *workflows*, que gera consultas pelos mais variados atributos. Nesse caso, a escolha mais apropriada é a fragmentação horizontal. Que mantém as tuplas com todos os seus atributos, mas diminui a quantidade de tuplas a serem visitadas em uma busca, diminuindo o tempo de resposta. Considerando que os fatores mais relevantes para os cientistas, em geral, são a *tag* do *workflow* e a sua localidade, tendo em vista que ele tende a fazer um número maior de consultas em experimentos executados por seu centro de pesquisas, a escolha

por esses atributos para a fragmentação horizontal também acaba se tornando uma decisão natural.

Em relação ao projeto de fragmentação, os atributos utilizados foram *tag* e *localidade* da tabela *hworkflow*. Isso acontece por que assume-se, baseando-se na realidade, que a maioria do acessos será feita a um determinado *workflow*, normalmente aquele em que o cientista está trabalhando. Além disso, o cientista tende a acessar os dados da localidade em que ele está trabalhando na maioria das vezes. Embora no exemplo apresentado um cientista da Irlanda tenha interesse em dados gerados no Brasil, a partir do momento que o cientista da Irlanda começar suas próprias execuções é natural que o maior número de seus acessos passe a ser aos dados gerados pelos seus *workflows*, embora ainda seja comum o acesso a outros nós para obter informações que posam ser úteis sobre outras execuções.

Aplicou-se então uma fragmentação horizontal na tabela *hworkflow*. Levando em conta que o projeto de fragmentação precisa ser correto, ou seja, respeitar os conceitos de completude, reconstrução e disjunção utilizou-se a técnica de fragmentação horizontal apresentada por (Özsu e Valduriez, 2010) com a geração de mintermos a partir dos predicados. Os predicados utilizados, baseando-se nos dados presentes no cenário analisado e as freqüências com que são acessados, são os seguintes: P_1. tag="SciPhy"; P_2. tag="SciPhylomics"; P_3. tag="SciEvol"; P_4. tag="ExpX"; P_5. tag="exp"; P_6. tag="Scimultaneous"; P_7. localidade="Tóquio"; P_8. localidade="Irlanda"; P_9. localidade="SP".

Se gerarmos todos os mintermos possíveis, teríamos 2^9 mintermos. No entanto, as relações de implicação entre os predicados diminui consideravelmente esse número. As

relações são basicamente: se um *workflow* possui uma *tag*, esse mesmo *workflow* não poderá possuir nenhuma outra *tag*; se um *workflow* está sendo executado em uma localidade, ele não poderá estar sendo executado em nenhuma outra localidade; é possível ter um *workflow* com *tag* diferente de todas as *tags* apresentadas nos predicados de P_1 a P_6; não é possível ter um *workflow* com uma localidade diferente das apresentadas nos predicados de P_7 a P_9.

A Figura 16 mostra uma representação dos fragmentos gerados a partir desta técnica.

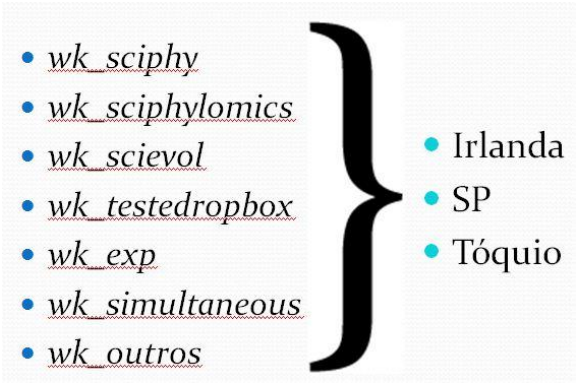

Figura 16 - Representação dos fragmentos gerados

Como se pode observar na figura, foram criados fragmentos levando-se em consideração cada uma das seis *tags* apresentadas nos predicados de P_1 a P_6 e cada uma das localidades apresentadas nos predicados de P_7 a P_9. Sendo assim foram criados fragmentos para os *workflows* com as *tags* SciPhy, SciPhylomics, SciEvol, ExpX, Exp, Scimultaneous para cada uma das localidades Irlanda, SP e Tóquio. Além disso existe também um fragmento *wk_outros*, para cada uma das localidades, onde devem ser armazenados *workflows* com *tags* diferentes das apresentadas anteriormente. Um total de vinte um fragmentos foram gerados: sete fragmentos *wk_sciphy*, *wk_sciphylomics*,

wk_scievol, wk_expx, wk_exp, wk_simultaneous e *wk_outros* para cada uma das três localidades Irlanda, SP e Tóquio. É importante ressaltar que o fragmento *wk_sciphy* de Tóquio, possui registros diferentes do *wk_sciphy* de qualquer outra localidade, e isso serve para todas as outras *tags*, respeitando o princípio de disjunção.

4.3. Análise de Desempenho

Para realizar a análise de desempenho, foram consideradas consultas que são frequentemente utilizadas pelos cientistas no dia a dia. Existem diversas variações que representam a realidade, mas para os experimentos foram selecionadas nove consultas consideradas muito importantes no cotidiano dos cientistas. A seguir, cada uma delas será apresentada.

1. Listar os rótulos e as descrições de todas as atividades de um determinado *workflow*. Essa é uma consulta extremamente comum, pois se sabe que um *workflow* é constituído por uma série de atividades. Saber quais são essas atividades e o que cada uma delas faz, informação obtida através do atributo descrição, é de fundamental importância para entender o funcionamento do *workflow* como um todo.
2. Listar todas as ativações para cada uma das atividades de um determinado *workflow*. Essa é uma outra consulta muito importante pois, como foi dito anteriormente, para entender o funcionamento do *workflow* é fundamental entender como funcionam as atividades que o constituem. Por outro lado, para se entender de fato o comportamento de uma atividade, é importante analisar as informações referente às suas ativações.

3. Listar todas as atividades com estado diferente de “finished” de um determinado *workflow* em São Paulo, ou seja, todas as atividades que ainda não foram executadas ou não finalizaram sua execução. Essa consulta é importante pois a execução do *workflow* depende diretamente da execução de cada uma de suas atividades, o que torna a realização dessa consulta importante enquanto os *workflows* estão sendo executados, para entender em que etapa da execução ele está e inclusive tomar ações para contornar problemas que surjam durante a execução.
4. Buscar a duração de todas as atividades com estado igual a “*finished*” de um determinado *workflow* na Irlanda. A partir dos atributos referente aos tempos de início e de fim da atividade, é possível obter a duração da mesma. O tempo de execução de um *workflow* depende diretamente do tempo de execução de cada uma de suas atividades. Sendo assim, é útil para o cientista avaliar os tempos de execução das atividades de um determinado *workflow* a fim de avaliar quais atividades podem ser responsáveis por aumentar o tempo de execução do experimento e ainda estimar o tempo de execução de um experimento a ser executado, baseado nas informações históricas obtidas.
5. Listar o horário de início de cada uma das atividades com status igual a “*running*” de um determinado *workflow* em Tóquio. Nessa consulta, parte-se do princípio que um experimento está sendo executado e existem atividades que ainda estão em execução. Para essas atividades, saber o momento em que ela se iniciou pode ajudar o cientista a estimar quanto tempo terá de esperar para o fim da execução. Para realizar essa estimativa, ele se baseará também nos dados obtidos através da consulta anterior.

6. Buscar a duração de cada uma das ativações que constituem uma determinada atividade. Essa consulta é importante por razões semelhantes às que justificam a importância das consultas anteriores. A partir das informações de duração obtidas através dessa busca, é possível realizar estimativas para o futuro ou ainda avaliar os tempos de execução e comparar com as configurações, fazendo um paralelo que relaciona os parâmetros configurados e o tempo que se leva para executar uma ativação.
7. Ler mensagem de erro, obtida através do atributo *terr*, de todas as ativações com *status* de saída diferente de zero de um determinado *workflow*. O *status* de saída indica se ocorreu algum problema na execução da ativação. Caso o *status* seja igual a zero, não ocorreram problemas na execução da ativação, caso contrário algum problema ocorreu e para o cientista é fundamental entender qual o problema ocorrido. Para isso, analisa-se a mensagem de erro gerada por todas as ativações que apresentaram problema na execução.
8. Recuperar todos os *workflows* que contêm uma determinada atividade especificada. Nesse tipo de consulta, o cientista tem a oportunidade de conhecer *workflows* já existentes e executados que possam ser reaproveitados ou reutilizados. Através dessa pesquisa, ele filtra todos os *workflows* que contêm uma atividade que lhe interessa de alguma forma. E por essa pesquisa ele pode observar o comportamento da atividade quando inserida em outros contextos, que podem ser úteis a ele.
9. Recuperar, por ordem crescente de execuções de *workflows*, as datas e horas de início e término das ativações, *tags* dos *workflows* e suas descrições, bem como o nome de todas as atividades associadas a todas as execuções dos *workflows* que foram executados e não contenham nenhuma ativação que

executou com erro. Essa consulta é fundamental pois permite que os cientistas monitorem em tempo real as execuções individualizadas de cada ativação de cada *workflow*. Aproveitando-se do fato de que o SciCumulus gera os dados de proveniência em tempo real, é possível utilizar essa consulta para uma análise prévia do experimento, avaliando se ele está de acordo com o que se espera ou não.

Cada uma das nove consultas apresentadas foram executadas no ambiente distribuído e no ambiente centralizado, ambos localizados na nuvem da Amazon. Os tempos de resposta ,obtidos através da execução das consultas, foram comparados a fim de analisar o desempenho conseguido com a estratégia de fragmentação e se essa estratégia possui vantagens em relação ao ambiente centralizado em termos de tempo de resposta, tendo em vista que foram apresentadas diversas vantagens acerca do ambiente onde se realizam os experimentos, naturalmente distribuído pelo mundo.

Os resultados obtidos para os tempos de resposta são apresentados na Figura 16.

Consulta	Tuplas acessadas	Tempo em ms (centralizado)	Tempo em ms (distribuído)	%
1	105	826	405	49% ↓
2	5828	8408	1852	22% ↓
3	15	391	208	53% ↓
4	23	374	209	56% ↓
5	11	406	199	49% ↓
6	1614	1278	1352	0,6% ↑
7	125	1743	773	44% ↓
8	42	231	722	213% ↑
9	15259	16973	26052	53% ↑

Figura 17- Resultados dos experimentos

Para a realização desse estudo de desempenho foi efetuada uma série de execuções das consultas descritas anteriormente, obtendo assim uma média dos tempos de execução. Nas sucessivas execuções, realizadas a fim de obter a média, não foram observadas oscilações significativas no tempo obtido, por isso o desvio padrão é desconsiderado aqui, por ser um valor de pouca significância em relação ao tempo total. É importante salientar também que a elaboração das consultas levou em consideração que a maioria dos cientistas realiza pesquisas buscando um *workflow* em particular.

Podemos observar conforme o esperado que ocorreu uma melhora nos tempos de execução para a maioria das consultas. É possível notar esse resultado, pois essas consultas utilizam como restrição a pesquisa por um determinado *workflow*, beneficiando assim os cientistas que buscam o seu experimento de trabalho. Essa evolução ocorre devido à criação dos fragmentos da tabela que contém os *workflows*, acabando com a necessidade de varrer cada registro da tabela em busca de um determinado *workflow,* conforme ocorre na busca em nós centralizados, e passam a utilizar os fragmentos para isso.

Podemos citar também um caso particular de consulta que não envolve a pesquisa por algum *workflow*, é o caso da consulta seis, que busca a duração das ativações de uma atividade. O tempo de resposta para essa pesquisa apresentou resultados equivalentes na base centralizada e distribuída, isso ocorre devido ao baixo valor do tempo de acesso à tabela atividade em cada nó distribuído, ou seja , existiriam no máximo três acessos à tabela atividade na base de dados distribuída enquanto que na centralizada apenas um.

Contudo a necessidade por consultas envolvendo diferentes *workflows* existe e para esses casos a fragmentação pode gerar resultados piores que a busca em nó centralizado.

Os casos em que o tempo de execução apresentou um resultado pior na base distribuída ocorreram devido à necessidade de fazer a junção dos fragmentos, remontando a tabela original para a geração dos resultados. Esse procedimento fez com que a consulta oito, que tem o objetivo de recuperar os workflows que possuem uma atividade específica, tivesse um aumento de 213% no seu tempo de execução.

Algumas consultas podem ter sido influenciadas pelo tempo de acesso, levando em consideração a distância dos nós. Um exemplo é a consulta três, que faz uma busca em São Paulo. No caso distribuído o acesso ao nó de São Paulo tende a ser mais rápido por estar mais próximo, enquanto o acesso ao nó centralizado, localizado em Singapura, tende a ser um pouco mais lento. Por outro lado, a consulta cinco é realizada em Tóquio, ou seja, o nó distribuído se encontra mais longe do que o nó centralizado, ainda assim observou-se ganhos no tempo de consulta. É importante lembrar que esses tipo de influência da distância reflete a realidade, tendo em vista que os centros de pesquisa estão espalhados pelo mundo.

Para este experimento utilizamos uma simplificação do modelo de custo, desconsiderando fatores monetários e a confiabilidade das máquinas utilizadas, pois esses fatores não puderam ser medidos. Utilizamos os tempos necessários para a realização das consultas para avaliar os resultados obtidos.

Capítulo 5

Conclusão

Os *workflows* científicos apresentam-se como uma abordagem muito interessante para a modelagem de experimentos científicos baseados em simulações. No entanto, é necessário realizar uma série de configurações tanto na estrutura do *workflow*, como no ambiente em que ele será executado para que o experimento seja executado de maneira correta.

Para realizar tais configurações, informações históricas sobre *workflows* já executados anteriormente podem ser de grande utilidade, sejam eles executados ou não pela mesma equipe. E é nesse ponto que os dados de proveniência se mostram tão úteis para o gerenciamento da execução de *workflows*: dados de execuções anteriores de um *workflow* contém informações acerca das atividades executadas, as diferentes ativações utilizadas, os tempos de execução, os resultados gerados, possíveis erros ocorridos dentre outros.

Obter essas informações traz grandes benefícios para os cientistas pois a partir delas é possível reproduzir a execução de um *workflow* a partir de uma execução anterior, além de realizar um melhor gerenciamento dos recursos necessários e análises de desempenho, bem como detectar erros com maior facilidade e em tempo real.

Como os dados de proveniência são utilizados frequentemente para tomar decisões em tempo real, é importante ter consultas com um tempo de resposta pequeno, para que essas não afetem o desempenho do *workflow* como um todo.

Tendo isso em mente e levando em consideração que os dados de proveniência tendem a ser gerados de forma distribuída, por diversos centros de pesquisa e universidades, foi proposto um projeto de fragmentação a fim de distribuir os dados de proveniência de maneira a minimizar o tempo necessário para que as consultas sejam realizadas, diminuindo o possível impacto negativo no tempo de execução de um *workflow* que depende dessas informações.

Essa proposta levou em consideração dois fatores principais: o fato de que as consultas realizadas por um cientista em geral giram em torno do *workflow* em que ele está trabalhando; e que, no geral, os dados gerados por aquele cientista são armazenados nas máquinas utilizadas por eles.

Tendo isso em mente, foi realizada uma proposta de fragmentação sobre os atributos *tag* e *localidade*, que representam respectivamente o rótulo para o *workflow* e o local onde os dados de proveniência estão armazenados. Uma série de consultas, frequentemente utilizadas por cientistas, foi selecionada e testada tanto no ambiente distribuído, representado pelos nós presentes em São Paulo, Tóquio e Irlanda, como no ambiente centralizado, representado pelo nó de Singapura.

Para a avaliação, adotamos a estratégia de obter as médias dos tempos de execução das consultas , considerando o tempo de comunicação coma máquina virtual e o tempo de pesquisa por uma informação especifica. Os resultados obtidos indicam uma melhora significativa no ambiente distribuído em consultas realizadas em um único fragmento. Essas consultas são realizadas com uma frequência grande, levando em consideração que os cientistas buscam em geral informações sobre um *workflow* específico. No entanto, consultas que varrem mais de um fragmento se mostraram em desvantagem nos ambientes distribuídos.

Analisando-se os resultados obtidos pode-se concluir que o ambiente distribuído mostrou-se vantajoso, já que no geral as consultas são realizadas para um *workflow* específico. Por outro lado, existe uma desvantagem significativa para consultas que abrangem diversos *workflows*. Embora para esse caso específico a desvantagem seja considerável, o fato de esse tipo de consulta não ser realizado com maior frequência e o caráter naturalmente distribuído do ambiente de execução de experimentos nos leva a crer que a fragmentação dos dados da maneira apresentada é uma boa forma de se melhorar as consultas mais frequentes em um ambiente distribuído.

Além disso, propomos como trabalhos futuros, uma análise mais detalhada das consultas, levando em consideração os tempos de acesso a cada um dos nós para a busca de informações, bem como os algoritmos e tempos de análise da consulta a fim de direcioná-la para o fragmento correto. Além disso, realizar um estudo de custos mais aprofundado, levando-se em consideração o modelo apresentado no Capítulo 3, que considera também o custo monetário para a utilização das máquinas para a execução do *workflow,* além da confiabilidade e disponibilidade das mesmas.

Referências Bibliográficas

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

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

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

Costa,F., (2012), *Heurísticas para Controle de Re-execução Paralela de Workflows Científicos em Nuvens de Computadores*. Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2012., UFRJ/COPPE

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

Davidson, S., Freire, J., (2008), "Provenance and scientific workflows: challenges and opportunities", *Computing in Science and Engineering*, v.10, n.3, pp 11-21.

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

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

Gadelha, L., (2012), “Gerência de proveniência em workflows científicos paralelos e distribuídos”. Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2012., UFRJ/COPPE

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

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

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

Moreau, L., Freire, J., Futrelle, J., McGrath, R., Myers, J., Paulson, P., (2008a), "The Open Provenance Model: An Overview", *Provenance and Annotation of Data and Processes*, p. 323-326.

Moreau, L., Ludäscher, B., Altintas, I., Barga, R. S., Bowers, S., Callahan, S., George Chin, J., Clifford, B., Cohen, S., et al., (2008b), "Special Issue: The First Provenance Challenge", *Concurrency and Computation: Practice and Experience*, v. 20, n. 5, p. 409-418.

Moreau, L., Missier, P., (2011). The PROV Data Model and Abstract Syntax Notation.*W3C Working Draft.(Work in progress.)*.

Moreau, L., Missier, P., Belhajjame, K., Cresswell, S., Golden, R., Groth, P., Miles, S., Sahoo, S., (2011). The PROV Data Model and Abstract Syntax Notation.

Napper, J., Bientinesi, P., (2009), "Can *cloud* computing reach the top500?". In: *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, p. 17-20, Ischia, Italy.

Ogasawara, E., (2011), *Uma Abordagem Algébrica para Workflows Científicos com Dados em Larga Escala*. Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2012., UFRJ/COPPE

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

Oliveira, D., Ocaña, K., Baião, F., Mattoso, (2012), "A Provenance-based Adaptive Scheduling Heuristic for Parallel Scientific Workflows in Clouds", *Journal of Grid Computing*, v. 10, n.3, pp.521-552

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

Oliveira, D., Ogasawara, E., Baião, F., Mattoso, M., (2010b), "SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows". In: *3rd International Conference on Cloud Computing*, p. 378–385, Washington, DC, USA.

Özsu, M., Valduriez, P., (2010), “Principles of Distributed Database System.”*ThirdEdition.*

Ruberg, G., (2001), “Um Modelo de Custo para o Processamento de Consultas em Bases de Objetos Distribuídos”

Simmhan, Y., Ingen, C., Subramanian, G., Li, J., (2010), "Bridging the Gap between Desktop and the *Cloud* for eScience Applications". In: *3rd IEEE Conference of Cloud Computing3rd IEEE Conference of Cloud Computing*, p. 474-481, Miami.

Simmhan, Y., Plale, B., Gannon, D., (2005), “A survey of data provenance in e-science”

Sousa, V., (2011), “Simiflow: Uma Arquitetura para Agrupamentos de Workflow por Similaridade.”

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

Viana, V., Oliveira, D., Mattoso, M., (2011), “Towards a Cost Model for Scheduling Scientific Workflows Activities in Cloud Environments”, in *2011 IEEE World Congress on Services (SERVICES)*, pp. 216–219.

Wallis, J. C., Mayernik, M. S., Borgman, C. L., Pepe, A., (2010), "Digital libraries for scientific data discovery and reuse: from vision to practical reality". In: *Proceedings of the 10th annual joint conference on Digital libraries*, p. 333–340, New York, NY, USA.

Wang, J., Crawl, D., Altintas, I., (2009), "Kepler + Hadoop: a general architecture facilitating data-intensive applications in scientific *workflow* systems". In: *4th Workshop on Workflows in Support of Large-Scale Science*, p. 1-8, Portland, Oregon.

Wang, L., Tao, J., Kunze, M., Castellanos, A. C., Kramer, D., Karl, W., (2008), "Scientific *Cloud* Computing: Early Definition and Experience". In: *10th IEEE HPCC*, p. 825-830, Los Alamitos, CA, USA

Youseff, L., Butrico, M., Da Silva, D., (2008), "Toward a Unified Ontology of *Cloud* Computing". In: *Grid Computing Environments Workshop, 2008. GCE '08*, p. 10, 1