UMA ABORDAGEM DISTRIBUÍDA E ADAPTATIVA PARA A GERÊNCIA DOS DADOS DE PROVENIÊNCIA DE EXPERIMENTOS CIENTÍFICOS EXECUTADOS EM NUVENS COMPUTACIONAIS

Flavio da Silva Costa

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

Orientadores: Marta Lima de Queirós Mattoso
Daniel Cardoso Moraes de Oliveira

Rio de Janeiro
Setembro de 2016

UMA PROPOSTA DE ARQUITETURA DISTRIBUÍDA E ADAPTATIVA PARA A GERÊNCIA DOS DADOS DE PROVENIÊNCIA DE EXPERIMENTOS CIENTÍFICOS EXECUTADOS EM NUVENS COMPUTACIONAIS

Flavio da Silva Costa

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

Examinada por:

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

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

Profª. Fernanda Araujo Baião Amorim, D.Sc.

Prof. Alexandre Plastino de Carvalho, D.Sc.

Prof. Alexandre de Assis Bento Lima, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

SETEMBRO DE 2016

Costa, Flavio da Silva

Uma Abordagem Distribuída e Adaptativa para a Gerência dos Dados de Proveniência de Experimentos Científicos Executados em Nuvens Computacionais / Flavio da Silva Costa. – Rio de Janeiro: UFRJ/COPPE, 2016.

XVIII, 109 p.: il.; 29,7 cm.

Orientadores: Marta Lima de Queirós Mattoso

Daniel Cardoso Moraes de Oliveira

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2016.

Referências Bibliográficas: p. 102-109.

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

À minha esposa Camille e aos meus pais.

Agradecimentos

Neste trabalho épico encerro um ciclo de aprendizado, o mais importante da minha vida e sem dúvida o mais laborioso. Na mesma proporção do esforço empreendido, vem a satisfação do dever cumprido e da bagagem de conhecimento e amizade, adquiridos ou reforçados. Falo com firmeza que jamais chegaria aqui sem a ajuda de grandes amigos, mestres excepcionais e da minha família. Quando chegamos perto do fim de um ciclo, paramos para refletir o que tornou possível tal realização e uma das coisas que invariavelmente concluímos é que não somos ninguém sozinhos. Nesse momento, o mínimo que podemos fazer é agradecer a todos aqueles que de alguma forma contribuíram para a concretização de mais um sonho. No entanto, pesa-nos a certeza que seremos injustos, pois de certo, muitos nomes faltarão. Sendo assim, deixo aqui meu obrigado aos que por ventura sejam omitidos em meus agradecimentos. No entanto, o erro do esquecimento seria ainda maior se não agradecesse, em especial, a algumas pessoas sem as quais nada disso seria possível. Sendo assim agradeço:

À minha esposa Camille, minha melhor amiga de todas as horas. Aquela que representa os três grupos que citei incialmente: Família, Amigos e Mestres. Nos momentos difíceis dessa caminhada ela esteve sempre pronta a me ajudar, com sua dedicação inabalável, com seus conhecimentos e seu amor. Esteve sempre pronta pra me dar força nos momentos de cansaço e até escrever algumas linhas de código para me ajudar. Nos últimos anos você foi a pessoa mais importante na minha vida e será assim pra sempre. Aos meus pais Marco Antonio e Gilvanete que abriram mão de muito para tudo me oferecer. Ter nascido seu filho já fez de mim um vencedor, todo o resto foi lucro. Vocês são um exemplo na minha vida e vivem sempre comigo em meu coração. Nunca conseguirei retribuir à altura. Nunca serei capaz de fazer por vocês o que fizeram e fazem até hoje por mim. Aproveito aqui para agradecer também aos meus novos pais de coração, Luiz Antonio e Terezinha, pois ao casar, fui agraciado com o amor desta família que me recebeu como um filho e esteve comigo nos últimos anos torcendo e me apoiando no que fosse preciso. Hoje celebramos todos juntos esta vitória!

À minha família pela torcida e companheirismo: Minha irmã Erica, meus sobrinhos Clarisse e Marco Antonio, minhas afilhadas Fernanda, Maryana e Eduarda. E aos irmãos de coração: Felipe Assis, Giselle Furtado, Jeane Quintiliano, Roberto Aldilei e Fernando Pereira. Agradeço também aos meu padrinho Euller, e em especial a minha

madrinha Inacira por estar sempre comigo em todas as minhas conquistas e me tratar sempre com muito carinho.

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

À professora Marta Lima de Queirós Mattoso, pela oportunidade de fazer parte de sua equipe e pela orientação precisa, eficiente e presente, contribuindo em muito para a excelência desse trabalho. Seguirei sendo um grande admirador seu. Ao professor Geraldo Bonorino Xexéo, a quem serei eternamente grato pela oportunidade de aqui estar. Hoje os agradeço defendendo minha tese.

Ao meu primo Helio Jaques que despertou em mim a curiosidade dos pesquisadores, e esteve sempre disposto a me ajudar com as questões da ciência e da vida. Hoje essa vitória também é sua

À Kary Ocaña por toda ajuda na questão dos experimentos dessa dissertação e por sempre se mostrar disposta e paciente para esclarecer as questões da bioinformática. Sempre esteve pronta a me ajudar e me animar em todos os momentos desta caminhada.

Em muito especial ao professor Daniel Oliveira que não mediu esforços para me ajudar em todas as questões envolvidas neste projeto. Um grande amigo que a vida me deu!

Aos professores Fernanda Baião, Alexandre Plastino e Alexandre Assis por terem aceitado fazer parte desta banca.

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

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

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

Por fim agradeço a Deus pela oportunidade de dar mais este passo.

Agradeço.

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

UMA ABORDAGEM DISTRIBUÍDA E ADAPTATIVA PARA A GERÊNCIA DOS DADOS DE PROVENIÊNCIA DE EXPERIMENTOS CIENTÍFICOS EXECUTADOS EM NUVENS COMPUTACIONAIS

Flavio da Silva Costa

Setembro/2016

Orientadores: Marta Lima de Queirós Mattoso

Daniel Cardoso Moraes de Oliveira

Programa: Engenharia de Sistemas e Computação

A ciência baseada em simulação computacional é fortemente dependente dos avanços tecnológicos. Muitos *workflows* científicos são formados por milhares de atividades (invocações de programas) e, em apenas uma execução, estas atividades podem ser executadas milhares de vezes, para que o resultado final seja gerado. Neste contexto, a computação de alto desempenho é uma alternativa importante e, em especial, as nuvens de computadores. As nuvens podem oferecer um alto poder de processamento com baixa complexidade na sua utilização a um custo flexível, o que as tornam uma alternativa viável para a ciência baseada em simulação. Entretanto, mesmo executando em ambientes de nuvem com diversos nós computacionais, o volume de dados (dados de execução dos experimentos e dados de proveniência – que representam o histórico da execução do *workflow)* produzido e consumido e que deve ser gerenciado, pode se tornar um problema. Essa gerência se for realizada de forma centralizada, pode impactar no desempenho da própria execução do workflow, uma vez que os sistemas atuais são responsáveis tanto por executar o *workflow* quanto gerenciar os dados de proveniência. Aproveitando-nos da característica elástica dos recursos da nuvem, apresentamos nessa tese estratégias que buscam melhorar a gerência dos dados de proveniência ao mesmo tempo em que melhoram a eficiência dos experimentos executados, tendo como principal norte duas vertentes: redução do tempo de consultas e de despesas.

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

A PROPOSAL OF AN ADAPTIVE AND DISTRIBUTED ARCHITECTURE FOR MANAGING PROVENANCE DATA OF SCIENTIFIC EXPERIMENTS EXECUTED ON CLOUDS

Flavio da Silva Costa

September/2016

Advisors: Marta Lima de Queirós Mattoso

Daniel Cardoso Moraes de Oliveira

Department: Systems and Computer Engineering

Science based on computational simulation is highly based on advances in computer *software* and *hardware* solutions. Many *workflows* are composed by thousands of activities (program calls) and in just one execution, these activities can be executed thousands of times to generate the expected result. In this context, high-performance computing is an important alternative, in particular cloud computing. Clouds provide high processing power at a flexible cost, which can match the scientist budget and present low complexity involved in its use. However, even when executing these experiments in the elastic cloud environment, the amount of data (domain specific and provenance – that represent the historic of workflow past executions) produced and consumed that needs be handled, can be a problem. If the centralized management strategy is adopted, the execution of the workflow can be impacted, as the actual systems are responsible for the workflow execution and for the provenance management too. In this scenario, we have many opportunities to bring more efficiency to the experiments in terms of reduction of time and money expended. In our thesis, we address these problems and based on strategies that use the elasticity cloud properties, we reach some good results.

Índice Geral

Índice de Figuras

Índice de Tabelas

Capítulo 1 - Introdução

Os *workflows* científicos podem ser definidos como uma abstração capaz de modelar o fluxo de atividades e de dados em um experimento (Deelman et al. 2009). Ao longo dos últimos anos os *workflows* se tornaram um padrão para a modelagem de experimentos científicos que são baseados em simulações computacionais (Mattoso *et al.* 2010). Com o passar do tempo, os *workflows* científicos aumentaram muito sua necessidade por capacidade de processamento por conta da complexidade dos algoritmos envolvidos e do grande volume de dados a processar. Devido a essas características, para que os experimentos pudessem ser executados com maior velocidade, os ambientes de processamento de alto desempenho (PAD), tais como *clusters*, grades computacionais (Foster e Kesselman 2004) e as nuvens computacionais (Vaquero *et al.* 2009, Oliveira *et al.* 2010a) passaram a ser utilizados. Essas infraestruturas complexas, equipadas com máquinas com vários núcleos computacionais, são utilizadas pelos Sistemas de Gerência de *Workflows* Científicos (SGWfC). Esses sistemas gerenciam as rodadas dos experimentos de forma a distribuir a execução de diversas atividades do *workflow* nos diversos recursos computacionais disponíveis. Os SGWfCs têm como objetivo ajudar os cientistas na difícil tarefa de gerenciar o problema da enxurrada de dados (Gorton *et al.* 2008). Usando essas tecnologias, os cientistas têm expandido a execução de seus *workflows* para além dos computadores paralelos individuais e pequenos *clusters* de seus laboratórios para centenas ou milhares de núcleos computacionais.

Além de executar um *workflow* em um ambiente de PAD como os mencionados anteriormente, muitas vezes, os cientistas têm que capturar e gerenciar os dados de proveniência de seus experimentos (Freire *et al.* 2008). A captura de dados de proveniência em experimentos científicos em larga escala é uma questão fundamental, uma vez que permite a reprodutibilidade e a análise dos resultados e sua validação (Davidson e Freire 2008), fundamentais para a ciência. Além disso, a proveniência já é bastante utilizada na comunidade de *e-Science* para alguns propósitos adicionais, dos quais podemos destacar: o tratamento de falhas em atividades do *workflow* (Costa *et al.* 2012), a gerência do experimento (Valerio *et al.* 2008) e o escalonamento adaptativo de atividades do *workflow* em ambientes de nuvem (Oliveira *et al.* 2012a). No entanto, uma vez que os *workflows* científicos continuam a crescer em uma escala sem precedentes, as estratégias de gerência dos dados de proveniência devem ser revistas de

modo a aperfeiçoar seu uso e minimizar os custos envolvidos quando trabalhamos em ambientes pagos (Ailamaki 2011).

Em muitas das abordagens existentes, a execução de *workflow*s científicos ocorre de forma distribuída e a proveniência produzida por esses experimentos, muitas vezes, é copiada de um recurso remoto (máquina virtual por exemplo) para um repositório centralizado (um banco de dados relacional, por exemplo). Como mencionamos anteriormente, a informação histórica sobre a execução de experimentos é muito importante para alguns SGWfCs. A proveniência produzida durante a execução de um experimento poderá servir de parâmetro na configuração dos recursos a serem utilizados nas rodadas de novos experimentos, mesmo que estas novas execuções ocorram em um ambiente ligeiramente diferente daquele onde os dados históricos foram produzidos. Independentemente da sua origem, o que realmente importa neste contexto é encontrar a proveniência obtida a partir do mesmo tipo de *workflow* que estamos prestes a executar ou que está em execução, de modo a ter uma estimativa de seu comportamento (tempo de execução, custo financeiro, etc.). Ter também, na base de proveniência, os parâmetros e arquivos utilizados em um experimento, facilita muito a reprodutibilidade e validação do mesmo, mas devemos considerar que será mais um fator para aumentar o tamanho da base de dados.

Uma das estratégias utilizadas é trabalhar com um repositório de proveniência centralizado, porém, em muitos cenários, esta não é uma alternativa interessante, pois tende a ser pouco escalável. Podemos pegar como base um ambiente onde equipes geograficamente separadas compartilham dados de seus experimentos. Para cada experimento executado, um grande volume de dados que poderá ser gerado deverá ser armazenado. Na estratégia centralizada, toda essa informação é migrada para um repositório único, o que vai criar um ponto único de falha e, possivelmente, um gargalo para os experimentos. No entanto, se os dados de proveniência relacionados a uma série de experimentos forem armazenados de forma distribuída, independente do local onde são produzidos, este tipo de problema tende a ser mitigado. O que temos visto na prática são repositórios distribuídos em múltiplos nós, sendo cada um, geralmente, centralizado e vinculado a um grupo de pesquisa (máquinas físicas ou virtualizadas em diferentes institutos de pesquisa ou universidades). Assim, cada instituto de pesquisa está executando seus experimentos e armazenando a proveniência em um repositório centralizado, que pode residir na Internet ou em redes privadas. Os dados centralizados

de um centro de pesquisa podem ser importantes para outro grupo de pesquisadores e vice-versa. Buscamos assim novas estratégias que ajudem a fazer com que a proveniência possa ser gerenciada de forma distribuída e dessa forma melhorar a escalabilidade, evitar o problema de haver um ponto único de falha, melhorar o desempenho do experimento com redução de tempo e custo, dentre outras questões. Para atingir tal objetivo podemos usar recursos disponíveis da nuvem. No entanto, quando estamos trabalhando com este tipo de ambiente, as soluções clássicas para distribuição e alocação de dados podem ser de difícil aplicação ou mesmo inaplicáveis. Uma das principais vantagens deste ambiente é a escalabilidade, a qual, se explorada adequadamente, tirando o máximo de proveito e utilizando seus recursos dinamicamente de modo a só se pagar por aquilo que efetivamente é necessário, é possível conseguir resultados mais ágeis com menor custo. Trata-se de um cenário ainda em construção que carece de soluções adequadas às caraterísticas apresentadas. Esse trabalho está inserido neste contexto, onde há escassez de recursos financeiros, mas onde também não se pode abrir mão da busca por melhorias de desempenho. Sabemos que esta é a realidade enfrentada por uma série de pesquisadores.

Semelhante às bases de dados distribuídas (Özsu e Valduriez 2011), os repositórios de proveniência podem ser fragmentados, replicados ou mesmo sofrerem pré-processamento de seus dados, agregando informações que poderão ser usadas de forma mais eficiente no futuro. Quando repositórios de proveniência são replicados, várias réplicas idênticas do mesmo repositório são alocadas em diferentes locais físicos. Pode ser inviável, em muitos casos, devido ao volume de dados a ser replicado. Por outro lado, quando repositórios de proveniência são fragmentados, um repositório de proveniência específico é "particionado" em vários repositórios menores, de tal forma que todo o repositório possa ser reconstruído a partir dos fragmentos gerados. Da mesma forma, a distribuição de repositórios de proveniência pode melhorar o desempenho e a qualidade das consultas nas máquinas dos cientistas, mas devemos apresentar estratégias que descubram onde um dado de proveniência específico está armazenado, especialmente quando o repositório de proveniência está fragmentado (pesquisa por dados de proveniência em repositórios replicados é bastante trivial, uma vez que submetemos a mesma consulta para todos os repositórios). A aplicação dessas regras em ambientes elásticos é interessante pois nos permite adotar estratégias diferenciadas para cada problema a ser enfrentado.

De acordo com (Gehani *et al.* 2010) há três tipos de estratégias que podem ser adotadas para a gerência dos dados de proveniência nos SGWfC distribuídos: (i) os dados de execução e os dados de proveniência são centralizados; (ii) os dados de execução produzidos e consultados pelo SGWfC são distribuídos mas a proveniência é transportada para uma base de dados central periodicamente e (iii) ambos são distribuídos. Nos SGWfCs atuais, soluções estão na primeira e segunda categoria, a qual tem limitações na grande escala. Para superar a limitação em relação à proveniência distribuída, várias soluções na categoria (iii) apoiam a busca pela proveniência no nível do sistema de arquivos, como apresentado em (Zhao *et al.* 2013). No entanto, nestas soluções de sistema de arquivos distribuídos, a proveniência está no nível do sistema operacional (grão fino), sendo pouco conectada ao nível mais abstrato de proveniência. Esta categoria de proveniência armazena um histórico detalhado sobre como o experimento se desenrolou de modo que seja possível reproduzi-lo em todos os detalhes. É essa proveniência que realmente nos interessa neste contexto.

Atualmente alguns SGWfCs fazem uso de dados estatísticos, durante a execução, para poder dimensionar os recursos necessários que deverão ser compatíveis com a complexidade do problema. O objetivo deste trabalho é propor uma estratégia de apoio à execução de *workflows* científicos, oferecendo gerência de proveniência de forma distribuída e adaptável, transparente ao solicitante, por meio de um serviço. Seja para melhorar o desempenho da execução (escalonamento adaptativo), para facilitar a análise de resultados ou para redução de custos. Para atacar essas questões, desenvolvemos algumas estratégias.

Uma dessas estratégias é a proposta de uso de um *data warehouse (DW)* (Kimball e Margy Ross 2002), que através do pré-processamento dos dados de proveniência, pode acelerar a busca por informações estatísticas dos experimentos, que são importantes para análises por parte dos cientistas e também pelas máquinas de execução. Com o uso deste *DW* podemos ter dados pré-processados que podem ser levados para perto do experimento acelerando sua utilização. Como trabalharemos apenas uma parcela da informação, reduzimos a quantidade de espaço necessário para armazenamento e com isso reduzimos o custo financeiro desta operação. Para ilustrar o cenário do uso de dados compartilhados, consideremos o exemplo a seguir, ilustrado na Figura 1.

Kary é uma bióloga que trabalha para um centro de pesquisa em São Paulo, Brasil. Ela está particularmente interessada em bioinformática, na área de genômica

comparativa (Abadio *et al.* 2011), e executa seus experimentos usando o *workflow* científico SciHmm (Ocaña et al. 2011a). Kary executa o SciHmm usando o SciCumulus, um SGWfC que utiliza nuvens computacionais como infraestrutura (Oliveira *et al.* 2010b). Particularmente, Kary executa seus experimentos na nuvem da Amazon EC2. Para superar as flutuações de desempenho do ambiente (algo frequente em nuvens), o SciCumulus usa informações históricas sobre experimentos para tentar prever o consumo de recursos e para estimar a alocação de infraestrutura na nuvem, fazendo isso adaptativamente a cada situação (para reduzir o tempo total de execução e custo financeiro) (Oliveira *et al.* 2012a). Desta forma, Kary precisa de dados históricos para executar seus *workflows* de forma mais eficiente. Considere agora que Kary está iniciando um novo projeto de pesquisa na área de filogenia (Dávila e Kary A. C. S. Ocaña 2011) onde o *workflow* SciPhylomics (Oliveira *et al.* 2012b) deverá ser usado. No entanto, Kary nunca executou SciPhylomics antes, ou seja, não há dados de proveniência relacionados ao SciPhylomics em seu repositório de proveniência local, e ela tem um orçamento limitado para usar e um prazo apertado para finalizar seu estudo.

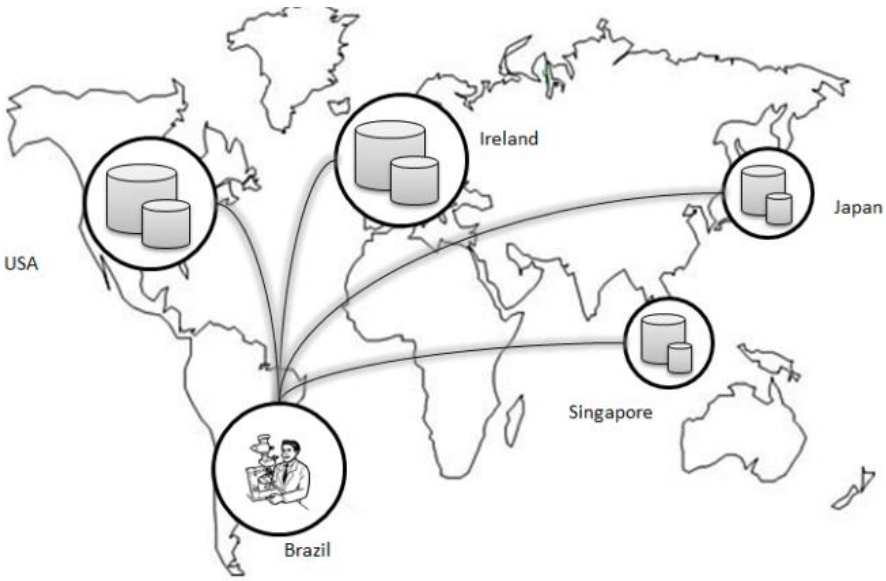


Figura 1 Cenário teórico de bases de dados de proveniência distribuídas

Desta forma, ela precisa usar escalonamento adaptativo oferecido pelo SciCumulus, mas em sua base centralizada ainda não há dados de proveniência disponíveis para a realização da previsão de consumo de recursos. No entanto, Kary sabe que seus colaboradores em todo o mundo (Mathew nos EUA, Charles na Irlanda, Ian em Cingapura e Tatsuo em Tóquio) poderiam ter dados históricos de proveniência sobre filogenia que poderiam ser compartilhados com ela. Nesse contexto, Kary não

precisa dos resultados dos *workflows*. O que lhe interessa são os metadados relacionados às execuções, ou seja, início e fim das atividades, consumo de recursos de infraestrutura, taxa de falha, etc., pois são esses elementos que serão utilizados para dimensionar os recursos necessários da nuvem. No cenário proposto, o *workflow* SciPhylomics só foi executado por Tatsuo e Charles, mas infelizmente Kary não tem essa informação, pois tais dados encontram-se em outras bases. Desta forma, Kary enfrenta a questão de como localizar dados específicos em repositórios de proveniência distribuídos. Na vida real, Kary está restrita a seu repositório centralizado de dados de proveniência. Caso a informação necessária não exista neste banco de dados, ela vai realizar uma execução às cegas do *workflow* para armazenar as primeiras informações sobre o experimento. Esta ação pode ser muito custosa em relação ao tempo que será gasto ou aos recursos financeiros necessários. Com a solução proposta nesta tese, esperamos contornar este problema.

Nesta tese, mostraremos que através da utilização de dados de proveniência de forma distribuída, podemos alcançar bons tempos de resposta das consultas utilizadas, tanto na etapa analítica do experimento, quanto das utilizadas para melhorar o tempo de execução dos *workflows*. Dessa forma conseguimos obter redução de custo na execução do experimento (uma vez que estamos trabalhando em um ambiente pago) e também reduzir o tempo de análise dos pesquisadores. Ao adotar uma abordagem distribuída conseguimos ainda um repositório de proveniência mais fácil de escalar, aumentando assim a quantidade de dados disponíveis. O conjunto de serviços que desenvolvemos é capaz de localizar o dado necessário para a execução de cada experimento. O que propomos neste trabalho é um método híbrido, no qual os dados produzidos por um experimento serão tratados localmente e toda a informação que possa ser compartilhado será armazenada de forma otimizada seguindo uma estratégia baseada em DW. Como será demonstrado, esta abordagem se mostrou interessante de ser adotada, trazendo redução de tempo e custo para os experimentos, dentre outros pontos. Vamos explorar a elasticidade da nuvem, demonstrando que é possível dimensionar os recursos de forma dinâmica, entregando aos experimentos a infraestrutura necessária para que cheguem aos resultados destacados anteriormente em relação ao tempo e o custo proveniente da execução de um determinado experimento.

A abordagem proposta é projetada como um serviço para ser acoplada a SGWfCs que sejam baseados em proveniência para gerenciar sua execução, como o

SciCumulus e o Chiron (Ogasawara *et al.* 2013). A base de dados compartilhada também pode ser usada para as análises dos cientistas através de consultas pré-estabelecidas ou *Ad-Hoc*. Em nossa tese propomos ainda uma maneira de modelar o banco de dados de proveniência de modo a facilitar o acesso a informação associada a um determinado experimento.

1.1 Caracterização do Problema

Nossos esforços estão concentrados na melhor maneira de gerenciar dados de proveniência de forma distribuída para melhorar o desempenho das consultas analíticas e às voltadas ao escalonamento adaptativo em nuvem. As soluções baseadas no uso de nuvem computacional ainda apresentam alguns desafios inerentes à própria tecnologia em questão. Quando pensamos nos ambientes tradicionais de PAD pensamos em uma estrutura pré-configurada, que será usada durante um tempo para se obter um determinado resultado. Quando utilizamos a nuvem mudamos o paradigma, pois agora temos que pensar no melhor uso a se fazer do ambiente de modo a balancear desempenho com o custo envolvido. Explorar a elasticidade deste ambiente é um ponto chave e precisamos ainda de avanços nas aplicações existentes de modo a tirar maior proveito desta característica. À medida que os experimentos se tornam mais complexos, mais dados de proveniência relacionados com estas execuções são capturados, armazenados e consultados. Além disso, alguns SGWfCs são baseados em proveniência e, nestes casos, precisam consultar estes dados em tempo de execução de forma eficiente e por meio de sua análise é possível durante a execução, melhorar o desempenho do experimento.

Nos SGWfC, tais como o Swift (Zhao et al. 2007), VisTrails (Callahan et al. 2006), Pegasus (Deelman *et al.* 2007), dentre outros, os dados de proveniência são gerados somente no final da execução do workflow e não podem ser acessados ao longo do experimento para tomada de decisões, tais como ajuste do ambiente de execução. A proveniência do SciCumulus é gerada em tempo de execução, possibilitando ao cientista fazer uma análise prévia dos dados para avaliar se o experimento está satisfatório ou não. Ou ainda, é possível que o próprio SGWfC tome decisões baseado na análise da proveniência.

Estamos interessados em um cenário atualmente muito comum, no qual SGWfCs estão executando experimentos de forma distribuída em várias máquinas virtuais na

nuvem. Neste contexto, nos parece que os SGWfCs devem, idealmente, gerenciar a proveniência de forma também distribuída, pois dadas as características do cenário apresentado, esta parece ser uma boa estratégia para que a gerência da proveniência seja mais eficiente. Outro ponto importante é oferecer mecanismos eficientes que possibilitem a consulta à proveniência gerada já em tempo de execução para assim tornar possíveis as estratégias de escalonamento de recursos ou *steering* (Dias et al. 2011). Outro ponto a ser endereçado é a necessidade de oferecer o recurso ideal para o experimento, de modo a atingir as expectativas dos pesquisadores em relação a tempo de execução e orçamento. Não encontramos hoje uma solução que agregue estas duas características: a gerência da proveniência de forma distribuída com possibilidade de acessar em tempo de execução de forma estruturada toda proveniência gerada e que seja capaz de adaptar o ambiente onde esta proveniência esteja sendo armazenada. As soluções existentes armazenam sua proveniência por meio de registros de execução (*logs)* que, em alguns casos, são transformados em registros de banco de dados após a execução do experimento. Mesmo aquelas soluções que já armazenam sua proveniência de forma estruturada em um banco de dados, o fazem em um ambiente pré-configurado que não será adaptável a um aumento ou redução de demanda, como em nossa proposta. Em nossa tese desejamos ainda atacar a questão das consultas que são executadas durante o experimento pelos SGWfCs com foco em adaptabilidade e as consultas que informarão ao SGWfC se o ambiente de execução do experimento está sendo suficiente ou se precisa ser redimensionado (para maior ou menor capacidade). Algumas consultas podem ser executadas milhares de vezes e, ao melhorar o seu tempo de resposta, o tempo total do experimento é diminuído. A consulta à base de proveniência em tempo de execução deve ser feita de forma eficiente para que não gere um *overhead* para o experimento. Uma estratégia aqui proposta é considerar o conjunto dos dados de proveniência como um grande repositório disperso, armazenado em diferentes locais. Para dimensionar o uso da nuvem de maneira ideal, precisamos buscar estratégias que apontem em que momento o sistema deve alocar mais recurso e quando pode liberá-los. Precisamos ainda estudar qual a melhor maneira de armazenar estes dados de proveniência, qual o melhor modelo ou quais adaptações das modelagens de dados existentes devemos fazer para termos uma base de dados otimizada e consistente.

1.2 Hipótese: Serviço Para Gerência de Dados de Proveniência na Nuvem Baseado em Uma Abordagem Distribuída e Adaptativa

Se criarmos um conjunto de serviços distribuídos e adaptativos capazes de gerenciar os dados de proveniência de forma distribuída, esperamos ter ganhos em escalabilidade, desempenho das consultas, disponibilidade dos dado e redução de custos. Com a criação deste conjunto de serviços, dados de proveniência serão gerenciados por meio de técnicas de banco de dados distribuídos e de *Data Warehouse*. Com a utilização de estratégias para melhorar a utilização dos recursos da nuvem, seremos capazes de reduzir os gastos com infraestrutura. Explorando sua elasticidade será possível ter uma utilização mais eficaz deste ambiente. Para que a utilização destas estratégias seja acoplável a um SGWfC já existente, a implementação deverá ser criada sob a forma de serviços que poderão ser usados em tempo de execução para melhorar o desempenho dos experimentos.

1.3 Organização do Texto

O texto foi organizado da seguinte forma: No Capítulo 2 discorremos sobre a fundamentação teórica de experimentos científicos em larga escala. Já no Capítulo 3, apresentamos as estratégias empregadas na literatura para armazenamento de dados de proveniência e indicamos qual foi a adotada por nós. No Capítulo 4, apresentamos o ProvSearch, nosso conjunto de serviços responsáveis por tornar o uso da proveniência em ambiente distribuído mais eficiente sem trazer com isso grande complexidade. Apresentamos ainda nosso DW de proveniência e todo o caminho percorrido até chegar a sua modelagem. No Capítulo 5, são apresentados os resultados experimentais de nossa tese, e concluímos, no capítulo 6, capítulo no qual também resumimos as contribuições trazidas para nossa área de pesquisa.

Capítulo 2 - Fundamentação Teórica de Experimentos Científicos em Larga Escala

Nas próximas seções abordamos as questões envolvidas em experimentos científicos computacionalmente dependentes de PAD. Segmentamos o texto, dividindo-o da seguinte maneira: A Seção 2.1 apresenta o conceito de experimento científico baseado em simulação, que é o nosso objeto de estudo, enquanto que a Seção 2.2 define o ciclo de vida de um experimento científico. A Seção 2.3 apresenta o conceito de proveniência de dados no contexto de *workflows* científicos e a Seção 2.4 trata de pontos relevantes das nuvens computacionais.

2.1 Experimentos Científicos Baseados em Simulação

Nos últimos anos, a utilização de sistemas de *workflows* para a execução de experimentos científicos baseados em simulações computacionais vem aumentando consideravelmente (Mattoso *et al.* 2010). Os experimentos em larga escala requerem recursos computacionais cada vez mais potentes para a sua execução e em muitos casos são realizados por equipes geograficamente dispersas (Gorton *et al.* 2008).

Formalmente, um experimento científico pode ser definido como "um teste executado sob condições controladas, que é realizado para demonstrar uma verdade conhecida, examinar a validade de uma hipótese, ou determinar a eficácia de algo previamente não explorado" (Soanes e Stevenson 2003). Um experimento também pode ser definido como "uma situação, criada em laboratório, que visa observar, sob condições controladas, a relação entre os fenômenos de interesse" (Jarrard 2001). Neste contexto, a palavra "controle" ou o termo “condições controladas” são usados para indicar que há esforços para eliminar, ou pelo menos reduzir, o máximo possível, os erros ocasionais durante uma observação planejada (Juristo 2001). Sendo assim, podemos concluir que um experimento científico tenta seguir um conjunto de ações controladas, previamente planejadas. Estas ações controladas incluem variações de testes e seus resultados são geralmente comparados entre si para aceitar ou refutar uma hipótese científica. Os experimentos científicos são a maior preocupação da comunidade científica (Mattoso *et al.* 2010).

Em nosso trabalho estamos interessados na categoria de experimentos científicos baseados em simulação (Travassos e Barros 2003), ou simplificadamente, experimentos científicos. Este tipo de experimento é utilizado nos mais diversos domínios científicos como, por exemplo, análises filogenéticas (Ocaña et al. 2011b), genômica comparativa (Ocaña et al. 2011a), processamento de sequências biológicas (Lemos *et al.* 2004), estudos na área de saúde (de Almeida-Neto *et al.* 2011, Goncalez *et al.* 2011, Patavino *et al.* 2012, Sabino *et al.* 2011), prospecção de petróleo em águas profundas (Carvalho 2009, Martinho *et al.* 2009, Ogasawara *et al.* 2011, Oliveira *et al.* 2009), mapeamento dos corpos celestes (Hey *et al.* 2009), ecologia (Hartman *et al.* 2010), agricultura (Fileto *et al.* 2003), busca de genes ortólogos dos tripanosomas causadores de doenças tropicais negligenciadas (Coutinho *et al.* 2011, Dávila *et al.* 2008), dinâmica de fluidos computacional (Guerra *et al.* 2009, 2012, Guerra e Rochinha 2009a, 2009b, 2010, Lins *et al.* 2009), estudos fisiológicos (Porto *et al.* 2011), monitoramento aquático (Pereira e Ebecken 2011) dentre outros. Todos esses exemplos podem ser considerados de larga escala por consumirem e produzirem um grande volume de dados e são um ponto de inflexão que merece o desenvolvimento de estudos específicos.

Para que seja viável o desenvolvimento destes tipos de experimentos, todos considerados de larga escala, um ambiente de PAD é necessário, pois para sua resolução muitas horas de processamento serão necessárias em ambientes como *clusters* e supercomputadores, grades computacionais, ambientes de computação voluntária e, mais recentemente, as nuvens de computadores. Esses experimentos são especialmente complexos de serem gerenciados devido à grande quantidade de programas envolvidos na simulação e a quantidade de recursos computacionais necessários. Não é trivial controlar o volume de dados manipulados, evitar e contornar falhas de execução, etc. Essa tarefa se torna ainda mais complexa se necessitarmos capturar e armazenar descritores da execução para garantir a reprodutibilidade do mesmo (Davidson e Freire 2008). Essa captura e armazenamento é uma característica fundamental para que um experimento seja considerado "científico" de fato.

Em muitos casos, os experimentos científicos são formados por um conjunto de programas que devem ser executados de forma encadeada, produzindo e consumindo uma grande quantidade de dados. Um complicador é adicionado a esse contexto que consiste no fato de os ambientes computacionais de processamento poderem ser heterogêneos. Nesses experimentos, cada programa pode ser executado consumindo um

grupo específico de parâmetros e dados, cada qual com a sua própria semântica e sintaxe. A saída de um programa é normalmente utilizada como entrada para outro programa no encadeamento (Cavalcanti *et al.* 2005). Outro ponto importante a ser destacado é a característica dos grupos de pesquisa que trabalham de forma dispersa e precisam compartilhar dados dos experimentos e dados de proveniência (a definição de proveniência será mais bem explicada a seguir). À medida que a quantidade de informação a ser manipulada aumenta, modelos de gerência de informação já consolidados podem não ser capazes de explorar todas as possibilidades de uso de ambientes configurados para execução em nuvem.

Workflows científicos apresentam uma estratégia mais interessante para controlar o encadeamento de programas. Ao invés de usarmos uma abordagem *ad hoc* manual ou baseada em *scripts*, os *workflows* científicos podem ser definidos como uma abstração para modelar o fluxo de atividades e de dados em um experimento. Em *workflows* científicos, essas atividades são geralmente programas ou serviços que representam algoritmos e métodos computacionais sólidos (Barker e van Hemert 2008). Esses *workflows* são controlados e executados por um Sistema de Gerência de *Workflow* Científico (SGWfC), que visa apoiar a configuração e execução dos *workflows*. Há muitos SGWfCs disponíveis, como o Kepler (Altintas *et al.* 2004), o Taverna (Hull *et al.* 2006), o VisTrails (Callahan *et al.* 2006), o Pegasus (Deelman *et al.* 2007), o Askalon (Fahringer *et al.* 2005), o P-Grade (Glatard *et al.* 2007, Kertész *et al.* 2007), o DagMan (Couvares *et al.* 2007), o Triana (Taylor *et al.* 2007b), o WOODS (Medeiros *et al.* 2005) e o Swift/T (Wozniak et al. 2013), cada uma com suas próprias características, vantagens e desvantagens.

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

de vida do experimento científico, vamos formalizar os conceitos de *workflow* científico. Vejamos uma primeira definição:

Segundo o *Workflow Management Coalition* (WfMC) (WfMC 1997), é: "A automação de um processo de negócio, completo ou apenas parte dele, através do qual documentos, informações ou tarefas são transmitidos de um participante a outro por ações, de acordo com regras procedimentais".

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

Com o avanço das técnicas, temos visto a utilização do *workflow* em outros ramos da ciência, como a biologia e a astronomia, inicialmente. Áreas que anteriormente executavam seus experimentos manualmente ou por meio de *scripts* foram naturalmente migrando para a utilização de *workflows* científicos como abstração para especificação de seus experimentos (Hey *et al.* 2009). O termo *workflow* científico é usado para descrever *workflows* em algumas destas áreas da ciência, nas quais se compartilham as características de manipulação de grandes volumes de dados, heterogeneidade na representação dos dados e demanda por alto poder de processamento (Taylor *et al.* 2007a). Por simplificação, nesta tese um *workflow* científico é definido como um grafo acíclico dirigido (do inglês *Directed Acyclic Graph* ou DAG) (Cormen *et al.* 2009) chamado *W(A, Dep)*. Os nós, ou nodos, ($A = \{a_1, a_2, ..., a_n\}$) em *W* correspondem a todas as atividades (invocação de programas) do *workflow* a ser executado (em paralelo ou sequencialmente) e as arestas (*Dep*) estão associadas à dependência de dados entre as atividades de *A*.

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

modelada como $dep(a_i, a_j, ds) \leftrightarrow \exists O_k \in Input(a_j) \mid O_k \in Output(a_i)$ e *ds* é o volume de dados transferidos entre essas duas atividades (independente da unidade de medida utilizada) (Oliveira 2012).

2.2 Ciclo de Vida do Experimento Científico

A seguir apresentamos o modelo do ciclo de vida de um experimento científico, de acordo com Mattoso *et al.* (2010). A Figura 2 apresenta o ciclo de vida de um experimento cientifico em larga escala. Como podemos observar na imagem, múltiplas etapas são percorridas pelo cientista várias vezes no curso do experimento, de acordo com as seguintes fases: execução, composição e análise. Cada fase possui um subciclo independente, que é percorrido em momentos distintos do curso do experimento e manipula diferentes tipos de descritores de proveniência.

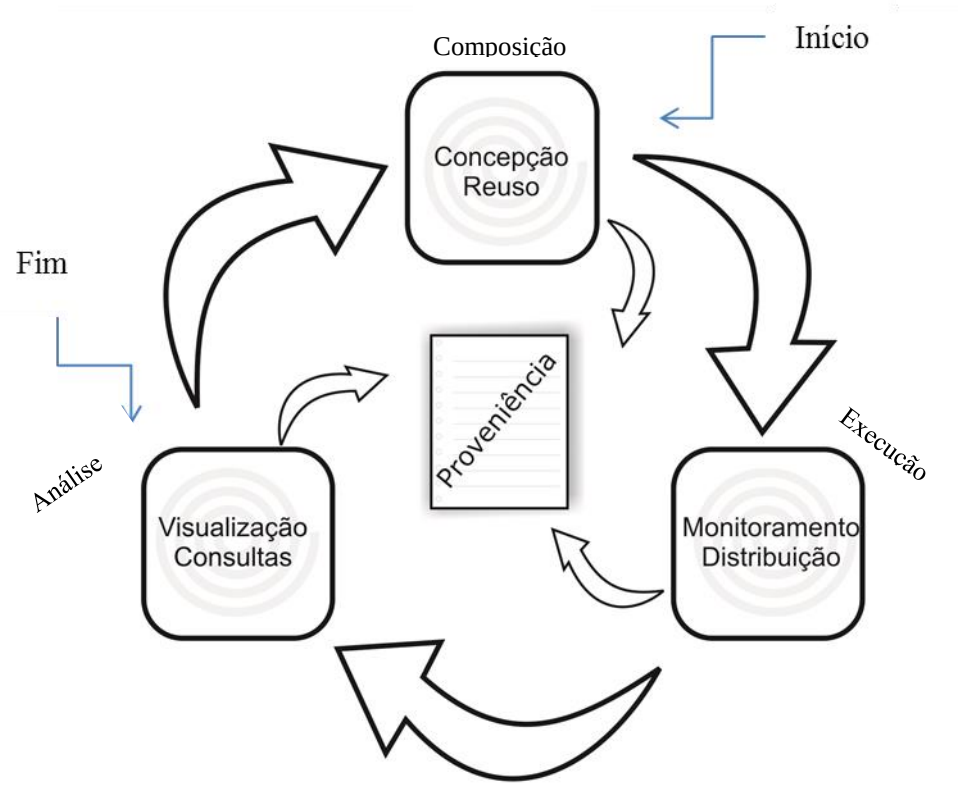


Figura 2 Ciclo de vida do experimento científico adaptado de Mattoso *et al.* 2010

A fase de composição é responsável pela estruturação e criação de todo o experimento, mas principalmente é responsável por instituir a sequência lógica de atividades, os tipos de dados de entrada e parâmetros que devem ser fornecidos e os

tipos de dados de saída que são gerados. Esta fase pode ser ainda decomposta em duas subfases: a concepção e o reuso. A concepção é responsável pela criação do experimento enquanto que o reuso é responsável por recuperar um experimento já existente e adaptá-lo para um novo propósito.

No entanto, é na fase de execução que concentramos nossos esforços, pois é a fase com maior dependência computacional. Esta é a etapa responsável por tornar concreta e executável uma especificação de *workflow* de forma que possa ser executada por um SGWfC. Portanto, nesta fase definimos exatamente os dados de entrada e os parâmetros a serem entregues ao SGWfC, a fim de executar o experimento, ou seja, criamos uma instância do *workflow* para ser executada. Esta fase pode ser decomposta em duas subfases: distribuição e monitoramento. A subfase de distribuição está relacionada com a necessidade da execução de atividades do *workflow* em ambientes de PAD, principalmente devido a necessidades de desempenho. A subfase de monitoramento está relacionada à necessidade de verificar o estado atual da execução do *workflow*, uma vez que este pode executar por um longo tempo.

Finalmente, a fase de análise é responsável por estudar os dados gerados pelas fases de composição e execução. Essa fase é altamente dependente dos dados de Proveniência (Freire *et al.* 2008) que foram gerados nas fases anteriores e pode ser decomposta em duas novas subfases: visualização e consulta. Além disso, na fase de análise, o cientista pode enfrentar duas situações diferentes ao analisar os resultados de um experimento: (i) o resultado é provável que seja correto ou (ii) baseado nos resultados, a hipótese é refutada.

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

2.3 Proveniência de Dados

Quando pensamos no termo proveniência, em geral associamos a palavra procedência ou origem. Procurando no dicionário, encontramos a seguinte definição: "*s.f.* Origem, procedência: mercadorias de proveniência estrangeira; uma pessoa de proveniência ignorada" (Buarque de Holanda 2004). De todas as definições existentes, a que nos

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

Historicamente podemos dividir o registro da proveniência em dois grupos: a prospectiva e a retrospectiva (Freire *et al.* 2008). A proveniência prospectiva está interessada em capturar e armazenar os dados relativos à estrutura do processo que levou à geração de um determinado produto. Ou seja, no contexto de *workflows* científicos, a proveniência prospectiva está preocupada em capturar os dados relativos à estrutura do *workflow* bem como as configurações de ambiente utilizadas para executá-lo. Já a proveniência retrospectiva tem o foco em capturar os dados e seus descritores produzidos a partir da execução de um determinado processo, no nosso caso, de um determinado *workflow*. Dados de proveniência retrospectiva englobam tempos de início e fim de execução, arquivos produzidos, erros que ocorreram, informações de desempenho de atividades, entre outros.

Os dados de proveniência podem ser capturados de diferentes formas, mas a principal questão na captura se refere à granularidade dos mesmos. Buneman e Tan (2007) criaram uma classificação interessante a do "grão fino" (do inglês *fine grain*) e do "grão grosso" (do inglês *course grain*). A proveniência de "grão grosso" se refere ao registro do histórico de um conjunto de dados, enquanto que a proveniência de "grão

fino" se refere ao registro da linhagem de um item específico de dados. Nesta tese, abordaremos os dois tipos de proveniência.

Para termos dados de proveniência estruturados segundo modelos já bem discutidos e internacionalmente difundidos, começamos seguindo os modelos de dados que se baseiem nas recomendações iniciais do W3C PROV (Moreau e Missier 2013) (evolução do *Open Provenance Model* (OPM) (Moreau et al. 2008)), modelo que propõe uma representação genérica de proveniência. O W3C PROV expressa as relações causais entre Processos, Agentes, Artefatos e Papéis. O Modelo do W3C PROV foi ainda especializado para representar a questão da proveniência obtida da execução de *workflow* científico. Esta especialização recebeu o nome de PROV-Wf (Oliveira *et al.* 2014)(Costa *et al.* 2013).

Uma das principais vantagens de se utilizar recomendações como estas é garantir a interoperabilidade de descritores de proveniência oriundos de ambientes heterogêneos independentemente da tecnologia e dos SGWfCs utilizados. Este ponto é fundamental para tornar nossa proposta desacoplada da solução de SGWfC e será revisitado adiante. Por esse motivo, principalmente o W3C PROV e o PROV-Wf já vêm sendo utilizados por alguns SGWfCs como forma de representar a proveniência de forma única, facilitando assim a sua utilização em diferentes contextos, independente do sistema de gerência que a gerou.

2.4 Nuvem Computacional

A computação em nuvem (do inglês *Cloud Computing*) (Kim *et al.* 2009, Marinos e Briscoe 2009, Napper e Bientinesi 2009 , Vaquero *et al.* 2009, Wang *et al.* 2008) surgiu como um novo paradigma de computação distribuída, onde serviços baseados na *Web* têm como objetivo permitir a diferentes tipos de usuários obterem uma grande variedade de recursos de *software* e *hardware*. Desta forma, a computação, como costumávamos conhecer há alguns anos, mudou completamente. A nuvem passou a apresentar uma nova possibilidade em relação aos já consolidados ambientes *desktop*. Os usuários são capazes de acessar programas, documentos e dados de qualquer dispositivo capaz de se conectar à internet. Mesmo aqueles de hardware reduzido, como celulares. Conceitos de elasticidade, disponibilidade e segurança estão diretamente relacionados a esse ambiente e novos desafios surgem desse novo cenário. A computação em nuvem apresenta um grande potencial para apoiar experimentos

científicos que necessitem de ambientes de PAD. A natureza das necessidades da computação científica se encaixa bem com a flexibilidade e a elasticidade sob demanda, oferecida pelas nuvens, onde o uso efetivo de seus recursos pode reduzir o custo real com equipamentos e sua consequente manutenção, além da questão das atualizações constantes de *software* e *hardware*.

As nuvens podem proporcionar um ambiente adequado para a execução paralela de *workflows* científicos que demandem PAD. Entretanto, muitos problemas continuam sem solução como, por exemplo, os gargalos de transferência de dados, a flutuação de desempenho das máquinas virtuais, a falta de interoperabilidade dos ambientes, etc. Desta forma, novas propostas são necessárias para fornecer soluções mais robustas para os cientistas executarem seus *workflows* em paralelo em nuvens, de forma sistemática.

Capítulo 3 - Modelo de Proveniência

Uma parte importante da gerência de dados de proveniência é o modelo utilizado para representar esses dados. Nosso modelo atual é uma evolução de modelos mais antigos como o ProvOne (Ludascher et al. 2014). Que por sua vez já é uma evolução do *Open Provenance Model* (OPM) (Moreau *et al.* 2008a, 2008b). A seguir abordamos em mais detalhes essa questão.

3.1 ProvWf X ProvOne

O ProvONE é um modelo para a representação da proveniência produzida pela execução de um *workflow* científico (Ludascher et al. 2014). Este nome foi escolhido por conta do contexto onde este modelo foi desenvolvido, o projeto DataOne, o qual trabalha para a criação de uma infraestrutura que apoie a criação de uma grande base de dados científicos que possa ser utilizada por toda a comunidade científica do Globo. Como os dados podem ser produzidos por uma grande quantidade de SGWfC, o ProvONE está sendo desenvolvido para unificar as diferentes modelagens existentes, dando aos dados de diferentes origens uma mesma forma, de modo que seja possível criar uma interoperabilidade e facilitar o entendimento dos mesmos. Esse modelo de dados surgiu de uma evolução do PROV (Moreau e Missier 2013), que também trata-se de um conjunto de regras para modelar proveniência.

Em 2012, antes do surgimento do ProvOne, nossa equipe de pesquisa já havia sentido a necessidade de criar uma modelagem que se adequasse às necessidades de representação dos dados de proveniência dos experimentos científicos. Por conta desse anseio, propusemos PROV-WF (Costa et al. 2013), uma especialização do PROV. Este trabalho foi apresentado em um congresso onde muitos pesquisadores responsáveis pela criação do ProvOne participaram. Na época eles entenderam a necessidade de se ter uma modelagem específica para os dados de proveniência dos experimentos científicos.

Implementamos PROV-WF em um banco de dados relacional, permitindo a análise de dados de proveniência de uma forma bem estruturada, com SQL. No entanto, dependendo da experiência dos cientistas ou a complexidade das consultas, pode ser muito difícil para o cientista lidar com um esquema de dados de proveniência. Mas se o cientista trabalhar em conjunto com um especialista, no sentido de criar um conjunto de

consultas, este poderá, sempre que for necessário, acessar os dados de proveniência facilmente, com suas consultas pré-configuradas.

A seguir detalharemos PROV-WF e como este é utilizado em conjunto com os SGWfCs, em especial o SciCumulus. Apesar dos avanços trazidos por ProvOne, PROV-WF ainda segue as diretrizes básicas que determinam o ProvOne. Sendo assim, não foi preciso remodelar nossa base de dados, pois mesmo datando de 2012 ainda está de acordo com as boas práticas apontadas pelo grupo DataOne.

3.2 Exemplo Modelagem de Proveniência

3.2.1 SciCumulus

Utilizaremos como exemplo de SGWfC o SciCumulus. Apresentaremos suas principais características e como a proveniência armazenada por ele está modelada. Conhecendo um pouco mais de um SGWfC que trabalhe na nuvem, temos uma dimensão da complexidade envolvida.

3.2.1.1 Arquitetura

O SciCumulus foi projetado para ser um SGWfC, que orquestra a execução paralela do *workflow*, utilizando a infraestrutura de nuvem. O SciCumulus orquestra a execução das *tarefas* do *workflow* científico em um conjunto distribuído de máquinas virtuais (*cluster* virtual), oferecendo dimensionamento dos recursos durante o curso de execução do *workflow*. O SciCumulus (Oliveira *et al.* 2010b, 2010c, 2012c, de Oliveira *et al.* 2011) foi projetado para seguir uma arquitetura de quatro camadas, sendo a primeira delas a Camada Cliente. Seus componentes são instalados nas máquinas dos cientistas. Os componentes desta camada despacham as atividades do *workflow* para serem executadas em paralelo no ambiente de nuvem. A segunda camada é a de Distribuição. Esta cria e gerencia a execução paralela de *tarefas* em uma ou mais máquinas virtuais instanciadas em um ambiente de nuvem. Temos também a Camada de Execução: seus componentes são instalados nas várias máquinas virtuais envolvidas na execução paralela das *tarefas*. Ela é responsável pela execução de programas encapsulados em tarefas e por coletar e armazenar dados de proveniência. Finalmente temos a quarta camada, a Camada de Dados, responsável por armazenar dados de entrada e dados de proveniência consumidos e gerados pela execução paralela das atividades dos

workflows científicos. Além disso, essa camada tem informações sobre as características do ambiente coletadas por um agente que monitora o experimento.

Espera-se com esta aplicação isolar o cientista das complexidades envolvidas no gerência de execuções de atividades em paralelo. Sua configuração é relativamente simples, e uma vez feita, o cientista terá em suas mãos um poderoso ambiente para executar seus experimentos. De maneira simples é possível ainda ter acesso à proveniência dos dados de execução, muito importante para o trabalho do cientista. No SciCumulus é possível utilizar dois tipos de paralelismo, mas no contexto deste trabalho exploramos apenas um, o paralelismo de varredura de parâmetros.

O controle desse tipo de paralelismo é difícil quando a gerência é feita de forma *ad-hoc* em ambientes de nuvem, devido à grande quantidade de tarefas a serem gerenciadas. Desta forma, o SciCumulus fornece a infraestrutura necessária para dar o apoio computacional para o paralelismo de *workflows* com coleta de proveniência distribuída. A arquitetura do SciCumulus é simples e pode ser teoricamente implantada em qualquer ambiente de nuvem (como o Amazon EC2 ou o GoGrid), diminuindo o esforço de desenvolvimento por parte dos cientistas. A seguir detalhamos a arquitetura do SciCumulus e explicaremos cada um de seus componentes. A Figura 3 apresenta a arquitetura conceitual do SciCumulus em seus quatro níveis principais.

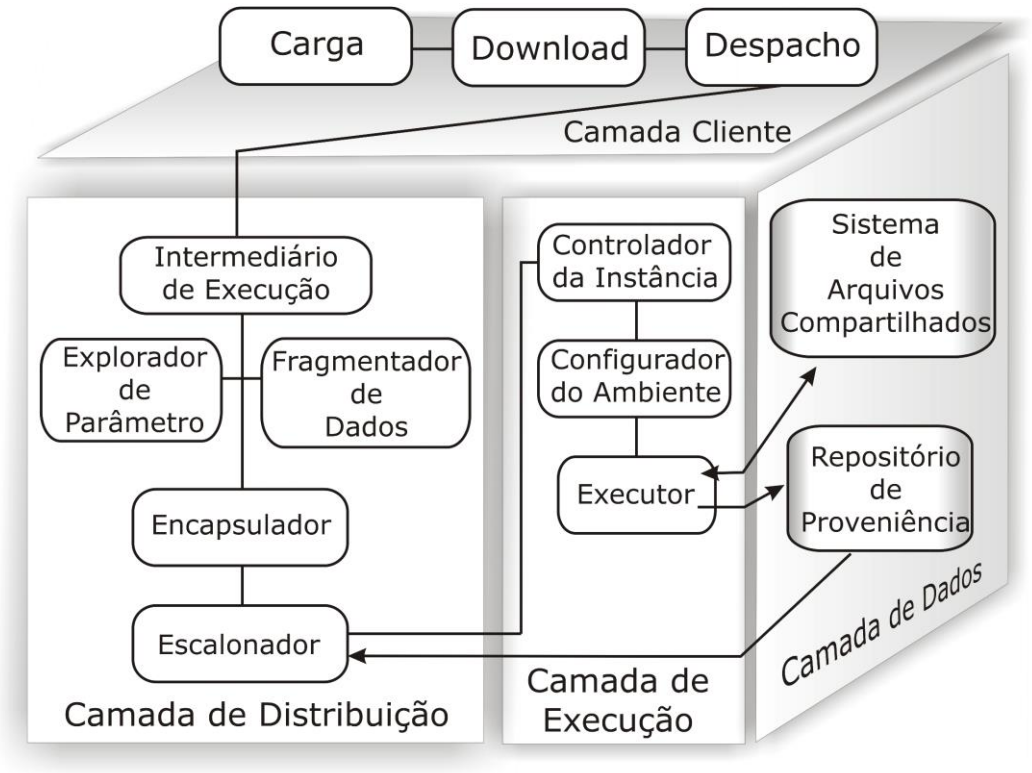


Figura 3 Arquitetura Conceitual do SciCumulus adaptada de Oliveira (2012)

A Camada Cliente é responsável por iniciar a execução paralela de atividades de *workflows* na nuvem. Os componentes desta camada se situam na fase de configuração do ciclo de vida do *workflow* executado em nuvem, uma vez que a transferência dos dados e o despacho da atividade para a nuvem são configurações prévias à execução paralela do *workflow*. Eles iniciam a execução paralela do *workflow* na nuvem. Há três componentes principais nesta camada: o Componente de Carga, o Componente de *Download* e o Componente de Despacho. O Componente de Carga é responsável por mover os dados para a nuvem, enquanto que o Componente de *Download* fica responsável por recolher das máquinas virtuais os dados gerados e colocá-los na máquina do cientista, ou seja, esses componentes são responsáveis pela preparação de dados de entrada e sua carga e pela transferência dos dados de saída. Aqui cabe uma reflexão em relação ao consumo de banda de transferência de dados. Os componentes de carga e de *download* são componentes que devem ser preferencialmente invocados quando o experimento científico lida com dados de entrada de tamanho de pequeno a médio e produz também uma saída de tamanho de pequeno a médio para ser transferida.

Isto se dá uma vez que a transferência de dados para o ambiente de nuvem ocorre sobre uma conexão comum com a internet. Se os cientistas pretendem executar uma varredura de parâmetros usando um arquivo de 100 MB como entrada para os programas do *workflow*, é possível transferir essa quantidade de dados sem grandes impactos no desempenho do *workflow*. Entretanto, esta transferência pode não ser viável quando os cientistas desejam lidar com 100 GB de dados de entrada, por exemplo. Desta forma, os cientistas podem transferir dados de suas estações de trabalho para a nuvem *a priori*, para que não ocorra um impacto de transferência de dados. O Componente de Despacho cria um arquivo de configuração (que pode ser representado como um arquivo XML) que armazena a configuração para a execução distribuída (*i.e.* ele informa os parâmetros que devem ser explorados, por exemplo) e inicia o processo de execução das *tarefas* na nuvem se comunicando diretamente com os componentes da Camada de Distribuição. Toda a informação presente no XML em conjunto com os dados de proveniência gerados nos fornecem informações importantes sobre o experimento.

Na Camada de Distribuição ocorre o controle da execução de atividades paralelas em nuvens por meio da criação e gerência das *tarefas* que contêm o programa a ser executado, a estratégia paralela a ser seguida (indicando quais parâmetros explorar, por exemplo), os valores dos parâmetros e dados de entrada a serem consumidos. A camada é composta por seis componentes: o Intermediário de Execução, o Explorador de Parâmetros, o Fragmentador de Dados, o Encapsulador e o Escalonador. Estes componentes se situam na camada de execução do ciclo de vida do *workflow* executado em nuvens. O Intermediário de Execução faz a ligação entre a Camada Cliente e a Camada de Distribuição. Ele inicia a execução paralela do *workflow* por meio da análise dos metadados fornecidos pelo Componente de Despacho na Camada Cliente. De acordo com a estratégia paralela escolhida pelos cientistas, esse componente pode invocar o Explorador de Parâmetros e/ou o Fragmentador de Dados. O Intermediário de Execução também é responsável por enviar mensagens de sincronização para o Componente de Despacho na Camada Cliente. O Explorador de Parâmetros manipula as combinações de parâmetros recebidos pela Camada Ciente para uma atividade específica do *workflow* que está sendo paralelizado (ou o sub-*workflow*, caso a tarefa seja composta de mais de uma atividade). Este componente produz as combinações dos valores dos parâmetros a serem consumidos por diversas *tarefas*.

O Fragmentador de Dados atua quando os dados de entrada devem ser fragmentados (e armazenados em um sistema de arquivos compartilhado, por exemplo), e gera subconjuntos dos dados que podem ser distribuídos ao longo das máquinas virtuais durante a execução, juntamente com os valores de parâmetros. É importante ressaltar que o Componente Fragmentador de dados invoca um programa definido pelos cientistas que fragmenta os dados de entrada. O Encapsulador gera todas as *tarefas* a serem transferidas para as máquinas virtuais envolvidas na execução paralela.

O Escalonador define quais máquinas virtuais receberão uma *tarefa* específica para executar. Note que as máquinas virtuais podem ser fornecidas por qualquer provedor de nuvem. O Escalonador tem de levar em conta as máquinas virtuais disponíveis para uso, as permissões para acesso e o poder computacional de cada uma delas. O Escalonador do SciCumulus funciona em dois modos diferentes: estático e adaptativo. No modo estático, o Escalonador considera apenas as máquinas virtuais disponíveis no início da execução, o que pode levar a problemas de desempenho. Por conta disto, este modo de trabalho do Escalonador não é indicado.

Já o modo adaptativo visa analisar os recursos disponíveis para a nuvem durante o curso da execução do *workflow*, a fim de usar mais (ou menos) máquinas virtuais e adaptar o tamanho do grupo de atividades de acordo com a capacidade de processamento das máquinas virtuais disponíveis. Baseado no escalonamento escolhido, o Escalonador transfere as *tarefas* para as máquinas virtuais disponíveis e controla a sua execução. Além disso, pode também combinar os resultados parciais, se necessário, para organizar os resultados finais a serem transferidos para a Camada Cliente.

A Camada de Execução é responsável por invocar os códigos executáveis nas diversas máquinas virtuais disponíveis para uso. Ela é composta por três componentes principais: o Controlador de Instância, o Configurador do Ambiente e o Executor. Similarmente aos componentes da Camada de Distribuição, esses componentes fazem parte da fase de execução do ciclo de vida de um *workflow* executado em nuvem. O Controlador de Instância faz a conexão entre a Camada de Distribuição e a Camada de Execução. Ele é também responsável por controlar o fluxo de execução na instância quando a *tarefa* é associada a dois ou mais aplicativos para serem executados (um sub-*workflow*, por exemplo). O Configurador do Ambiente é responsável por montar toda a estrutura para a execução do *workflow*, definindo as variáveis de ambiente e criando diretórios necessários para gerenciar a execução. Ele desencapsula a tarefa, cria réplicas

do programa para um local específico e cria espaços de trabalho (do inglês *workspaces* - áreas diferentes no sistema de arquivos compartilhado para armazenar informações sobre uma execução de uma tarefa específica) para armazenar arquivos de configuração e os dados a serem consumidos. O Executor invoca a aplicação específica e coleta os dados de proveniência, armazenando-os em um repositório também localizado na nuvem.

Finalmente, a Camada de Dados contém todos os repositórios de dados utilizados pelo SciCumulus. Ela possui dois componentes principais: o Sistema de Arquivos Compartilhados, e o Repositório de Proveniência. Os componentes da Camada de Dados se situam na fase de análise do ciclo de vida do *workflow* executado em nuvem. O repositório de proveniência contém dados de proveniência importantes (Freire *et al.* 2008) coletados durante o curso do experimento. Além disso, ele contém informações sobre o ambiente de nuvem, como os tipos de máquinas virtuais disponíveis, as máquinas virtuais instanciadas no momento e as características de localidade destas máquinas. Essa informação é captada por um agente que monitora o ambiente à procura de mudanças no mesmo (novas máquinas virtuais disponíveis ou máquinas virtuais destruídas). O Sistema de Arquivos Compartilhado contém todos os dados de entrada consumidos por diversas tarefas durante a execução paralela.

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

3.2.1.2 **Modelo de Proveniência do SciCumulus**

O SciCumulus foi projetado e implementado para operar em nuvens de computadores, assim como seu modelo de proveniência. Ele representa todas as informações sobre o experimento que está sendo executado e sobre o ambiente de nuvem propriamente dito. Este modelo de proveniência tem como base o W3C PROV (Moreau e Missier 2013) que visa facilitar a interoperabilidade de metadados de proveniência oriundos de ambientes heterogêneos. O PROV é uma recomendação aberta e focada no ponto de vista da interoperabilidade entre os dados de proveniência dos diversos SGWfC, sempre respeitando a comunidade de seus colaboradores, revisores e usuários. A ideia principal do PROV é representar as relações causais entre processos, agentes, artefatos e papéis envolvidos em uma execução de *workflow* científico, seja ele paralelo ou não. O PROV é uma representação padrão de proveniência de dados para a maioria dos SGWfC

(Altintas *et al.* 2004, Callahan *et al.* 2006, Deelman *et al.* 2007, Fahringer *et al.* 2005, Hull *et al.* 2006, Taylor *et al.* 2007b, Zhao *et al.* 2013) e ao utilizá-lo, na verdade, estamos seguindo um padrão de representação/armazenamento de dados já consolidado, o que dá credibilidade ao nosso trabalho. A Figura 4 apresenta o esquema projetado para o repositório de proveniência do SciCumulus. Todas as informações relacionadas a execuções anteriores de *workflows* científicos são recuperadas a partir do repositório de proveniência do SciCumulus. Este modelo é representado por meio de um diagrama de classes da *Unified Modeling Language* (UML) (Bezerra 2007, Fowler 2004) e foi modelado com base nos requisitos levantados com cientistas.

O agente cientista (Scientist) representa uma pessoa que utiliza os recursos computacionais para executar o *workflow* na nuvem. Além disso, o cientista está associado a uma máquina virtual (VirtualMachine). A máquina virtual é criada por meio de uma imagem que contém o sistema operacional, programas e dados para a execução de um *workflow*. Além disso, cada máquina virtual tem um custo associado, que se baseia tanto no fornecedor escolhido, como no local onde a máquina virtual é instanciada. O agente máquina virtual estabelece uma associação entre um cientista e um *workflow*, que é composto por um conjunto de atividades (Activity). Cada atividade é responsável por executar um programa em uma máquina virtual com uma configuração específica. A invocação de um programa dentro de um *workflow* (ExecutionInstance) usa um conjunto de parâmetros que podem ser vistos como parte da proveniência retrospectiva do experimento científico. (Costa et al. 2013).

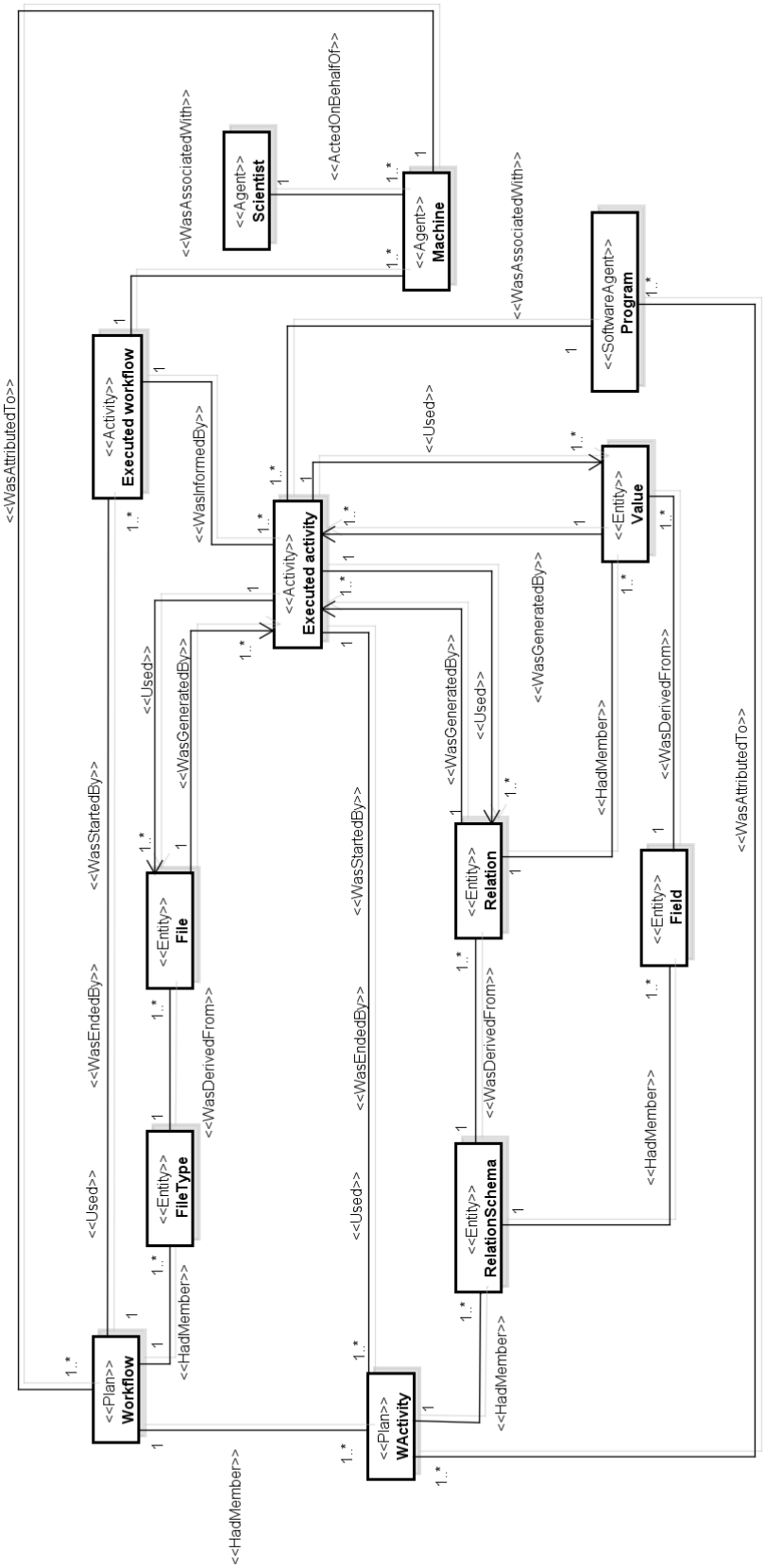


Figura 4 Modelo de proveniência do SciCumulus baseado no PROV

O modelo em questão é composto por três partes principais: a estrutura do experimento, a execução do experimento e a configuração do ambiente.

Podemos destacar ainda a classe Arquivo (File), que representa todos os arquivos consumidos e produzidos por uma execução do workflow, e a entidade Tipo de Arquivo (FileType), que representa o tipo de arquivo esperado pelo programa do workflow. As entidades que consomem e produzem arquivos e dados apresentam associações que expressam suas funções como, por exemplo, "usada por" (used) e "gerada por" (WasGeneratedBy). De acordo com o modelo de dados PROV, um agente é descrito como algo que pressupõe algum tipo de responsabilidade. Além disso, podemos ter um agente de programa como uma entidade que é capaz de executar um programa. Assim, Máquina e Programa são considerados, respectivamente, um agente e um agente de programa. Além disso temos as entidades Relação (*Relation*) que informa os parâmetros usados na invocação de um programa e a entidade Valor (Value) com seu respectivo valor. Posso ter ainda arquivos sendo utilizados na execução de atividades. Estes são representados pela classe Arquivo (File).

Capítulo 4 - ProvSearch

Neste capítulo apresentamos o ProvSearch, um conjunto de serviços capazes de gerenciar a proveniência de forma distribuída, desacoplado da gerência de execução do *workflow*. Ou seja, de maneira transparente ao SGWfC, o ProvSearch se torna responsável por armazenar e consultar a proveniência gerada pelo experimento científico em execução. Fica ainda responsável, sempre que for preciso, por obter dados históricos de experimentos passados. Especificamente neste ponto pensamos em estratégias que visam ao ganho de desempenho na execução de consultas, o que representa uma diminuição no tempo total de execução do experimento. O conjunto de serviços descritos serão disponibilizados na nuvem.

4.1 Modelo Conceitual

Nossa abordagem visa a distribuir os dados de proveniência gerados pela execução distribuída do *workflow* em um conjunto de servidores de banco de dados de proveniência inter-relacionados, formando assim um ambiente integrado (não centralizado) e auto adaptável para a gerência de dados de proveniência de forma distribuída o qual batizamos de ProvSearch. Na figura Figura 5 podemos ver sua estrutura que possui quatro elementos principais: (i) nós de banco de dados, (ii) nós de controle, (iii) API ProvSearch e (iv) *data warehouse* global de proveniência. A API é a interface entre ProvSearch e os SGWfCs existentes. Nesta API existem vários métodos para consulta e armazenamento de elementos dos dados de proveniência. O SGWfC não tem conhecimento de como esses métodos são implementados ou em que local os dados de proveniência estão armazenados. A única informação necessária é a localização das máquinas que estão executando o *workflow*. Uma vez que os SGWfC invocam a API, o nó de controle é acionado. O nó de controle na verdade pode possuir redundância, pois serve como um monitorador dos recursos disponíveis e em utilização. Ele é o responsável por identificar qual o nó do banco de dados irá armazenar os dados de proveniência para uma execução específica. Ele escolhe baseando-se em um modelo de custos que está apresentado na seção 4.3. Uma vantagem da utilização de nuvens é que é possível utilizar as suas características de elasticidade para adaptar a quantidade de recursos de acordo com uma procura crescente. Por não ser trivial identificar quando a demanda está crescendo ou diminuindo, desenvolvemos um componente de

monitoramento dos recursos, baseado no uso de *flash crowd*, conforme explicamos na seção 4.2. Aos nós de bancos de dados e dos *data warehouses* centralizado, existe uma ou N máquinas virtuais associadas.

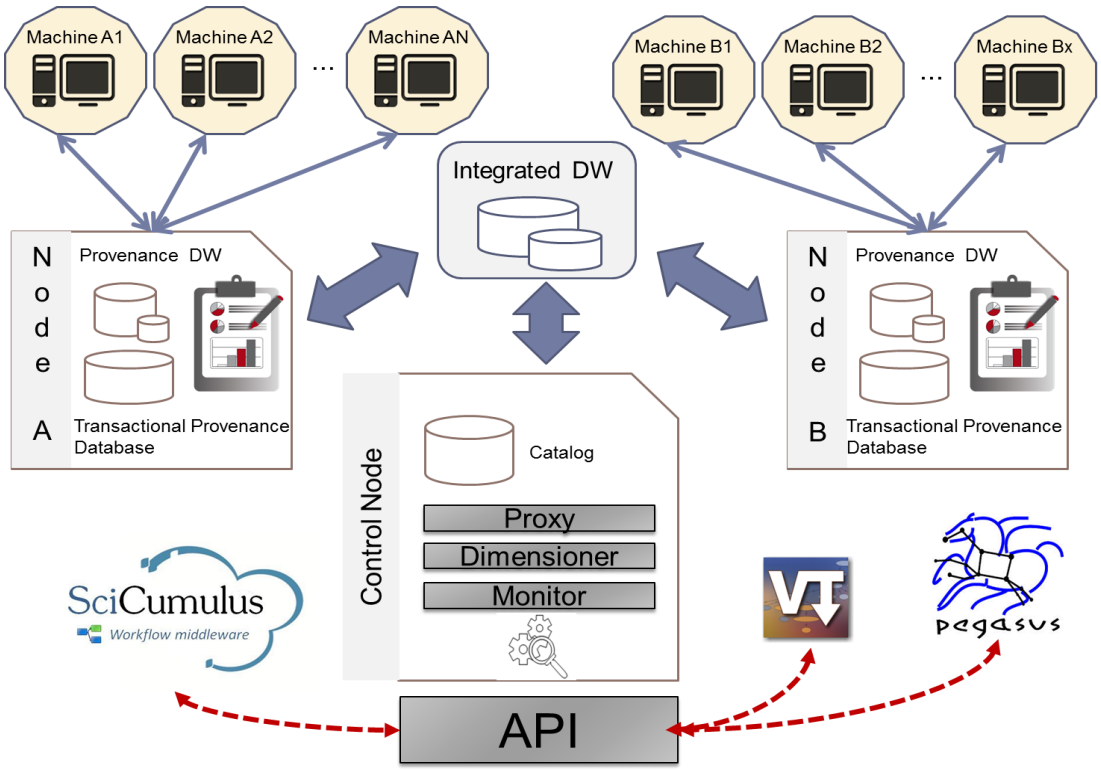


Figura 5. ProvSearch.

Com esta metodologia, o componente *Dimensioner* do nó de controle é capaz de dinamicamente escalonar os recursos da nuvem, utilizando o conceito de entropia marginal (Watanabe 1960) para identificar quando os nós de banco de dados existentes estão sendo utilizados acima de sua capacidade. A entropia marginal H (X) pode ser definida como a informação média contida em uma variável aleatória X com a distribuição de probabilidade p (x) e é utilizada como uma medida de incerteza. Usando essa estratégia, ProvSearch descobre se ele precisa realizar um redimensionamento vertical (quando o poder de computação do nó de banco de dados é aumentado) ou horizontal (quando a quantidade de nós de banco de dados é aumentada). Definido o intervalo para avaliar o nível de utilização de um nó de banco de dados específico, se a partir de um tempo t para um tempo t' identifica-se um aumento X de demanda (de acordo com as regras de flash crowd que serão apresentadas) por processamento, toda a infraestrutura é adaptada.

O uso de entropia e de correlação total comparado a outras tecnologias (tais como crescimento exponencial ou até mesmo outra função) para monitoramento de recursos nos pareceu mais simples. Esta estratégia pode obter os valores máximos para entropia e de correlação total com observações simples sobre a forma como os servidores estão sendo utilizados. Esta adaptação sob demanda tem um custo envolvido, já que estamos usando recursos da nuvem. Este custo financeiro é compartilhado proporcionalmente de acordo com a utilização entre todos os usuários do ProvSearch e tende a ser menor do que um uso individual.

Cada unidade responsável por armazenamento de dados (node) contém um Sistema de Gerência de Bancos de Dados Distribuídos (SGBDDs) instalado com duas bases de dados diferentes: uma para armazenar os dados de proveniência tradicionais (tais como início e fim do tempo, etc.) e o outro apenas com as estatísticas (por exemplo, o tempo médio de execução de um programa específico, a porcentagem de erros para uma máquina específica, etc.). Além do nó de banco de dados, há também um *data warehouse* integrado que está relacionado com a proveniência dos diversos nós de execução de experimentos e contém informação estatística de todos os *workflows* executados. A ideia principal é que este *data warehouse* de proveniência centralizado seja acessível a todos os nós de banco de dados e possa ser consultado quando localmente não existam estatísticas sob o experimento que vai ser executado. Uma vez que ProvSearch encontra as estatísticas que está procurando, as informações encontradas são transferidas do *data warehouse* de proveniência centralizado para o repositório local de proveniência para ser acessado pela máquina de execução. O *data warehouse* local, funcionaria como um *cache* do *data warehouse central.* Como transferimos apenas informações estatísticas, a quantidade de dados trafegados não representa um problema. Com essa estratégia, os dados de proveniência estarão o mais próximo possível de onde o *workflow* é executado. Para realimentar este *data warehouse* de proveniência integrado, no final de cada dia, cada nó de banco de dados executa um processo de ETL (Extract, Transform and Load – processo de extração e transformação de dados) para resumir dados locais e carregá-los no *data warehouse* de proveniência centralizado. Com essa estratégia, ProvSearch pode acessar todas as informações necessárias, melhorando a precisão das análises, com baixo custo na execução do experimento em termos de tempo ou custo.

Para que seja viável a utilização do conjunto de serviços disponibilizados, é preciso que o SGWfC seja compatível com as ideias apresentadas, no que tange o uso da proveniência.

4.2 Flash Crowd

Quando trabalhamos com recursos sob demanda uma das dificuldades envolvidas é achar a maneira mais eficiente de escalonar estes recursos, de modo a ter a melhor relação custo benefício para os usuários. Nesse sentido, foi importante o uso de uma estratégia que nos possibilitasse ao longo da execução dos experimentos determinar quando era a hora de aumentar a capacidade computacional, que trás consigo mais gastos, e quando era a hora de diminuir a infraestrutura montada. Um evento de flash crowd pode ser definido quando o crescimento do número de acessos pode ser mapeado com uma função exponencial ou quando este excede os limites definidos. Em [67], o evento de flash crowd fica caracterizado como o período no qual as taxas de requisições para um domínio específico aumentam exponencialmente.

Semelhante às nossas necessidades, encontramos trabalhos que buscam detectar na nuvem eventos denominados de *flash crowd* (Carvalho 2015) que, em linhas gerais, ocorrem quando servidores de aplicação se tornam incapazes de lidar com um volume repentino de requisições. Por exemplo, quando um site de notícias tem seu volume de acessos aumentado de maneira abrupta por conta de uma catástrofe natural ou um atentado terrorista, como o das Torres Gêmeas. Existem diversos trabalhos que apresentam mecanismos para rápida detecção desses eventos de modo que seja possível reconfigurar os servidores evitando assim o colapso da fonte de dados. Fazendo um paralelo com nossas necessidades nos experimentos científicos, também precisamos de mecanismos que detectem o aumento da necessidade por recurso computacional e posteriormente saiba o momento de reduzir a infraestrutura para evitar gastos desnecessários.

Quando estamos executando um experimento científico onde o SGWfC é baseado no uso de proveniência, em algumas situações o acesso de leitura/escrita à base de proveniência pode se tornar um gargalo para execução do *workflow*. Para evitar esse tipo de problema precisamos de mecanismos para detectar o mais rápido possível esse tipo de situação de modo a contorná-la. O aumento da necessidade de recursos pode ser visto como um evento de *flash crowd* e assim podemos adaptar as estratégias

empregadas para o tratamento desse tipo de evento no contexto dos experimentos científicos. Muitos trabalhos dessa área baseiam a estratégia de detecção pela observação do número de acessos. Essa abordagem pode não ser a estratégia ideal, pois, em alguns casos, poucos acessos já levam a máquina do banco de proveniência para um nível alto de utilização, próximo a 100%. Por outro lado, posso ter um número alto de leituras à minha base de proveniência, sem com isso levar o serviço a uma queda de desempenho. Utilizamos como métrica de monitoramento, o consumo de CPU ao longo do tempo, pois este índice nos fornece uma medida de como a máquina está se comportando à medida que o tempo vai passando. No futuro buscaremos aprimorar esta estratégia utilizando outros indicadores, como uso de memória, por exemplo.

Conforme já mencionamos, nosso serviço de monitoramento, responsável por detectar um pico de utilização da máquina do banco de dados de proveniência, é baseado na utilização de CPU. Uma maneira simples de decidir se a máquina estaria sendo utilizada acima de sua capacidade seria esperar que o índice de utilização de CPU ultrapassasse um certo limite, como 99%, por exemplo. No entanto, este tipo de análise pode ser muito superficial. A máquina após atingir um pico como esse, em um intervalo de tempo muito curto pode voltar a operar em limites mais baixos. Ao trabalhar com detecção baseada nos mecanismos de *flash crowd*, reduzimos a chance de aumentar um recurso computacional só porque este atingiu um pico isolado de utilização acima de certo valor.

Uma maneira de constatar um evento de *flash crowd* é por meio da informação sobre o número de acessos (Stavrou et al. 2004),(Wendell e Freedman 2011). No entanto, mesmo um crescimento exponencial no número de acesso pode não representar um evento de *flash crowd* e nem levar a utilização máxima dos recursos da máquina do banco de dados de proveniência. Em nossa tese usaremos um mecanismo de detecção de eventos de *flash crowd* inspirado nas ideias de (Carvalho 2015). A seguir explicaremos como os eventos de *flash crowd* são identificados e como a estratégia empregada foi adaptada para detectar uma utilização acima da capacidade do nó da nuvem sob monitoramento.

Segunda a teoria da informação, entropia é uma medida de incerteza de uma variável aleatória, geralmente referenciada como entropia de Shannon, e correlação total é a medida que quantifica a redundância ou a dependência entre um conjunto de n variáveis aleatórias. Em um evento de Flash Crowd, o número de acessos para um

determinado conjunto de conteúdos aumenta significativamente, enquanto que para os demais conteúdos o número de acessos se mantém no mesmo patamar. Dessa forma, pode-se utilizar os conceitos de entropia, entropia conjunta e correlação total para analisar os acessos a conteúdos em um instante de tempo atual e um instante de tempo anterior, de forma a detectar o evento de Flash Crowd. Entretanto, para obter os valores das entropias e correlação total, primeiro é necessário calcular as probabilidades de acesso dos conteúdos, também chamados de objetos.

Por simplificação consideramos o tempo como uma variável discreta representado por um número inteiro não negativo. Para a avaliação da ocorrência de um evento de *flash crowd*, tomamos duas variáveis aleatórias X e Y que representam a probabilidade de um determinado conteúd ser acessado em um tempo *t'* e Y de um conteúdo ser acessado em um tempo t, separados por um intervalo de tempo w, onde *t' = t – w* (os conteúdos no contexto dos experimentos científicos podem ser vistos como os dados armazenados em bancos de proveniência em diversos nós computacionais). À medida que o evento de *flash crowd* começa, a correlação entre X e Y aumenta. Sendo assim, a primeira análise a ser feita é verificar se a correlação destas variáveis está caminhando para 1. Esta análise nos dará um indicativo que as duas variáveis em análise estão em um caminho crescente e em conjunto. Pela simples observação de todos os acessos nos tempos *t'* e *t*, as seguintes funções de massa de probabilidade (f e g) e suas correspondentes distribuições acumuladas (F e G) podem ser facilmente estimadas:

$$f(x) = \Pr(X = x)\,;\; F(x) = \Pr(X \leq x)\,; \tag{1}$$

$$g(y) = \Pr(Y = y)\,;\; G(y) = \Pr(Y \leq y)\,; \tag{2}$$

Dependendo do perfil de utilização das máquinas, as variáveis *X* e *Y* serão mais ou menos correlacionadas entre si. Uma maneira de calcular esta correlação é recorrendo ao coeficiente de correlação de Pearson, indicada a seguir por ρ. O coeficiente de correlação de Pearson entre duas variáveis é definido como a covariança de duas variáveis dividida pelo produto de seus desvios padrão. Quando aplicado a uma amostra, o coeficiente de correlação de Pearson, é geralmente representado pela letra ρ e pode ser referenciado como coeficiente de correlação de amostra. A fórmula para ρ pode ser obtida substituindo estimativas da covariância e variância baseadas em uma amostra. A fórmula para ρ é representada por:

$$\rho = \frac{\sum_{x\geq 0}(c\prime_x - a\prime)(c_x - a)}{\sqrt{\sum_{x\geq 0}(c\prime_x - a\prime)^2}\sqrt{\sum_{x\geq 0}(c_x - a)^2}} \quad (3)$$

Tabela 1 Descrição da notação utilizada

Variável	*Descrição*
t	Tempo atual
W	Intervalo de tempo da análise
t'	Tempo seguinte após instante t
X	Máquina acessada no instante t’
Y	Máquina acessada no instante t
$f(x)$	Probalidade de acessar uma máq x
$g(y)$	Probalidade de acessar uma máq y
$F(x)$	Probilidade de se acessar uma máq dentre as quais a máq x tenha o maior índice
$G(y)$	Probilidade de se acessar uma máq dentre as quais a máq y tenha o maior índice
ρ	Coeficiente de correlação de Pearson
c'_x	Número de acessos à máquina X no tempo t’
c_x	Número de acessos à máquina X no tempo t
a'	Média dos acessos à máquina X no tempo t’
a	Média dos acessos à máquina X no tempo t

Uma vez definida a função de correlação, utilizaremos a função de entropia para descobrir o início de um evento de *flash crowd*, pois quando este começa, a entropia marginal dada por H(X) e H(Y) diminui substancialmente.

$$H(X) = -\sum_{x\leq 0}\sqrt{f(x)\,log_2 f(x)} \quad (4)$$

$$H(Y) = -\sum_{y\leq 0}\sqrt{f(y)\,log_2 f(y)} \quad (5)$$

Em nossas análises levamos em consideração as entropias marginas H(X) e H(Y). Se ambas apresentam uma grande variação de comportamento, existe um

indicativo do início ou fim de um evento de *flash crowd*. Quanto maior for a redução da entropia, mais forte esta é considerada. Quando a entropia volta a valores mais altos, será o término da flash crowd.

Esta estratégia gera menos falsos positivos, mesmo em cenários de grandes flutuações de uso de CPU.

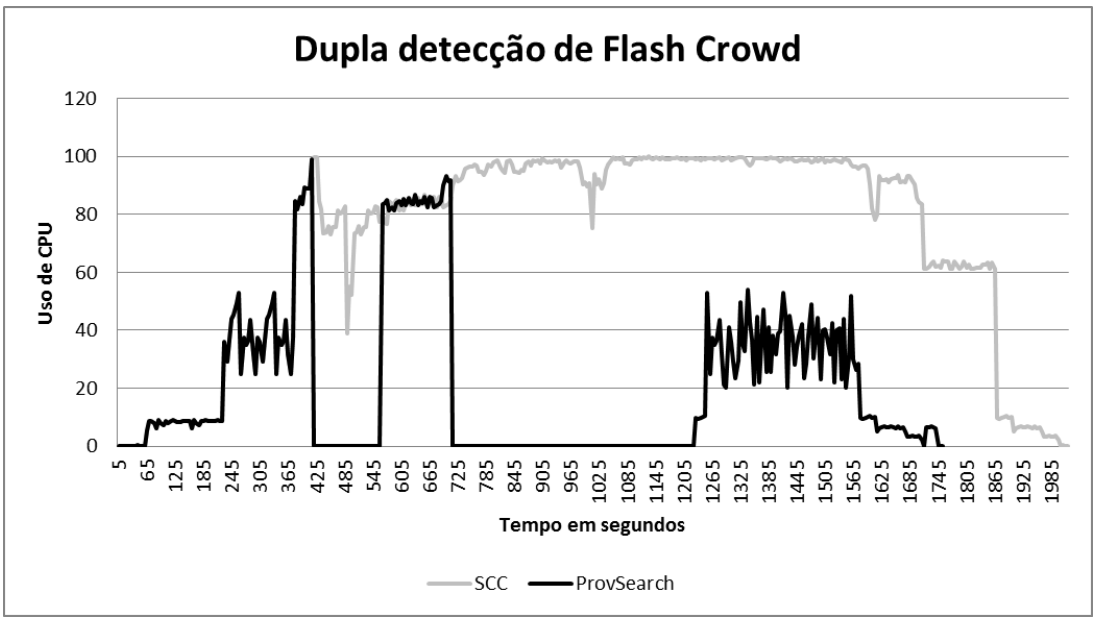


Figura 6 Exemplo de Flash Crowd

Na figura Figura 6 podemos verificar a detecção de dois eventos de *flash crowd* ao logo do experimento. O primeiro ocorre no tempo 365 e o segundo ocorre no tempo 725. Em ambos os casos, após a detecção, o serviço responsável pelo gerência dos dados de proveniência modificou a configuração do ambiente para que não houvesse queda de desempenho. Após o tratamento do primeiro evento de *flash crowd*, o experimento volta a ser executado no tempo 545. O gráfico teve uma tendência semelhante ao do experimento que foi executado sem nenhum tipo de estratégia. Com isso concluímos que a adaptação realizada na infraestrutura não foi suficiente para atender a demanda em questão. No entando no tempo 725 essa questão é notada, pois um novo evento de *flash crowd* é notado. Após essa segunda intervenção o nível de utilização de CPU passa a seguir abaixo do patamar de 60%. E o experimento finaliza 4 minutos antes no tempo 1745.

4.3 Modelo de Custo

No esquema apresentado na seção 4.1, o nó de controle é responsável por identificar qual será o nó de banco de dados que irá armazenar os dados de proveniência para uma execução específica. Neste momento ProvSearch se prepara para gerenciar a proveniência do experimento. Primeiramente deverá obter, por meio da análise histórica dos dados de proveniência, alguns parâmetros que servirão para determinar o nível de complexidade do experimento que será executado. Neste contexto, complexidade está sendo usada como um indicativo da maior ou menor necessidade de capacidade de processamento, memória e espaço de armazenamento. Quanto mais execuções do experimento tiverem ocorrido, melhor será a precisão da análise. No momento de apontar qual nó será responsável por gerenciar o uso dos dados de proveniência, alguns fatores deverão ser levados em consideração:

- Espaço de armazenamento total e disponível do nó;
- Memória total do nó e percentual médio sendo utilizado;
- Capacidade de processamento do nó e percentual médio sendo utilizado;
- Localização geográfica do nó;

Além das informações do experimento e da lista de nós disponíveis (e suas capacidades computacionais e localização), para que seja possível determinar o melhor nó a ser utilizado, ProvSearch tem acesso a uma tabela que detalha o valor gasto na utilização dos diferentes tipos de máquinas na nuvem. Por fim, o último conjunto de parâmetros faz referência ao grau de urgência do cientista pelo resultado do experimento e quanto este pode investir em processamento, uma vez que executará seu experimento em um ambiente com encargos financeiros. Quanto mais verba disponível, mais poder de processamento pode ser obtido, reduzindo, em geral, o tempo total da execução. Dimensionar da melhor maneira possível o conjunto de máquinas que serão utilizadas para executar o *workflow* e para gerenciar o uso da proveniência é uma prioridade, pois um mau dimensionamento de recursos da nuvem pode impactar o desempenho da execução do *workflow* ou aumentar (sem necessidade) o custo da execução. Este dimensionamento é muito complexo uma vez que as possibilidades de uso de diferentes máquinas são enormes. Em nosso trabalho, quando uma complexidade é endereçada, o algoritmo deve decidir o que é melhor: utilizar uma máquina (ou conjunto de máquinas) já existente, ou criar uma nova infraestrutura. Essa foi a

motivação para o desenvolvimento de um modelo de custo para o ProvSearch que fosse capaz de avaliar a complexidade envolvida no gerência da proveniência do experimento a ser executado, baseando-se na análise histórica da proveniência de outros experimentos semelhantes, e preparasse uma infraestrutura adequada. ProvSearch, tendo uma visão geral da infraestrutura existente, das possibilidades de criação de novas máquinas e recebendo como entrada a necessidade do cientista em relação a desempenho e custo, é capaz de apontar qual a melhor (ou próxima da melhor) infraestrutura a ser utilizada. Quando estudamos computação em nuvem, vários aspectos devem ser levados em consideração, como segurança, disponibilidade, índice de falha, dentre outros. Todos esses pontos são de fato muito importantes, mas no contexto desta tese vamos focar nossos esforços na infraestrutura. Neste modelo os usuários, que precisam executar experimentos em larga escala, demandam recursos (ou seja, máquinas virtuais) por um período de tempo e pagam apenas pelo que usam. No entanto, a determinação do montante de máquinas virtuais necessárias para uma execução de um *workflow* específico está longe de ser trivial, pois é difícil para os usuários estimarem o tempo de execução de atividades, a quantidade de arquivos de dados produzidos e os tempos de transferência associados.

O cenário de uso de máquinas virtuais apresentado anteriormente pode ser descrito por meio de uma formalização baseada na estratégia apresentada por Coutinho *et al.* (2014). Lembrando que nosso objetivo nesta etapa é determinar uma configuração da infraestrutura necessária para trabalhar com proveniência e precisamos de um método que seja capaz de alcançar tal objetivo. Antes de apresentar a estratégia desenvolvida, vamos abordar a notação usada para caracterizar o problema.

Seja P o conjunto de tipos de máquinas virtuais oferecido por um provedor de nuvem durante um conjunto de períodos de tempo. Consideremos ainda um conjunto de necessidades e restrições dos cientistas: C_M (Custo Financeiro Máximo); T_M (Tempo Máximo de Execução); D_s (Espaço em Disco para Armazenamento); M_c (Média de Memória) e G_f Média de Processamento. Temos ainda que cada tipo de máquina virtual $p \in P$ tem um C_p (custo financeiro) associado (ou seja, o custo de reservar a máquina virtual para um período de tempo, sendo os períodos de tempo ajustados de acordo com o provedor de nuvem, que pode ser 1h ou 1min) e recursos de computação como armazenamento em disco d_p, capacidade de memória m_p e um poder de processamento de g_p por período de tempo. Devemos ainda destacar a questão da localidade. Quanto

mais longe do experimento a máquina se localizar, maior será o gasto, pois esta máquina ficará mais tempo ativa por conta do tráfego de dados entre o experimento e a base de dados de proveniência. Por conta disso, criamos uma variável β_p que aumenta o custo do experimento de acordo com a distância entre o nó que está sendo avaliado e o local onde o experimento está ocorrendo. Por fim temos o N_M, que representa o número máximo de máquinas que posso alocar, o que é algo comum de existir em um servidor de nuvens.

Sendo assim, definimos X_{pit} como uma variável binária para cada $p \in P$, $t \in \mathrm{T} = \{1, \ldots, \mathrm{TM}\}, e\ i \in \{1, \ldots, N_M\}$, de tal modo que X_{pit} = 1, se e somente se a máquina virtual i do tipo p é utilizada no momento t. Caso contrário, X_{pit} = 0. Além disso, considere t_m como a variável que representa a última vez que uma máquina virtual foi atribuída pelo utilizador. O cenário descrito pode ser formulado da seguinte forma:

Primeiramente precisamos definir uma função objetivo (1) que busca minimizar o custo e o tempo (nesta fórmula os parâmetros sofreram uma normalização para que pudessem ser comparados em conjunto). No entanto, como esses dois objetivos são conflitantes, o cientista irá balancear a fórmula por meio de duas constantes, α_1 e α_2, que ponderam a importância do tempo de execução e do custo do experimento, respectivamente. Ou seja, o usuário pode decidir entre uma execução mais barata, com $\alpha_1$mais próximo de 1, ou uma execução mais rápida, com α_2 tendendo a 1.

$$(1) \qquad min\left(\alpha_1 \sum_{p \in P} \sum_{i=1}^{N_M} \sum_{t \in T} \beta_p C_p X_{pit} + \alpha_2 t_m\right)$$

Para chegar a uma configuração que atenda às restrições imposta pela análise histórica da utilização de recursos, submetemos a função objetivo a uma série de restrições, conforme abaixo:

A primeira restrição é o custo máximo permitido para o experimento (2). Ou seja, não podemos estourar esse orçamento na fase de concepção e adaptação de nossa infraestrutura.

$$(2) \qquad \sum_{p \in P} \sum_{i=1}^{N_M} \sum_{t \in T} \beta_p C_p X_{pit} \leq C_M$$

Da mesma maneira precisamos atender à demanda por espaço em disco, memória para processamento e capacidade de processador requerida pela gerência de proveniência do experimento (3), (4) e (5). Ou seja, não podemos falhar na execução de um experimento por falta de espaço para armazenar sua proveniência

(3) $$\sum_{p \in P} \sum_{i=1}^{N_M} d_p X_{pit} \geq D_s X_{p'i't} \quad , \quad \forall t \in T, \forall p' \in P, \forall i' \in \{1, \dots, N_M\}$$

(4) $$\sum_{p \in P} \sum_{i=1}^{N_M} m_p X_{pit} \geq M_c X_{p'i't} \quad , \quad \forall t \in T, \forall p' \in P, \forall i' \in \{1, \dots, N_M\}$$

(5) $$\sum_{p \in P} \sum_{i=1}^{N_M} \sum_{t \in T} g_p X_{pit} \geq G_f$$

Não podemos esquecer-nos de criar uma restrição para o número máximo de máquinas (6) que podemos instanciar no provedor dos recursos.

(6) $$\sum_{p \in P} \sum_{i=1}^{N_M} X_{pit} \leq N_M, \forall t \in T$$

Por fim formalizamos ainda que X_{pit}é uma variável binária (7) e que o somatório de $\alpha_1 + \alpha_2$ é 1 (8).

(7) $X_{pit} \in \{0,1\}$

(8) $(\alpha_1 + \alpha_2) = 1$

Por simplificação iremos considerar que cada nó é uma máquina única com poder de processamento equivalente ao somatório de cada uma das máquinas individuais que compõem aquele nó.

Com base nesse conjunto de regras criamos um algoritmo capaz de preparar a configuração da infraestrutura inicial necessária para a gerência da proveniência do experimento. Criamos ainda um serviço que verifica toda infraestrutura interligada para mapear o grau de utilização de cada um dos nós existentes. Este serviço atualiza no banco de dados de cada nó um catálogo onde consta a capacidade computacional de cada um dos nós existentes, sendo eles: Memória, Disco e Capacidade Computacional, e seu grau de utilização. Este serviço é fundamental para o bom funcionamento do algoritmo mencionado.

Como estratégia inicial de solução, estamos utilizando um método de força bruta, que varre todas as possibilidades em busca da alternativa que minimize a função de custo. O tempo de processamento da solução não impacta a execução do *workflow* e para os casos estudados não representou mais que 1% do tempo total. Começamos com as máquinas mais baratas e vamos tentando utilizar máquinas mais caras, sem estourar o limite do orçamento ou até atender a restrição de tempo máximo.

Não podemos confundir a definição da infraestrutura inicial para o experimento com o monitoramento baseado em *flash crowd.* A elaboração da estrutura inicial é um mecanismo independente do monitoramento do uso do nó para redimensioná-lo em caso de utilização acima de sua capacidade. O mecanismo de *flash crowd* descrito só funciona quando o nó já está em uso e leva em conta apenas o uso de CPU. Para definir a infraestrutura inicial a ser utilizada, levamos em conta um conjunto maior de parâmetros. Nesse ponto da aplicação existem dois serviços de monitoramento, um que fica verificando o uso de todos os nós e catalogando isso em uma tabela, de modo que seja possível a qualquer momento ter uma ideia do cenário global de todos os nós acessíveis por ProvSearch. E existe o segundo serviço de monitoramento que fica restrito a um nó específico e sinaliza a necessidade de uma ação de redimensionamento pontual.

4.4 Modelagem do *Data Warehouse* de Proveniência

Além da já mencionada complexidade de se executar os experimentos modelados como *workflows*, existe ainda a complexidade de trabalhar de forma cooperativa em um ambiente distribuído. Uma das razões é o usuário nem sempre ter acesso aos dados de proveniência gerados por outro cientista, que já pode ter executado o mesmo experimento que ele. Não só isso, o usuário pode não saber onde acessar tal informação (Costa *et al.* 2014). Por isso, há uma necessidade de se repensar o modo de armazenamento dos dados de proveniência, garantindo que os usuários possam, não só ter um acesso facilitado aos dados de proveniência gerados por outros usuários, mas também saber que estes dados existem e podem ser acessados de forma otimizada.

Um dos nossos objetivos na tese é avaliar a utilização de *data warehouses* de dados de proveniência para otimizar a execução de consultas que possam tirar vantagem deste tipo de abordagem, como aquelas que fazem uso de agregação, por exemplo. No primeiro momento, nos preocupamos em atestar se as consultas usadas pelos cientistas

se beneficiariam do uso do *data warehouse*. Então foi preciso desenvolver o ambiente adequado para elaboração de testes. Para tal criamos o *data warehouse* centralizado e outro distribuído. Preocupamo-nos ainda com eventuais quedas de desempenho decorrentes dessa estratégia. Neste contexto, mostramos por meio da análise do desempenho de consultas representativas do cenário real que, em grande parte dos casos, o uso do *data warehouse* acelera a execução de consultas que são rodadas pelos SGWfCs, fazendo assim com que a execução como um todo tenha seu tempo reduzido. O *data warehouse* é uma das parte integrantes do ProvSearch.

4.4.1 Modelo Dimensional

O primeiro passo para se construir um *data warehouse* é a concepção de como ele organizará os dados armazenados, isto é, como será seu modelo de dados. Diferentemente da maioria dos bancos de dados transacionais, que em geral utilizam um modelo normalizado, um *data warehouse* utiliza uma modelagem desnormalizada. Este modelo, concebido nos anos 70, foi estruturado de forma dimensional para atender a necessidade humana de simplicidade (Kimball e Margy Ross 2002).

A ideia principal por trás desse modelo é tornar simples as operações realizadas sob o *data warehouse*, como, por exemplo:

Agregação (*Rollup)* onde temos a combinação de dados de uma ou mais dimensões, objetivando deixar uma informação pronta para ser visualizada. *Nos workflows* podemos ter o tempo médio de execução calculado das execuções de uma determinada atividade, verificando dados históricos dos últimos sete dias ou dos últimos anos. Essa informação será consumida de acordo com a necessidade do problema. Outra operação importante oferecida pelo *data warehouse* é o reverso da agregação (*Drill-Down*). A partir de uma informação consolidada, podemos rever os valores que foram responsáveis por definir o resultado em análise. Quando pensamos no tempo total de execução de um *workflow*, podemos estar interessados em descobrir qual atividade ou atividades formaram os maiores gargalos. Por meio desse tipo de operação, tal análise fica simplificada. Destacamos ainda as operações de fatiar (*Slice-Dice)* que também pode ser útil nas análises de proveniência, pois a partir de uma informação fechada de tempo de execução ou quantidade de falhas, podemos obter subconjuntos do todo e formar novos resultados (Kimball e Margy Ross 2002). Em um *data warehouse* estas operações são fundamentais devido à quantidade de dados armazenados. Quando

trabalhamos com bases de dados pequenas, podemos obter qualquer um desses resultados por meio de consultas de rápida execução. No entanto, quando as bases de dados aumentam de tamanho, obter informações como as descritas acima pode se tornar muito custoso. Vale frisar ainda que uma consulta relativamente simples, como a que calcula o tempo médio de execução de uma determinada atividade, pode ser executada milhares de vezes em um SGWfC. Se o gerenciador busca a informação já consolidada no *data warehouse*, no lugar de toda vez tentar calculá-la a partir da base transacional, haverá intuitivamente um ganho de desempenho. Desta observação que surgiu a ideia de adicionar um *data warehouse* ao nosso conjunto de serviços. O que utilizaremos dos conceitos de *data warehouse* serão os conceitos mais básicos, pois os conceitos básicos unidos às demais estratégias do ProvSearch trarão grandes reduções de tempo de processamento e custo de execução dos experimentos. Em seguida falaremos um pouco sobre as estratégias apresentadas para criação de nossa base de *data warehouse*.

Na modelagem dimensional, os dados são organizados em dois tipos de tabela: Tabelas de fatos e tabelas de dimensões.

As tabelas de fatos são aquelas que armazenam os valores detalhados de medidas ou fatos. Por exemplo, uma tabela que armazene quantidade e valor total da venda de um produto em um mês.

Fatos podem ser numéricos ou não, além de, se forem numéricos, poderem ser aditivos, semiaditivos ou não aditivos. Fatos aditivos são aqueles que podem ser somados em qualquer condição, sendo capazes de condensar informações com grande facilidade, tal qual a quantidade de Reais por venda. Fatos semiaditivos são aqueles que podem ser somados em determinadas condições. E, finalmente, fatos não aditivos são aqueles que não podem ser somados em condição nenhuma. É importante ressaltar que deve se priorizar fatos aditivos, pois raramente o usuário pede por somente uma linha da tabela de fatos, mas, sim, um conjunto delas.

Além disso, fatos são interseccionados com dimensões, tais como tempo e data, que definem a granularidade do fato. Quanto maior a quantidade de dimensões interseccionadas com uma tabela de fatos, maior será o nível de detalhamento com o qual o fato é armazenado na tabela, isto é, o fato possui uma maior granularidade.

Apesar de ser possível ter fatos textuais, é importante ressaltar que isto deve ser evitado ao máximo, pois fatos textuais são mais difíceis de analisar, pois tabelas de fato

normalmente contêm milhões de linhas. É importante notar também que a maioria dos casos onde encontramos fatos textuais, em geral, estes podem ser retirados da tabela de fatos e serem colocados em uma tabela de dimensões. Na figura Figura 7 temos um exemplo de uma tabela de fatos no contexto desta tese.

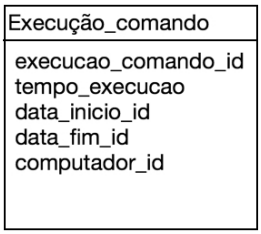


Figura 7. Exemplo de tabela de fatos.

A tabela de dimensões é a tabela que contém a descrição textual do "negócio". Normalmente, contém muitas colunas ou atributos, que descrevem a linha da tabela de dimensões. Diferentemente das tabelas de fatos, as tabelas de dimensões contêm poucas linhas, apesar de cada linha conter muita informação.

Tabelas de dimensões bem construídas são a chave para uma boa usabilidade de um *data warehouse*. Ao se construir uma tabela de dimensões, é importante que todos os atributos tenham nomes que os descrevam bem, pois é por meio destes atributos que a maioria das buscas com filtro serão feitas, tais como datas, computadores, empresas, produtos. Então, é importante que o usuário consiga saber facilmente qual atributo ele vai utilizar como restrição para obter os resultados desejados em uma consulta. É importante, também, evitar o uso de abreviações ao escolher o nome para um atributo, pois nem todos os usuários serão especialistas no assunto. Por fim, quanto mais se trabalha para se encontrar nomes adequados para os atributos de uma tabela de dimensões, mais legível fica o *data warehouse*, acarretando em melhor usabilidade do mesmo.

Os valores que cada atributo pode assumir pode ser tanto numérico, quanto textual. No caso dos valores numéricos, é importante analisar corretamente se este valor numérico é um fato ou uma característica de uma dimensão do *data warehouse*, isto é crucial para melhorar sua usabilidade. Por fim, na Figura 8, mostramos um exemplo de tabela de dimensões.

Figura 8. Exemplo de tabela de dimensões de tempo.

4.4.2 Metodologia

Há variadas metodologias utilizadas para projetar um *data warehouse* com vasta literatura na área como (Kimball e Margy Ross 2002), (Vaisman e Zimányi 2014), dentre muitas outras. Nossa aplicação de banco de dados de *data warehouse* é bastante simples, mas em conjunto com as demais estratégias traz grandes ganhos. Por conta disso fomos buscar as melhores práticas pra criação de nosso *data warehouse* na litetura clássica da áreaa.

Em nosso caso, nós já tínhamos o conjunto de consultas mais utilizadas nos experimentos e tabém fora deles, para análises pós execução, então partimos para uma clássica estratégia conhecida como *bottom-up* apresentada por (Kimball e Margy Ross 2002). Esta metodologia é interessante, pois o desenvolvimento do *data warehouse* é feito de forma incremental, evitando quaisquer problemas relacionados a necessidades novas que surjam no caminho e utilizando como base o conjunto de consultas pré-existentes.

Primeiramente, é necessário entender como é organizado um *data warehouse*. Um *data warehouse*, usualmente, é composto por Data Marts (DM). Os DMs são subconjuntos de dados do *data warehouse*. Cada DM contém um subconjunto de dados direcionado a um tipo de usuário final. Isto é vantajoso, pois permite o acesso descentralizado ao *data warehouse*, além de conseguir garantir um melhor atendimento à demanda do usuário final.

A metodologia *bottom-up* procura explorar as vantagens do uso de DMs para construir um *data warehouse*. Esta metodologia permite a construção incremental do *data warehouse* por meio de DMs independentes. Além disso, isto permite um desenvolvimento mais acelerado do *data warehouse*, pois é possível o desenvolvimento de vários DMs ao mesmo tempo. Também é importante destacar que facilita o enfoque

na construção de um *data warehouse* que atenda às necessidades principais do problema em questão.

No entanto, apesar da facilidade de desenvolvimento do *data warehouse* por meio desta metodologia, ela facilita a criação de um *data warehouse* com redundância de dados. Por isso, é importante atentar para a conformidade das tabelas de dimensões dos DMs, a qual garante que não haja redundância de dados à medida que se constrói novos DMs. Na prática, a conformidade das tabelas de dimensões se traduz na utilização da mesma tabela de dimensões para datas em diferentes DMs, o que evita a redundância de dados.

Para garantir esta conformidade das tabelas de dimensões entre DMs foi criada uma estratégia chamada *Data Warehouse Bus Matrix*. Esta ferramenta é uma tabela na qual as linhas são os nomes dos DMs e as colunas são as tabelas de dimensões existentes no *data warehouse*. Nesta tabela, marca-se quais tabelas de dimensões cada DM contém, o que é importante para verificar quais tabelas podem ser utilizadas por mais de um DM e quantas tabelas de dimensões cada DM tem.

4.4.3 Implementação

Neste capítulo serão apresentadas e descritas todas as informações relevantes sobre o processo da criação do *data warehouse*. Objetivamos ainda avaliar o desempenho e vantagens do uso desta abordagem no problema do uso da proveniência distribuída. Rodamos os primeiros experimentos para validar as estratégias adotadas e acreditamos ter encontrado um caminho promissor em relação à metodologia de construção do nosso *data warehouse*. A seguir detalhamos as questões envolvidas nessa etapa do projeto.

4.4.3.1 **Descrição dos esquemas**

No capítulo 3, foi descrito o esquema usado para armazenar os dados de proveniência do SciCumulus. Nesta seção, será descrita a criação do esquema do *data warehouse* a ser utilizado nos experimentos. Já explicamos como se constrói um *data warehouse* utilizando-se uma metodologia *bottom-up* e modelo dimensional. Para o nosso DW, foram desenvolvidos 3 DMs: *Activation, Workflow, Activity*. Para o desenvolvimento dos DMs, foram levadas em conta as consultas mais utilizadas representativas do conjunto de consultas utilizadas pelos SGWfC e pelos cientistas em suas análises pós

processamento. Vale ressaltar que nem todas são consultas típicas de DW, mas a modelagem proposta é capaz de obter resposta para todas as consultas que pesquisamos. A seguir apresentaremos a modelagem, deixando para apresentar as consultas quando abordarmos as análises comparativas de desempenho das diferentes estratégias que testamos.

No primeiro DM são armazenadas as informações relativas à execução de ativações. Ele contém cinco tabelas: *Activation Executions*, *Date*, *Workflow*, *Activity*, *Time* e *Computer*. A Figura 9 mostra como o DM foi modelado.

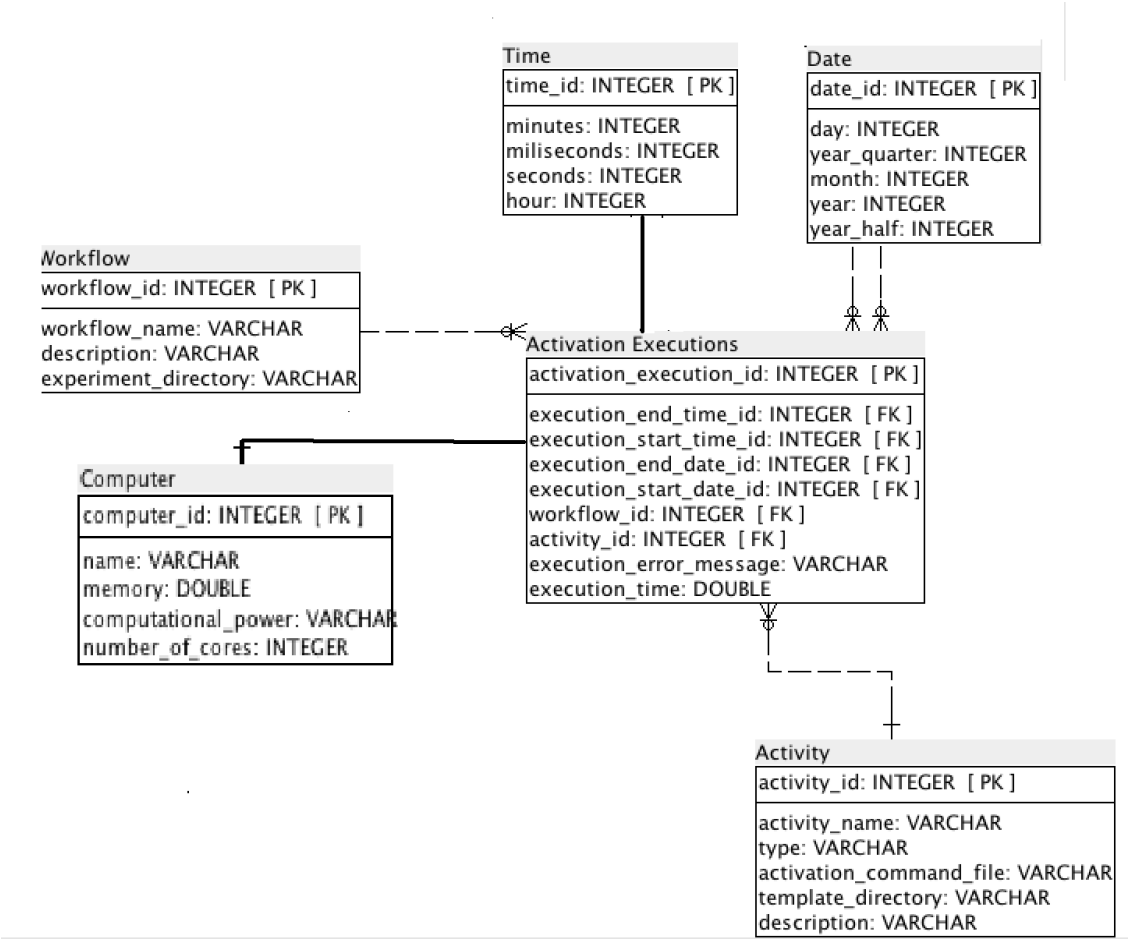


Figura 9. Modelo de Dados para o DM *Activation*

A tabela de *Activation Executions* é uma tabela fato e guarda informações de todas as atividades que foram executadas, sendo que uma ativação pode ser composta de mais de uma atividade. Dentro de um ciclo de execução de um *workflow*, uma ativação pode ser executada milhares de vezes, variando-se os parâmetros de entrada. Por conta disso, se quisermos saber a característica de uma ativação em relação a tempo de execução, por exemplo, devemos considerar uma média de todas estas execuções.

Este tipo de informação é muito importante no planejamento dos recursos para uma nova execução ou para a realização do redimensionamento dos recursos durante a execução e por isso a importância de um DM específico para tratar das ativações. Relacionados a este fato temos dimensões de tempo, através das quais podemos verificar esses agrupamentos em um período específico, de um mês ou um ano, por exemplo. Por essa modelagem, podemos utilizar a dimensão *Activity* para avaliar o comportamento das ativações, mesclando essa filtragem com um período de tempo, por exemplo.

No segundo DM são armazenadas as informações relativas à execução de *workflows*. Ele contém quatro tabelas: *Workflow Executions, Workflow*, *Date* e *Time*. A Figura 10 mostra como foi modelado o DM.

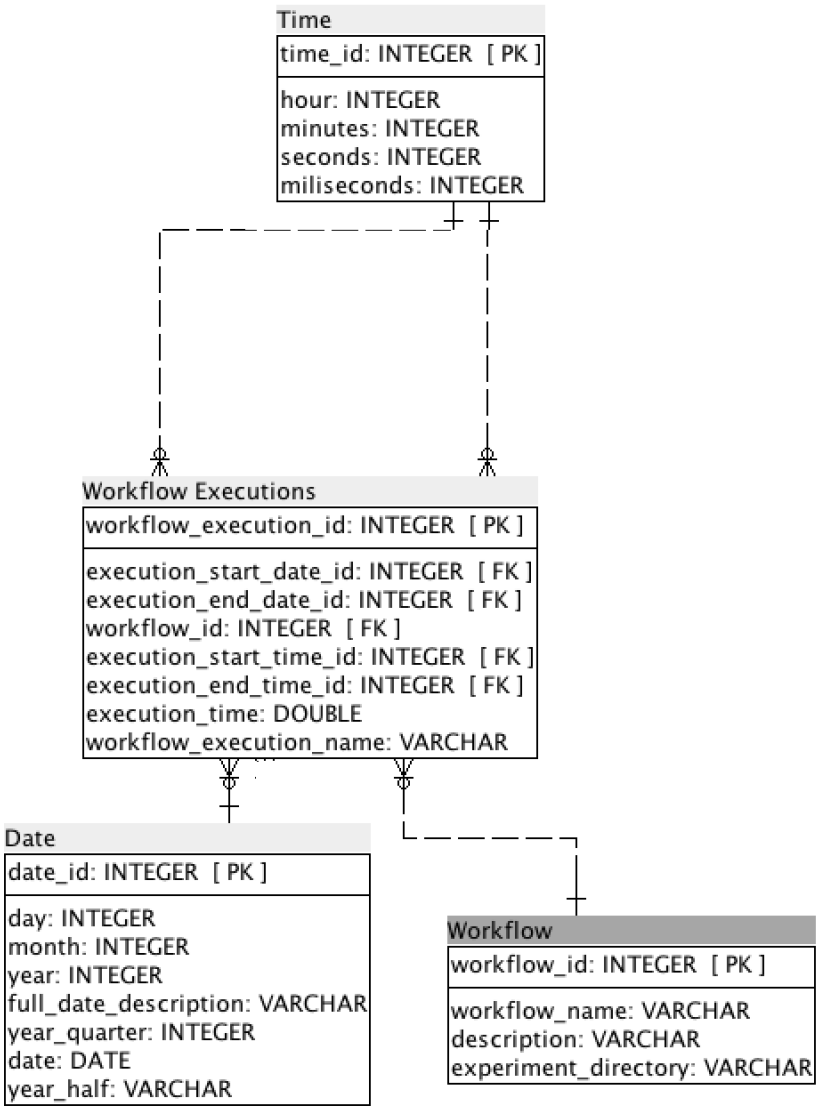


Figura 10. Modelo de Dados para o DM *Workflow*

Da mesma maneira que *Activity Executions*, *Workflow Executions* representa um *Workflow* (conceitual) sendo executado. As observações feitas anteriormente se aplicam aqui também. Ou seja, podemos avaliar o comportamento de um determinado *workflow* pela análise das médias de suas execuções por períodos diferenciados de tempo. Este tipo de análise é trivial em um ambiente de *data warehouse* e pode ser processada de forma muito rápida. Se tivermos que executar esse mesmo tipo de análise olhando para nossa base transacional, teríamos que rodar uma consulta mais pesada, que seria mais ou menos lenta de acordo com o volume de informação existente na base em questão.

No terceiro, e último, DM são armazenadas as informações relativas à execução de atividades. Ele contém cinco tabelas: *Activity Executions*, *Activity*, *Workflow*, *Time* e *Date*. A Figura 11 mostra como foi modelado o DM. Neste DM o foco está na análise do comportamento da atividade, independente da ativação da qual essa pode ter feito parte.

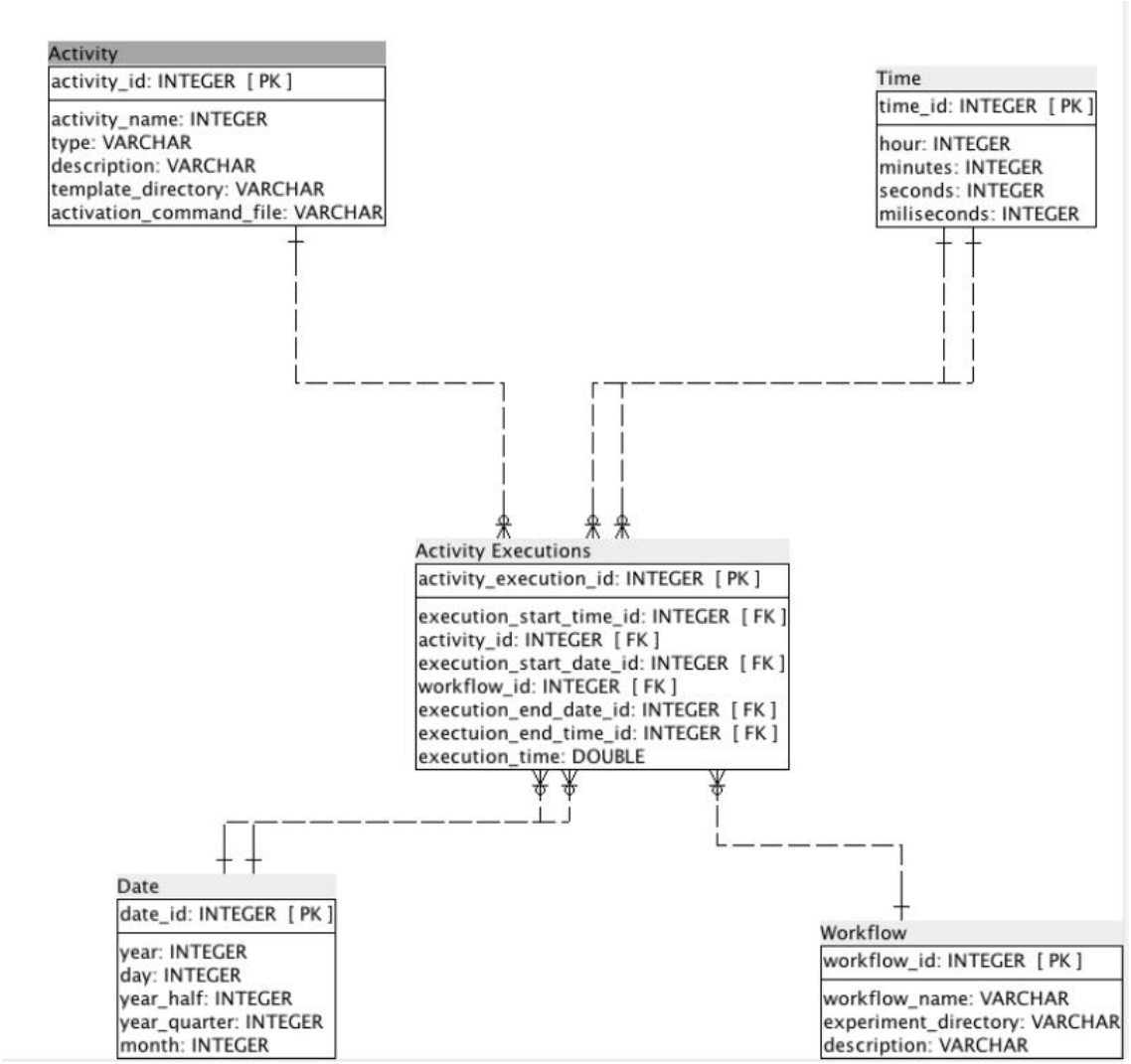


Figura 11. Modelo de Dados para o DM *Activity*.

Estes 3 DMs foram desenvolvidos de forma incremental e compõem a primeira versão do *data warehouse*. Neste *data warehouse*, as tabelas *Activity Executions*, *Workflow Executions* e *Activation Executions* são as tabelas de fatos e as tabelas *Time*, *Date*, *Workflow* e *Computer* são as tabelas de dimensões. Estes DMs trarão benefícios direto para as consultas que são executadas pelos SGWfCs no processo de escalonamento de recursos. Dessa forma poderemos acelerar a execução do *workflow* como um todo. Trarão ainda vantagens para análises pós processamento.

4.4.3.2 **Implementação Física**

Nesta seção serão descritos quais foram os passos para a implementação do *data warehouse* distribuído na nuvem. Quando falamos em bancos de dados distribuídos, é importante entender como essa distribuição é dada. Em geral, um banco de dados

distribuído se dá por um conjunto de fragmentos 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 enconsukttão retornar os resultados e formar o resultado final. Além disso, quando se trata de um volume de dados muito grande, sua distribuição pode tornar as consultas mais rápidas, pois possivelmente serão executadas em partes menores, que constituem um menor volume de dados a ser acessado. Existem duas estratégias fundamentais de fragmentação: 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 completude, a reconstrutibilidade e a disjunção dos fragmentos.

A completude garante que se uma relação R é decomposta entre os fragmentos FR= {R1, R2, R3, ..., Rn}, então cada item encontrado em R, pode também ser encontrado em um ou mais Ri's.

A reconstrutibilidade garante que se uma relação R é decomposta entre os fragmentos FR= {R1, R2, R3, ..., Rn}, deve ser possível definir um operador relacional que consiga reconstruir R a partir de suas partes Ri, para todo Ri em FR.

A disjunção garante que se uma relação R é decomposta horizontalmente entre os fragmentos FR= {R1, R2, R3, ..., Rn}, então qualquer item di encontrado em Rj, não pode ser encontrado em nenhum outro fragmento Rk, para k ≠ . 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 reconstrutibilidade, 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. Baseamos nossa distribuição do *data warehouse* nos conceitos acima descritos.

Inicialmente, o *data warehouse* foi implementado no PostgreSQL ("PostgreSQL" 2014), que está hospedado em um servidor da Amazon (Amazon EC2 2014). A implementação, neste caso, não foi distribuída, mas somente em um servidor.

Após a implementação inicial no servidor da Amazon, prosseguimos para a implementação de um *data warehouse* distribuído. Houve a tentativa de fazer esta implementação em várias soluções de SGBDs paralelos em busca de uma solução que atendesse aos requisitos. Isto se deu, pois o ambiente da nuvem torna a implementação de um banco de dados distribuído mais complexa. As soluções de paralelismo existentes nem sempre são adaptáveis para serem usadas na nuvem, o que motivou a procura por diferentes soluções, como irá ser demonstrado abaixo.

A primeira alternativa foi procurar uma solução para paralelismo já oferecida pelo PostgreSQL. No entanto, até aquele momento, só era possível ter um banco de dados paralelo com o uso de software externo desenvolvido por si mesmo.

A segunda alternativa foi o plugin PgPool-II (pgpool 2014), que pode ser integrado ao PostgreSQL para a criação de bancos de dados paralelos, necessitando-se apenas ter os bancos de dados criados em cada servidor que será utilizado para a criação do banco de dados paralelo. No entanto, o *plugin* se mostrou incompatível com o PostgreSQL, pelo menos para nuvem, e incompleto, apresentando diversos problemas mesmo quando somente implementando o banco de dados mostrado no tutorial.

A terceira alternativa foi o software Postgres-XC (Postgres-XC 2014), que é uma versão do PostgreSQL com recursos para paralelismo. No entanto, ela só oferece o recurso da replicação de bancos de dados em diferentes servidores.

A quarta alternativa foi o software SQL Server (SQL Server 2014), que é um SGBD oferecido pela Microsoft. É um SGBD robusto, oferecendo bom desempenho para consultas em Bancos de Dados com enormes quantidades de dados. No entanto, este SGBD não oferece solução para bancos de dados paralelos na nuvem, mesmo utilizando a própria nuvem da Microsoft – Azure. O único paralelismo utilizado neste SGBD é no momento de fazer uma consulta ao banco de dados, utilizando-se dos núcleos do processador para paralelizar a execução desta consulta.

Por fim, foi utilizado o software NuoDB (NuoDB 2014), que oferece uma solução de paralelismo na nuvem. Este software é um SGBD que permite o uso de banco de dados paralelos em dois ou mais servidores. De forma muito trivial, foi possível configurar este banco para operar na nuvem. Todas as operações a serem feitas no NuoDB podem ser feitas por uma interface *Web*. Foi necessário implementar o *data*

warehouse novamente no NuoDB devido ao fato de não ser integrado ao PostgreSQL e por ter pequenas diferenças no momento da criação de tabelas.

4.4.3.3 **Descrição das Consultas**

As consultas escolhidas foram baseadas em consultas típicas da área de proveniência (Challenge 2006), outras apontadas pelos especialistas como sendo um conjunto representativo do universo de consultas utilizadas e também consultas utilizadas pelos SGWfC. Cada uma delas apresenta informações sobre os experimentos, que são sinalizados para as consultas por meio de parâmetros, gerando assim uma espécie de relatório sobre os mesmos. No conjunto abaixo, apresentamos consultas que buscam por diferentes tipos de informação e mostraremos que as mesmas podem ser executadas na base do *data warehouse* sem prejuízo de tempo. As consultas analisadas apresentam características diversificadas. Nosso intuito é atestar que o uso de uma base de dados diferente da relacional é viável nesse contexto e em geral traz boas vantagens.

4.4.3.3.1 Consulta 1

```
SELECT   a.tag, a.description
FROM     hactivity a, hworkflow w
WHERE    a.wkfid = w.wkfid
 and w.tag like '%Sciphy%'
```

Essa consulta lista o nome e a descrição das atividades de um determinado *workflow*. Como pode ser visto, o parâmetro *w.tag* é quem filtra o resultado por nome de *workflow*.

4.4.3.3.2 Consulta 2

```
SELECT   a.tag, t.taskid
FROM     hactivation t, hactivity a, hworkflow w
WHERE    w.wkfid = a.wkfid
and a.actid = t.actid
and w.tag ='SciPhy'
ORDER BY a.tag
```

Essa consulta lista as ativações de cada atividade de um *workflow* especificado pelo usuário. Como pode ser visto, o parâmetro *w.tag* é quem filtra o resultado por nome de *workflow*.

4.4.3.3.3 Consulta 3

```
SELECT   a.actid, a.tag, a.status,
            AVG(EXTRACT(EPOCH FROM(a.endtime-a.starttime)))
                                                AS MediaTempoExecucao

FROM      hactivity a, h workflow w
WHERE   a.wkfid=w.wkfid
and w.tag like '%SciPhylomics%'
and a.status<> 'FINISHED'
and a.status is not null
GROUP BY a.actid, a.tag, a.status
```

Essa consulta lista por quanto tempo as atividades que terminaram com algum tipo de erro ficaram em execução. Como pode ser visto, o parâmetro *w.tag* é quem filtra o resultado por nome de *workflow* e o parâmetro *a.status* é quem filtra o resultado por status de execução.

4.4.3.3.4 Consulta 4

```
SELECT   a.actid, a.tag, date_part('epoch',endtime - starttime )*1000 as duracao
FROM      hactivity a, h workflow w
WHERE   a.wkfid=w.wkfid and a.status= 'FINISHED'
and w.tag like '%SciEvol%'
```

Essa consulta lista a duração da execução das atividades que já tiveram sua execução completa de um determinado *workflow*. Como pode ser visto, o parâmetro *w.tag* é quem filtra o resultado por nome de *workflow* e o parâmetro *a.status* é quem filtra o resultado por status de execução.

4.4.3.3.5 Consulta 5

```
SELECT    a.actid, a.tag, starttime
FROM      hactivity a, hworkflow w
WHERE     a.wkfid=w.wkfid
and w.tag like '%Sciphy%' and a.status='RUNNING'
```

Essa consulta lista o instante de início de execução das atividades que não tiveram sua execução completa em um determinado *workflow*. Como pode ser visto, o parâmetro *w.tag* é quem filtra o resultado por nome de *workflow* e o parâmetro *a.status* é quem filtra o resultado por status de execução.

4.4.3.3.6 Consulta 6

```
SELECT    x.taskid, date_part('epoch',x.endtime - x.starttime )*1000 as duracao
FROM      hactivity a, hactivation x
WHERE     a.actid=x.actid
and a.tag='raxml'
and x.status='FINISHED'
```

Essa consulta lista a duração da execução das ativações que já tiveram sua execução completa em uma determinada atividade. Como pode ser visto, o parâmetro *a.tag* é quem filtra o resultado por nome de atividade e o parâmetro *x.status* é quem filtra o resultado por status de execução.

4.4.3.3.7 Consulta 7

```
SELECT   e.tag as nomeWorkflow,
         a.tag as nomeAtividade,
         a.status as statusAtividade,
        AVG(EXTRACT(EPOCH FROM(a.endtime-a.starttime)))
                                                AS MediaTempoExecucao
FROM eactivity a, eworkflow e
WHERE status='FINISHED'
and a.wkfid = e.ewkfid
GROUP BY  e.tag as nomeWorkflow,
          a.tag as nomeAtividade,
          a.status as statusAtividade,
```

Esta consulta calcula a média de tempo de execução das atividades que fazem parte de um determinado workflow.

4.4.3.3.8 Consulta 8

```
SELECT    w.tag, a.tag,  t.taskid, t.exitstatus, t.processor, t.workspace, t.status,
          t.endtime, t.starttime, r.file, r.bootstrap,
          extract ('epoch' from (t.endtime-t.starttime))||',' as duration
FROM      hworkflow w, hactivity a, hactivation t, hkeyspace k, relation r
WHERE     w.wkfid = a.wkfid
and  a.actid = t.actid
and t.taskid = k.taskid
and r.ik >= k.iik
and r.ik <= k.fik
and w.tag like '%SciPhy%'
and r.bootstrap <> 100
and duration > (select avg(extract ('epoch'
                  from (t.endtime-t.starttime))||',') -  2.4*stddev(extract('epoch'
                   from          (t.endtime-t.starttime))||',')
                   FROM hactivation ac WHERE ac.actid=t.actid)
```

Essa consulta verifica se a duração de uma determinada atividade está extrapolando o limite da média de tempo de execuções históricas desta mesma atividade, apresentando uma série de informações a respeito.

4.4.3.3.9 Consulta 9

```
SELECT  w.wkfid,w.tag,a.tag,t.exitstatus,t.processor,t.workspace,t.status, 3799,5809
t.endtime,t.starttime,
extract ('epoch' from (t.endtime-t.starttime))||',' as duration
FROM      hworkflow w, hactivity a, hactivation t
WHERE   w.wkfid = a.wkfid
and a.actid = t.actid
and not exists (select * from hactivation a2
where a2.actid = a.actid
and a2.exitstatus <> 0)
```

Essa consulta lista, por ordem crescente de execuções dos *workflow*s, as datas e horas de início e término, nomes dos *workflows* e suas descrições, bem como o nome de todas as atividades associadas a todas as execuções dos *workflow*s que foram executados e que não contenham nenhuma ativação que executou com erro. Uma avaliação experimental das consultas será apresentada no capítulo 5.

4.5 Estratégias de Otimização do ProvSearch

ProvSearch conta com a possibilidade de utilizar três tipos de estratégias para otimizar o uso da proveniência. Estas estratégias podem ser usadas em conjunto ou de forma individual

A primeira estratégia que pode ser empregada é o aumento da máquina do banco de dados de proveniência. Trata-se da estratégia mais simples, mas, em muitos casos, já é suficiente para que o experimento finalize em um tempo reduzido. Antes de aplicar a estratégia, ProvSearch precisa ter um estado consistente do banco de dados, de modo que a máquina possa ser reconfigurada e religada. A Amazon fornece um conjunto de

APIs que facilitam muito esta tarefa de reconfiguração de máquina. De maneira trivial, é possível desligar a máquina, mudar o tipo de *hardware* da mesma e religá-la. Em alguns casos, conseguimos realizar todos esses passos em menos de dois minutos. Este tempo gasto na reconfiguração, pode levar a ganho da ordem de horas, dependendo do tamanho do experimento. Como as atividades dos experimentos são executadas em paralelo, precisamos aguardar que todas finalizem, escrevam suas proveniências e fiquem aguardando pelo retorno da máquina. Uma vez feita a reconfiguração, o experimento retorna de onde parou utilizando uma máquina de proveniência com mais capacidade.

A segunda estratégia deriva da primeira, mas é utilizada quando o aumento vertical não produz o efeito esperado. Os passos iniciais são os mesmos, mas, neste caso, estamos criando uma nova máquina que trabalhará em conjunto com a primeira, oferecendo maior poder de resposta. Essa nova máquina recebe uma cópia dos dados de proveniência do experimento em questão e passará a receber metade da demanda por escrita e leitura de proveniência do experimento. Ao término do mesmo, os dados nela armazenados são transferidos para a máquina do banco de proveniência original. Finalmente esta máquina pode ser desligada.

Em conjunto com essas duas regras, temos o uso do DW, que é sempre acionado quando realizamos determinadas categorias de consultas. O DW irá trabalhar em conjunto com o banco transional e será acionado sempre que uma consulta de agregação for solicitada ao serviço de consultas, por parte do SGWfC.

Capítulo 5 - Avaliação Experimental

Para que fosse possível testar nosso conjunto de serviços, elaboramos dois cenários de testes baseados em dois workflows com características diferenciadas. Queremos assim mostrar que as estratégias utilizadas surtem efeito positivo em dois cenários representativos dos problemas envolvidos na execução dos *workflows* científicos. O primeiro trata-se de um *workflow* sintético, baseado nos experimentos da área de petróleo e gás e o segundo *workflow* foi construído baseando-se em um experimento da área de bioinformática. Neste capítulo apresentamos as configurações utilizadas para modelar e executar os *workflows* acima mencionados e a seguir detalharemos melhor os dois cenários de testes. Serão apresentados também, os resultados experimentais obtidos. A ideia central deste capítulo é analisar o desempenho dos experimentos científicos quando usamos metodologias já consolidadas comparadas ao uso do ProvSearch. Apresentamos ainda níveis de utilização que podem ser realizados deste conjunto de serviços, e a diferença de desempenho em cada um dos cenários. Para ilustrar a diferença de desempenho de consultas representativas da área, montamos um conjunto de testes que serão apresentados na seção 5.1. Na seção 5.2 elencaremos algumas questões envolvidas na configuração do ambiente como características do *hardware* envolvido. Nas seções 5.3 e 5.4 descrevemos as características dos *workflows* envolvidos nos experimentos sendo o primeiro o *workflow* sintético de petróleo e gás e o segundo um da Bioinformática. Analisamos o desempenho dos experimentos sob vários aspectos, o custo envolvido nas execuções bem como a vazão de leitura e escrita alcançadas ao variar as estratégias empregadas.

5.1 Análise de Desempenho de Consultas Típicas Relacionadas à Execução de Experimentos Científicos

Nesta seção serão expostos os resultados das consultas no banco de dados relacional, no *data warehouse* e no *data warehouse* distribuído, com o objetivo de analisar o desempenho das consultas nas diferentes soluções.

Os computadores utilizados como servidores têm as seguintes configurações: Windows Server 2008 R2, processador intel i7 1.67 GHz, 8 GB de memória RAM e 50 GB de HD.

Nos gráficos a seguir, apresentamos de forma comparativa os resultados de cada consulta. É interessante notar que mesmo nos casos em que a diferença de desempenho é pequena, continua sendo importante, pois estas consultas são executadas muitas vezes seguidas, o que significa que uma diferença de 0,1s em uma execução de consulta, pode se tornar 100s se executarmos esta consulta 1000 vezes.

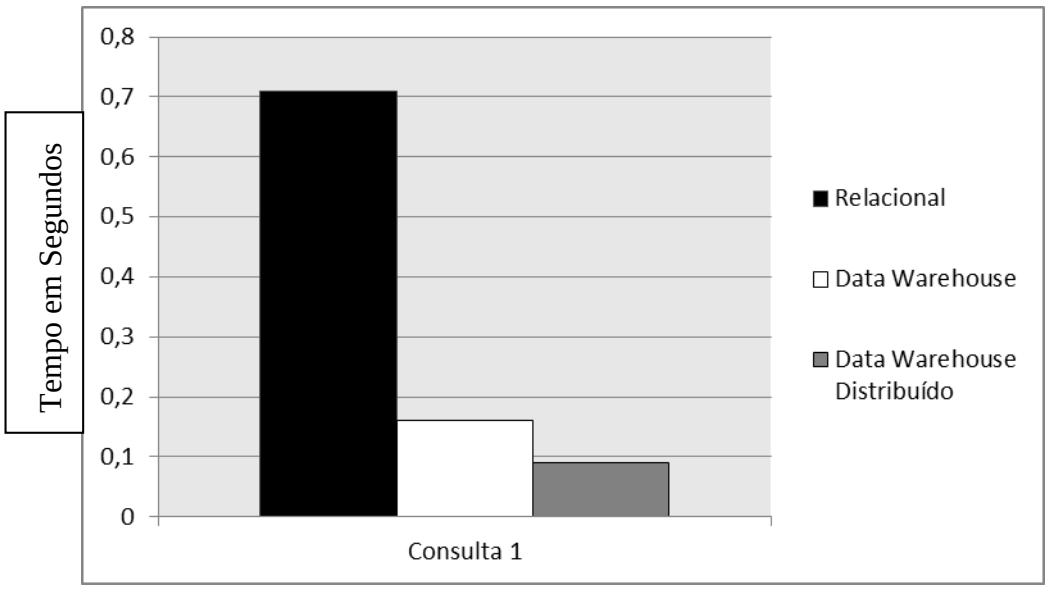


Figura 12. Gráfico comparativo dos tempos de execução da consulta 1.

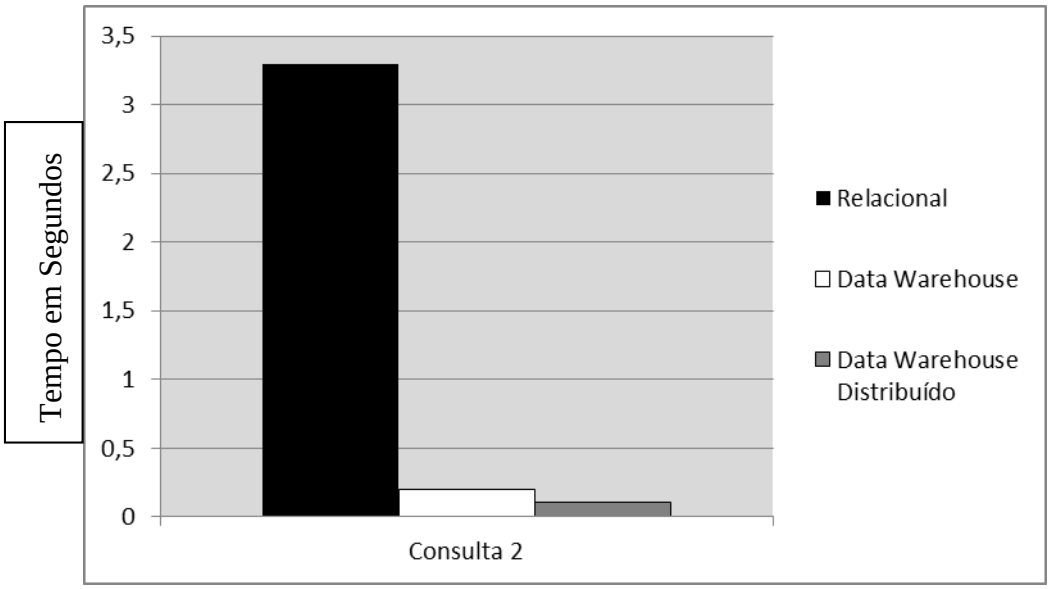


Figura 13. Gráfico comparativo dos tempos de execução da consulta 2.

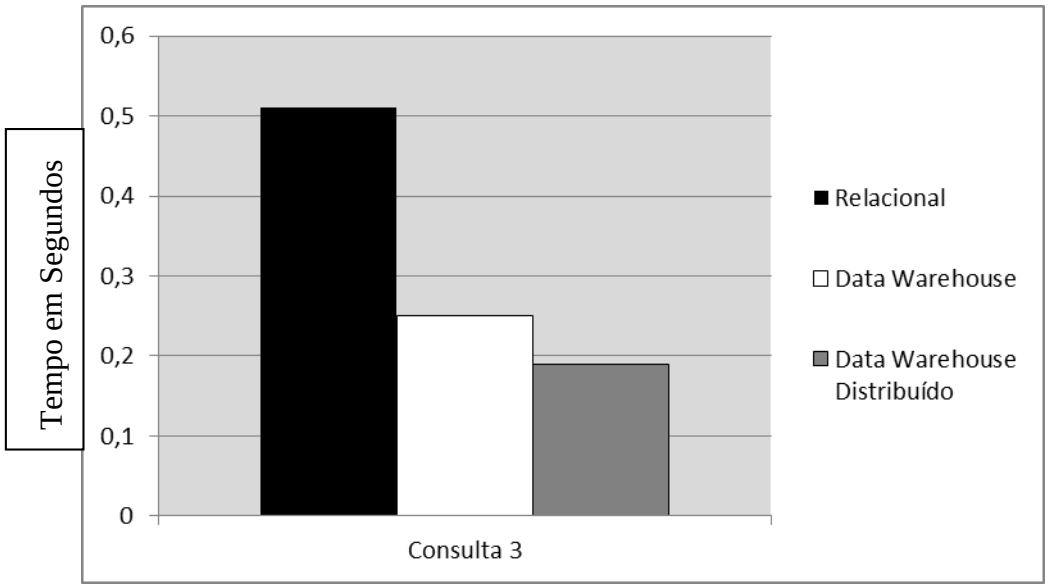


Figura 14. Gráfico comparativo dos tempos de execução da consulta 3.

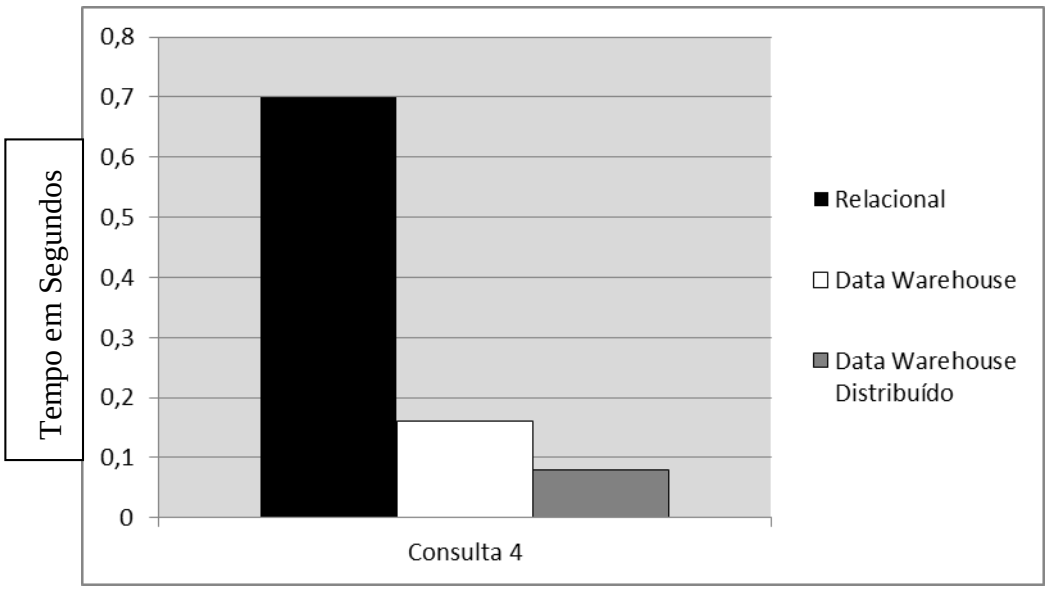


Figura 15. Gráfico comparativo dos tempos de execução da consulta 4.

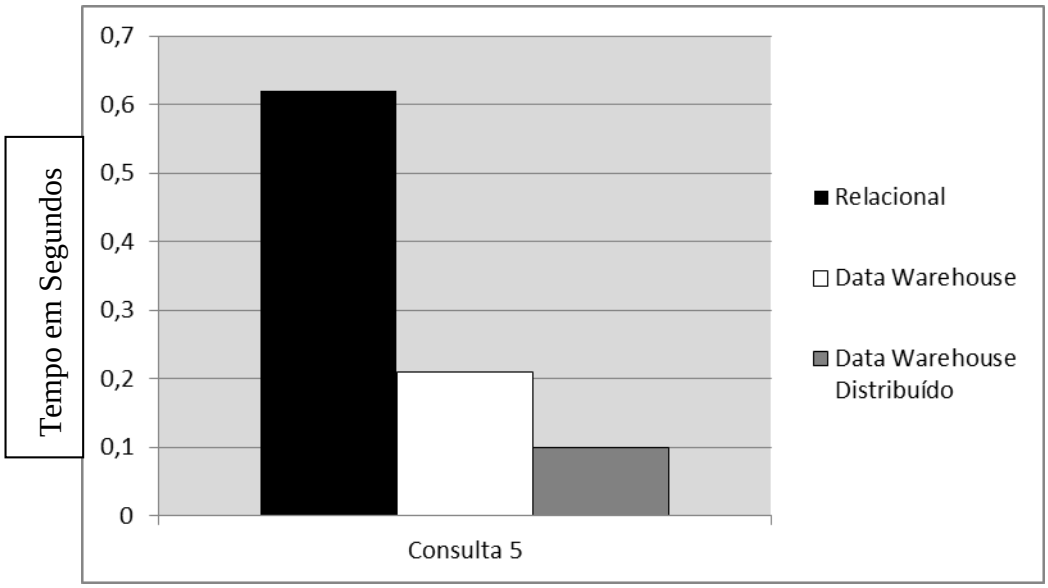


Figura 16. Gráfico comparativo dos tempos de execução da consulta 5.

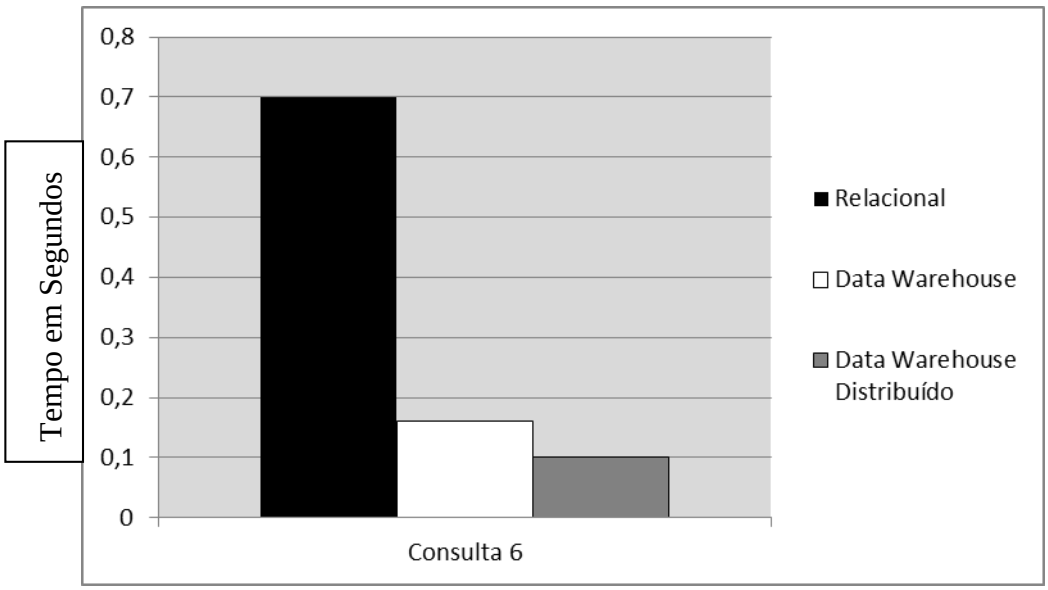


Figura 17. Gráfico comparativo dos tempos de execução da consulta 6.

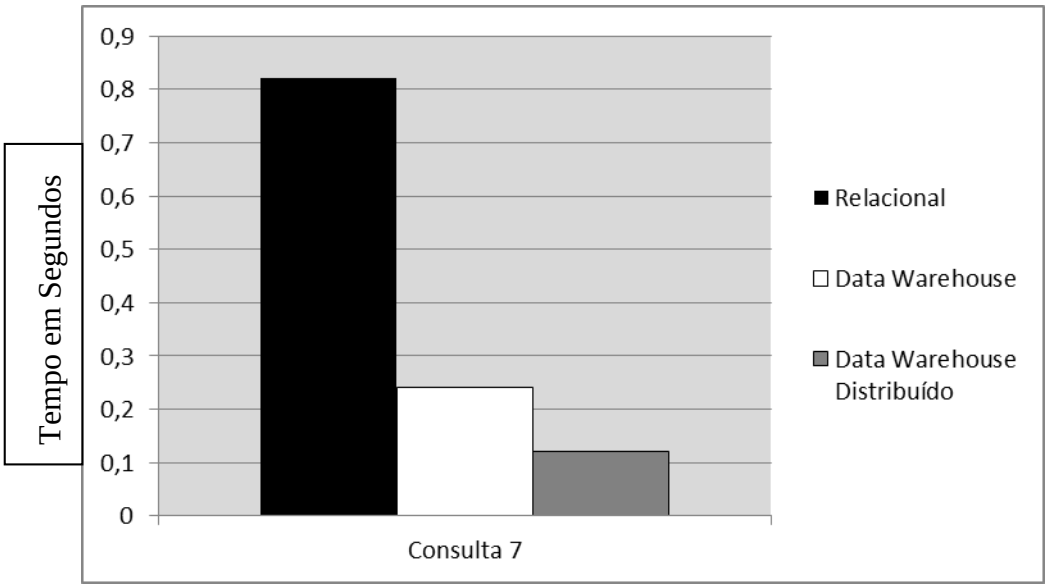


Figura 18. Gráfico comparativo dos tempos de execução da consulta 7.

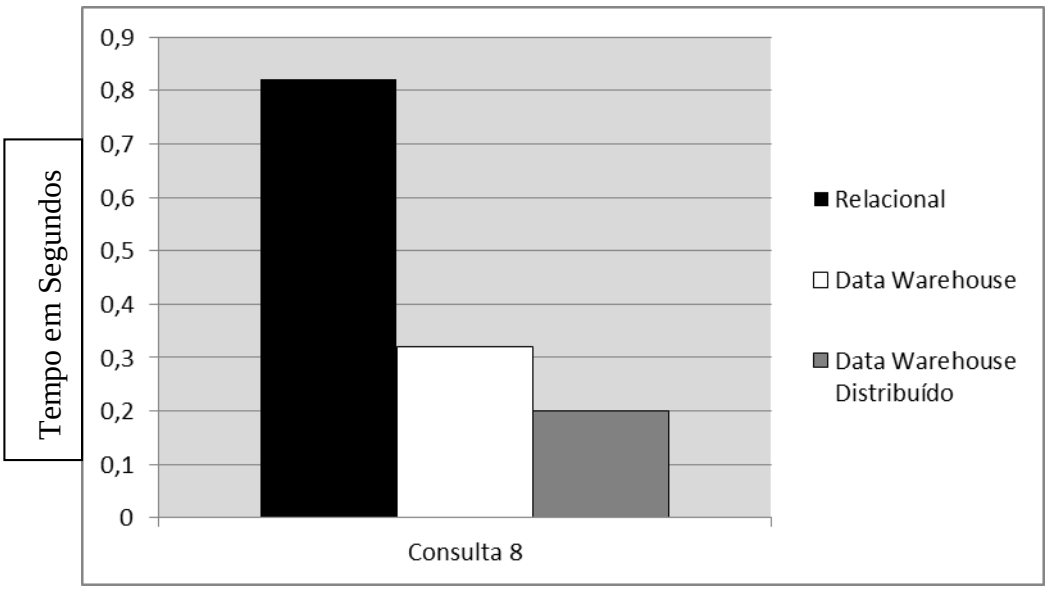


Figura 19. Gráfico comparativo dos tempos de execução da consulta 8.

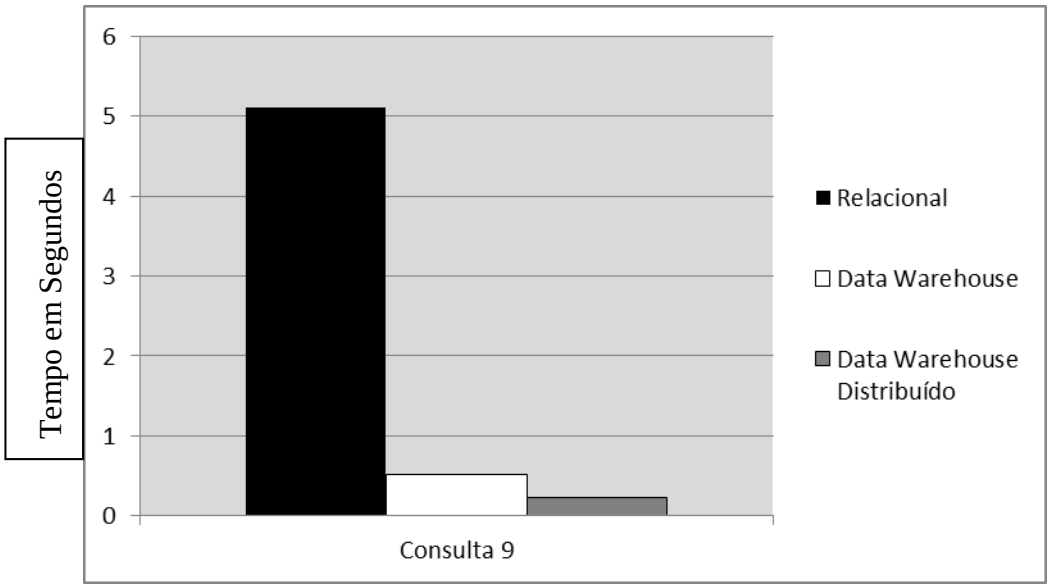


Figura 20. Gráfico comparativo dos tempos de execução da consulta 9.

Na tabela abaixo estão expostos os resultados para todas as consultas, para uma análise quantitativa. O ganhos foram constantes em todos os casos, uma vez que o *data warehouse* foi desenvolvido para atender exatamente a esta categoria de consultas.

Tabela 2. Tabela contendo os tempos de execução para cada consulta e para cada solução.

Consultas	Banco de Dados Relacional	Datawarehouse	Datawarehouse distribuído
Consulta 1	0,71s	0,16s	0,09s
Consulta 2	3,3s	0,20s	0,11s
Consulta 3	0,51s	0,25s	0,19s
Consulta 4	0,70s	0,16s	0,08s
Consulta 5	0,62s	0,21s	0,10s
Consulta 6	0,70s	0,16s	0,10s
Consulta 7	0,82s	0,24s	0,12s
Consulta 8	0,82s	0,32s	0,20s
Consulta 9	5,1s	0,52s	0,23s

Apesar de termos testado o *data warehouse* distribuído, no ProvSearch ficamos restritos ao *data warehouse* centralizado.

Nos experimentos que apresentaremos no próximo capítulo, utilizamos o DW local, próximo ao experimento. Mesmo com a base de proveniência bem grande, o DW tem tamanho bastante reduzido, da ordem de menos de 5% do tamanho da base de proveniência. Antes do início do experimento, o DW local é alimentado com informações do DW central.

5.2 Configuração do Ambiente

Para os experimentos a seguir, utilizamos a nuvem da Amazon (Amazon EC2) que fornece vários tipos de máquinas virtuais aos cientistas para instanciação e uso. Cada um destes tipos possui características únicas (capacidade de CPU, capacidade de memória RAM e capacidade de armazenamento) esses valores são aproximados, uma vez que para um conjunto de máquinas instanciadas, existem estruturas físicas da Amazon garantindo os requisitos de *hardware* solicitados, mas não existe uma máquina física para cada virtual. A definição de preços é efetuada por instância-hora consumida para cada instância, a partir do momento que uma instância é executada até ser encerrada ou interrompida. Cada hora parcial da instância consumida será faturada como uma hora cheia. Nos experimentos, quando apresentarmos as análises de custo, por simplificação, não consideraremos a análise com essa janela de uso de uma hora. Ao contrário, consideraremos o preço do uso do minuto do tipo da máquina. Esta simplificação não traz perda para as análises, pois no dia a dia de pesquisa, os experimentos podem rodar por horas ou dias e os ganhos aqui apresentados para os experimentos executados, se manterão proporcionais.

Existem vários tipos de máquinas virtuais, tais como a do tipo *micro* (micro – 1GB RAM, 1 núcleo, volume EBS de armazenamento de 30 GB), do tipo *medium* (medium - 2 GB RAM, 850 GB de armazenamento, 2 núcleos), do tipo *large* (large - 4 GB RAM, 850 GB de armazenamento, 4 núcleos) , do tipo *extra-large* (xlarge – 7,5 GB de RAM, 1.690 GB de armazenamento, 4 núcleos). Esses são apenas alguns exemplos de máquinas disponíveis, existem mais de 50 tipos de máquinas que pode ser instanciadas.

Cada uma das instâncias utiliza processadores equivalentes ao Intel Xeon quad-core. Cada máquina virtual instanciada nos experimentos nesta dissertação é baseada no

sistema operacional Linux Cent OS 5 (64 bits), e foi configurada com todos os *software*s necessários e bibliotecas como MPJ (Carpenter et al. 2000) e as aplicações da bioinformática e da área de petróleo e gás. Todas as máquinas virtuais foram configuradas para serem acessadas utilizando SSH sem verificação de senha (embora isso não seja recomendado devido a questões de segurança, que fogem ao escopo desta dissertação).

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

5.3 O Workflow Sintético Especificação de Risers

Para o primeiro experimento, modelamos um workflow sintético baseado no cenário de exploração de óleo em águas profundas através de superfícies tubulares, conhecidas como risers que pode ser visto na figura Figura 21. Os risers são as tubulações submersas que ligam as plataformas de produção e exploração de petróleo ao leito oceânico e por onde circulam o óleo extraído, gases, água e detritos sólidos provenientes da perfuração. Para que não ocorra nenhum tipo de acidente, como o rompimento destes cabos, uma série de variáveis devem ser levadas em consideração, tais como, profundidade de instalação, a correnteza a que serão submetidos, o volume de material que circulará, dentre muitos outros fatores. Para chegar à especificação ideal para o riser, as simulações de cenários são muito utilizadas e quanto maior o volume de dados a ser analisado melhor será o resultado obtido, trazendo com isso mais custo para o pesquisador dado o maior tempo a ser empregado na análise. Neste experimento executamos seis atividades que seguem em cadeia utilizando como entrada a saída produzida pelo programa imediatamente anterior (na cadeia apresentada). Na modelagem do *workflow*, fazemos um mapeamento da saída do programa X_n para o programa X_{n+1} e assim sucessivamente. Para o caso descrito, os arquivos iniciais, que guardam informações sobre o ambiente onde os risers serão submetidos, e sobre os próprios tipos possíveis de risers, foram armazenados em banco de dados. Este tipo de

informação é denominado de dado de execução e também pode ser visto como parte da proveniência.

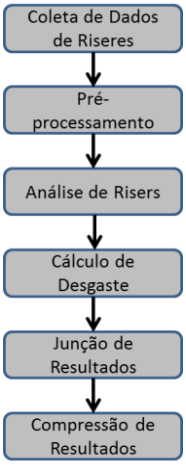


Figura 21. Representação gráfica do *workflow* sintético

Os experimentos foram realizados em uma base com um histórico de aproximadamente 1000 execuções anteriores e contendo 6.000.000 de arquivos chegando a um tamanho de 5 GB.

Este experimento foi modelado de tal maneira que durante sua execução se fez necessário acessar a base de proveniência para acessar informações analíticas sobre os risers tais como quais resultados produzidos, até um dado momento do experimento, possuem tensão menor que a média mais um desvio padrão e que o tempo gasto para produzi-los tenha sido fora da média de execução. Esta é uma consulta pesada e no contexto desse experimento é executada algumas vezes para verificar o andamento do mesmo.

```
SELECT f.fname, ea.workspace, extract(epoch from(ea.endtime-ea.starttime)) as tempoexecucao, element, node as executionnode, tension, fx, fy, fz, (select avg(tension)-2.4*stddev(tension) from oanalyzerisers) as threshold
FROM oanalyzerisers ar, efile f, eactivation ea, eactivity a
WHERE ar.sssai = f.fdir||f.fname
and f.taskid = ea.taskid
and ea.actid = a.actid
and tension < (select avg(tension)-2.4*stddev(tension) from oanalyzerisers)
and extract(epoch from(ea.endtime-ea.starttime)) > (select avg(extract(epoch from(t.endtime-t.starttime)))+2*stddev(extract(epoch from(t.endtime-t.starttime))) from eactivation t where t.actid = a.actid)
```

Outra consulta executada foi a que calcula a média de tempo de execução das atividades que fazem parte do *workflow*.

```
SELECT e.tag as nomeworkflow,
a.tag as nomeatividade,
a.status as statusatividade,
extract(epoch from(a.endtime-a.starttime)) as tempoexecucao
FROM eactivity a, eworkflow e
WHERE status='finished'
and a.wkfid = e.ewkfid
```

Além dessas consultas outras são executadas e acabam por impactar o desempenho da execução do experimento como um todo. Nesse ponto que ProvSearch irá atuar para reduzir o tempo necessário para obtenção da informação presente na base de proveniência. Não só reduzindo o tempo de acesso, mas oferecendo todos os mecanismos necessários para utilizar os recursos da nuvem de forma obter o máximo de desempenho dentro do orçamento permitido pelo cientista.

5.3.1 Análise de Desempenho

O primeiro passo realizado neste comparativo de análise de desempenho foi executar o experimento utilizando uma configuração fixa da máquina de execução. Neste caso configuramos 16 máquinas do tipo micro e não variamos esta configuração ao longo do experimento.

Conforme falamos na introdução, o primeiro experimento executa algumas consultas pesadas à base de proveniência. Estas consultas impactam diretamente no tempo total do experimento e sendo assim, ao variar a capacidade de processamento da máquina do banco de proveniência, obtivemos uma aceleração no tempo total do experimento. Na figura Figura 22 podemos ver o gráfico que mostra a evolução da redução de tempo do experimento a media que aumentamos a capacidade de processamento da máquina do banco de proveniência. Na linha mais abaixo, observamos que quando utilizamos o DW de proveniência, conseguimos reduzir ainda mais o tempo de execução. Na primeira rodada, utilizamos para o banco de proveniência uma máquina micro. Para a execução sem o ProvSearch, foram precisos em média 97 min para que o experimento finalizasse e a máquina do banco de dados trabalhou em grande parte do tempo com 100% de uso de CPU. Quando rodamos este

mesmo experimento utilizando o ProvSearch, conseguimos reduzir ambos os parâmetros, reduzindo o tempo do experimento a 12 minutos e o uso de CPU a 40%, sem ter que aumentar a capacidade de processamento da máquina de execução.

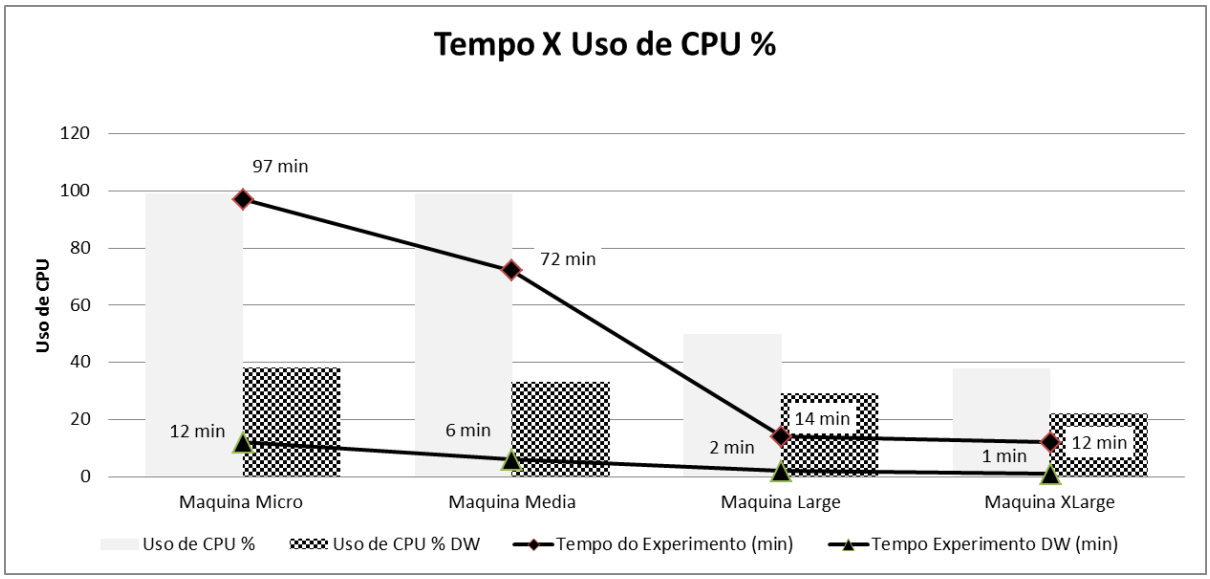


Figura 22. Gráfico comparativo dos tempos de execução X Uso de CPU

Conforme pode ser visto no gráfico, os ganhos de desempenho vão aumentando de acordo com o aumento da capacidade da máquina do banco de proveniência, mas podemos observar que para conseguir que o experimento rodasse com um tempo de 12 minutos utilizando-se o ProvSearch, uma máquina micro foi suficiente enquanto que para a execução sem estratégia, para obter um tempo equivalente foi necessário que o banco de proveniência fosse alocado em uma máquina xlarge, muito mais potente e cara.

A próxima análise que realizamos foi em relação à vazão de leitura da base de dados de proveniência ao longo do experimento. Com a utilização do ProvSearch, em todos os cenários obtivemos ganhos na quantidade de leituras de registros da base de dados. A diferença de vazão para cada uma das configurações do experimento pode ser vista no gráfico da figura Figura 23.

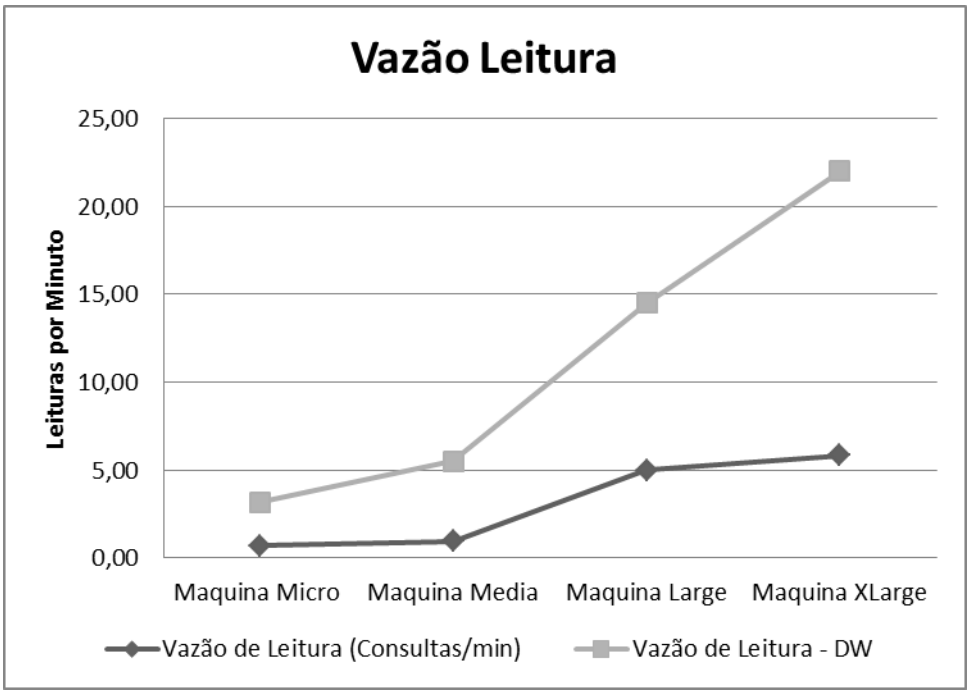


Figura 23. Vazão de Leitura

Rodamos ainda um experimento inibindo a aplicação da estratégia de uso do DW no ProvSearch. Sem esta estratégia ProvSearch começa aplicando a estratégia de aumento da capacidade do *hardware* da máquina.

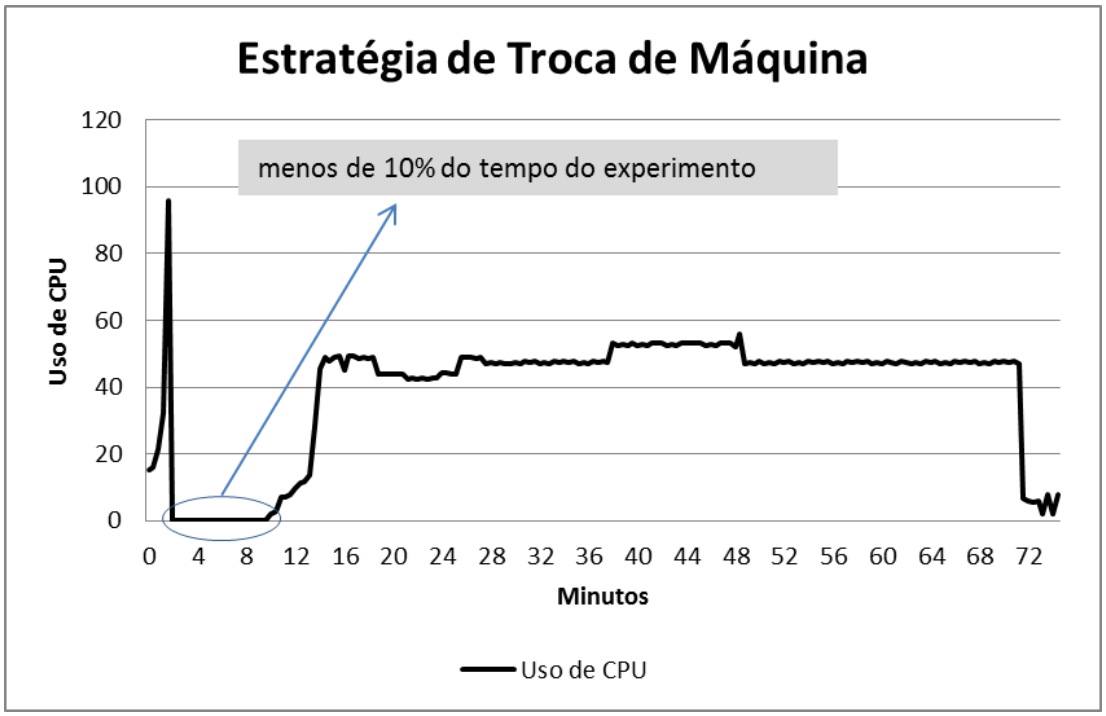


Figura 24. Custo do Experimento

No caso em questão partimos de uma máquina micro e aumentamos para uma máquina média com o tempo total de 72 minutos, fazendo com que o experimento fosse

25 minutos mais rápidos. A Amazon fornece mecanismos para troca de configuração de máquina baste eficientes que torna esta tarefa bem ágil. No gráfico da figura Figura 24 destacamos exatamente esta questão, o tempo envolvido na reconfiguração da máquina do banco de dados de proveniência. Em média foram gastos menos de 4 minutos para realizar esta tarefa, o que neste experimento representou menos de 10% do tempo total do mesmo.

5.3.2 Análise de Custo Financeiro

Outro ponto importante que não pode deixar de ser analisado diz respeito ao custo envolvido na execução do experimento. Focamos nossa análise no somatório do custo das máquinas envolvidas, ou seja, a máquina de execução do experimento mais a máquina do banco de dados de proveniência.

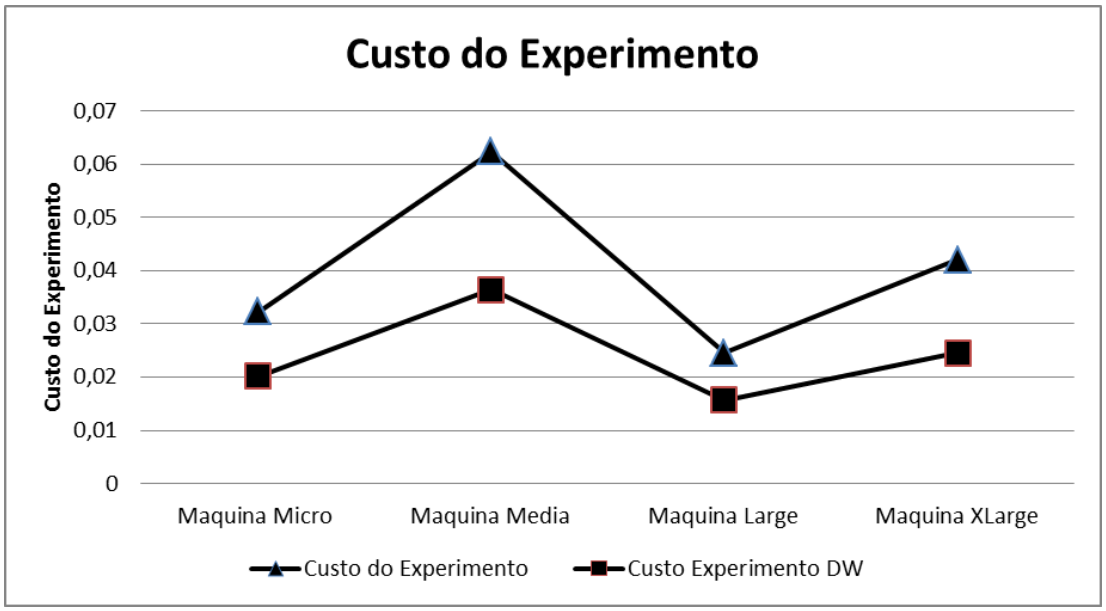


Figura 25. Custo do Experimento

Em todos os cenários analisados ProvSearch levou vantagem em relação ao experimento executado sem nenhuma estratégia, conforme pode ser observado no gráfico da figura Figura 25. Neste mesmo gráfico observamos um comportamento interessante em relação ao custo financeiro do experimento. Quando aumentamos a máquina do banco de proveniência de micro para média, acontece um ganho de desempenho e também um aumento de gasto. No entanto, no segundo nível de aumento, quando passamos de *media* para *large*, o tempo do experimento é reduzido, e o custo também. Em uma primeira análise pode parecer um contrassenso, mas a redução do tempo é tão abrupta, que mesmo utilizando-se uma máquina mais cara, acabamos por

gastar menos dinheiro, pois o tempo gasto na máquina *large*, é bem menor que o gasto na máquina *media*. Vale ainda ressaltar que, por simplificação, não estamos considerando o uso com uma janela de intervalo de uma hora (como a Amazon considera). Estamos avaliando o uso por minuto em relação ao custo da máquina por minuto, calculando, assim, o custo do experimento. Quando considerando execuções reais dos experimentos, que podem levar dias, essa questão é minimizada.

5.4 O Workflow SciPhy

Experimentos de bioinformática estão evoluindo rapidamente no contexto de projetos genômicos que analisam grandes quantidades de dados. Este fato traz consigo uma maior necessidade de infraestrutura computacional. O SciPhy (Ocaña et al. 2011b) é um workflow científico que foi desenvolvido para gerar árvores filogenéticas com máxima verossimilhança. Ele foi projetado inicialmente para trabalhar com sequências de aminoácidos, porém seu uso pode ser extrapolado para outros tipos de sequências biológicas. O workflow SciPhy é composto por sete atividades principais que são: a construção do alinhamento genético, a conversão de formato do alinhamento, a pesquisa sobre o melhor modelo evolutivo a ser usado, a construção da árvore filogenética, a concatenação dos alinhamentos produzidos de forma a gerar um super-alinhamento que servirá para inferir a relação do genoma completo, a escolha do melhor modelo evolutivo para o super-alinhamento e a construção da árvore filogenética que contém informações sobre todos os organismos associados. Estamos focados em soluções que tornem a execução de *workflows* como esse, uma tarefa mais eficiente e com custo reduzido.

Especificamente, esta tese trata das estratégias para aprimorar o uso dos recursos de elasticidade da nuvem, conseguindo, assim, reduzir o tempo dos experimentos, reduzir o tempo gasto nas análises realizadas pelos cientistas pós execução e ainda redução de gastos. Nesta etapa do processo de validação de nossas estratégias usamos como base o *workflow* SciPhy. As atividades acima citadas executam alguns programas da área biomédica, tais como: o MAFFT, o Kalign, o ClustalW, o Muscle, o ProbCons, o ReadSeq (Gilbert 2003) o ModelGenerator e o RAxML, conforme apresentado na Figura 26. Apesar de conceitualmente o SciPhy ser computacionalmente simples, uma vez que são “apenas” seis programas orquestrados, na prática ele pode ser demasiadamente complexo de ser gerenciado devido ao volume de dados trabalhados e

a quantidade de parâmetros que devem ser explorados, considerando que o cientista não conhece a priori qual a configuração que levará à árvore filogenética com melhor qualidade. Um experimento típico de filogenia pode analisar centenas ou milhares de arquivos multi-fasta, cada um contendo centenas ou milhares de sequências biológicas. Como o processamento destes arquivos e das combinações de parâmetros é independente, o SciPhy se torna um candidato interessante para explorarmos a execução paralela de suas tarefas.

Uma alternativa, dentro deste contexto, é utilizar o processamento paralelo na nuvem, com o qual é possível instanciar tantas máquinas virtuais quanto seja necessário, obtendo-se, assim, o poder de processamento para que o cientista consiga o resultado do seu experimento em um tempo satisfatório. Podemos ainda melhorar a eficiência do experimento, otimizando a manipulação dos dados de proveniência. Para este experimento, os arquivos de entrada e saída de dados não foram armazenados em banco de dados, então, o grande ganho a se obter na manipulação dos dados de proveniência foi na parte da escrita das informações.

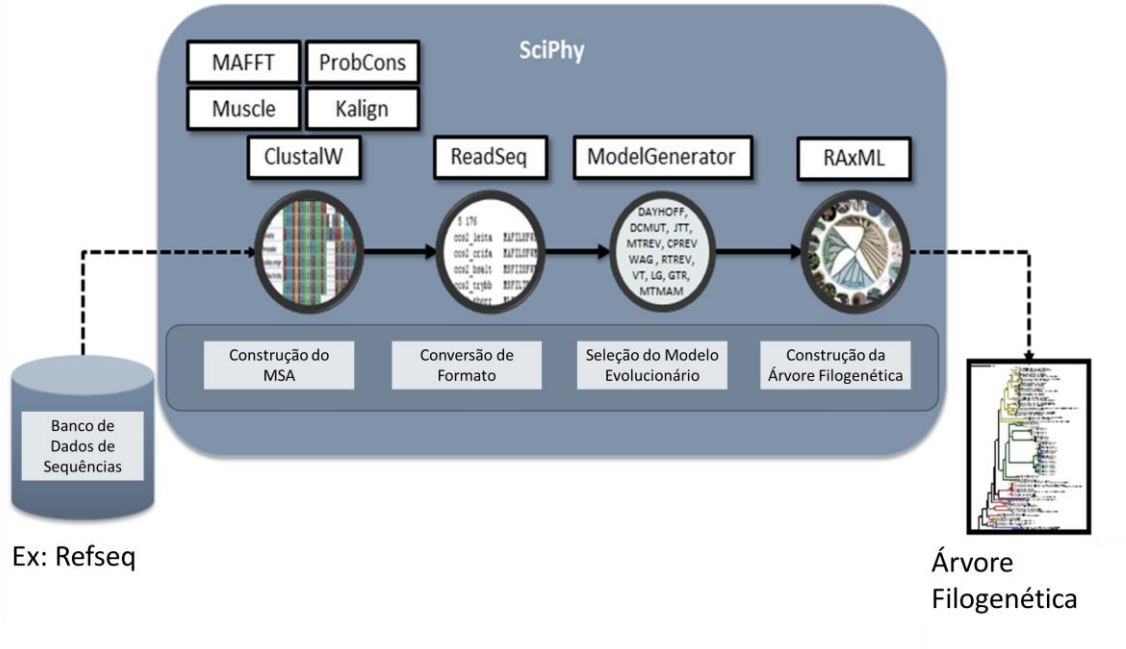


Figura 26 Fases da execução do SciPhy

Normalmente, uma execução do SciPhy em paralelo na nuvem, consumindo 50 arquivos multi-fasta, gera 1.250 atividades paralelas e exige cerca de 4 dias utilizando-se 16 núcleos virtuais. Neste experimento utilizamos arquivos multi-fasta reduzidos,

para que os tempos de processamento não tornassem proibitiva as análises, haja vista o tempo e recursos financeiros que seriam necessários.

Para executar o experimento, variamos o programa de alinhamento e o modelo evolucionário utilizado no *workflow* científico. Nossas execuções utilizam como entrada um conjunto de arquivos multi-fasta contendo sequências de proteínas extraídas do banco de dados biológicos RefSeq *release* 48 (Pruitt et al. 2009) e do ProtozoaDB (Dávila et al. 2008), disponibilizado pelo Prof. Dr. Alberto Martín Rivera Dávila da Fundação Oswaldo Cruz. Este conjunto de dados é formado por 1.600 arquivos multi-fasta de aminoácidos, e cada arquivo multi-fasta é constituído por uma média de 20 sequências. Para este experimento, utilizamos uma base de dados de entrada de aproximadamente 4 GB, que por nossa experiência, tem tamanho médio para experimentos de bioinformática de larga escala.

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

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

5.4.1 Análise de Desempenho

Antes de apresentar os resultados obtidos das comparações das diferentes abordagens, ressaltamos que os valores de tempo e uso de CPU apresentados foram obtidos por meio do cálculo da média de três execuções. O ambiente de nuvem é um ambiente de grande oscilação, principalmente quando usamos as máquinas mais baratas. Por essa razão

adotamos essa estratégia, no sentido de minimizar possíveis falhas de avaliação, melhorando assim o resultado final obtido.

A primeira análise que realizamos foi em relação ao uso de CPU ao longo do tempo. Para tal, executamos o mesmo workflow em cinco configurações diferentes de *hardware*. A base de proveniência relacionada ao experimento foi configurada em uma máquina *micro* da Amazon e para esta primeira análise não utilizamos nenhuma das estratégias apresentadas, mantendo a máquina de proveniência inalterada ao longo de todo experimento. Na figura Figura 27, para cada cenário são apresentados o tempo total do experimento na linha que decai e o uso médio de CPU da máquina de proveniência nas barras. Por exemplo, quando o experimento é executado com duas máquinas, o tempo total de execução foi de 140 minutos. A máquina de proveniência ficou subutilizada, tendo um consumo médio de CPU de 37%. Porém, à medida que aumentamos o número de nós da máquina de execução, ocorreu também um aumento do uso da máquina do banco de proveniência. Tomando como exemplo a execução com 16 nós, observamos que o uso médio de CPU foi de 80%. Aparentemente, ainda houve uma subutilização da máquina do banco de proveniência, uma vez que 80% não sugere um gargalo de utilização.

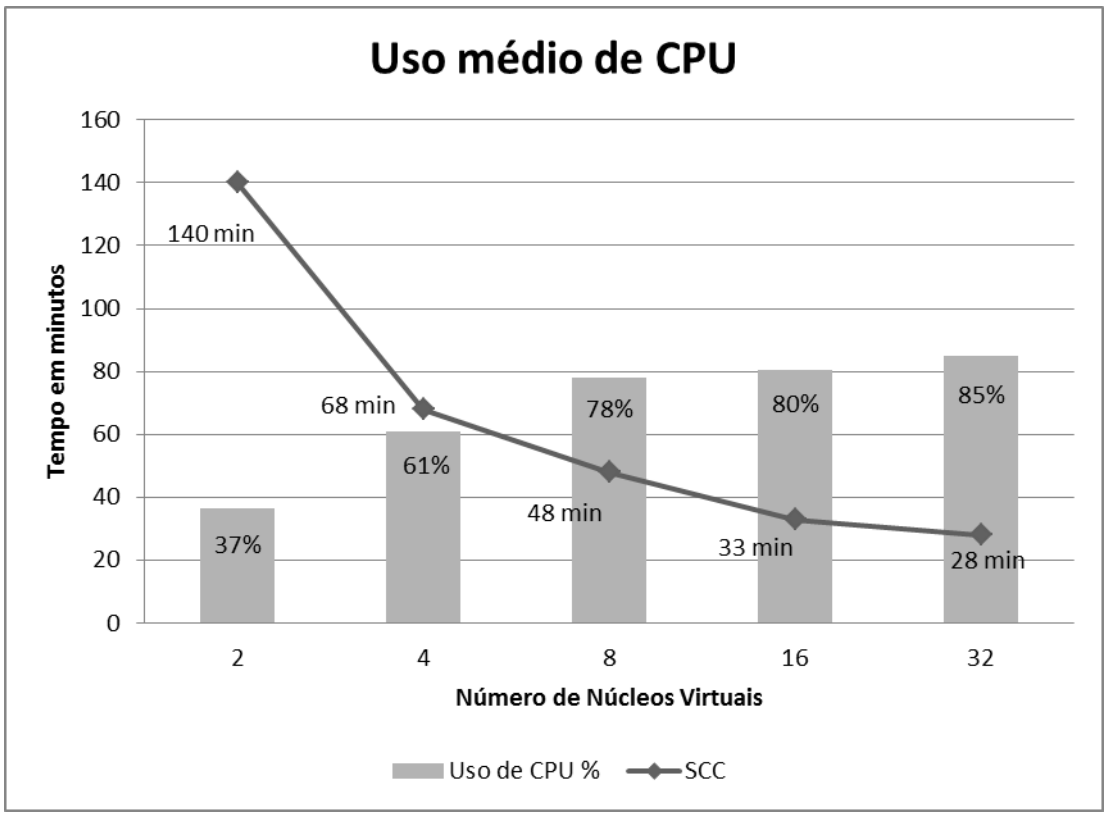


Figura 27. Uso médio de CPU

No entanto, quando observamos o gráfico da figura Figura 28, fica evidenciado que, por período considerável de tempo, a máquina do banco de dados de proveniência trabalhou no limite de CPU.

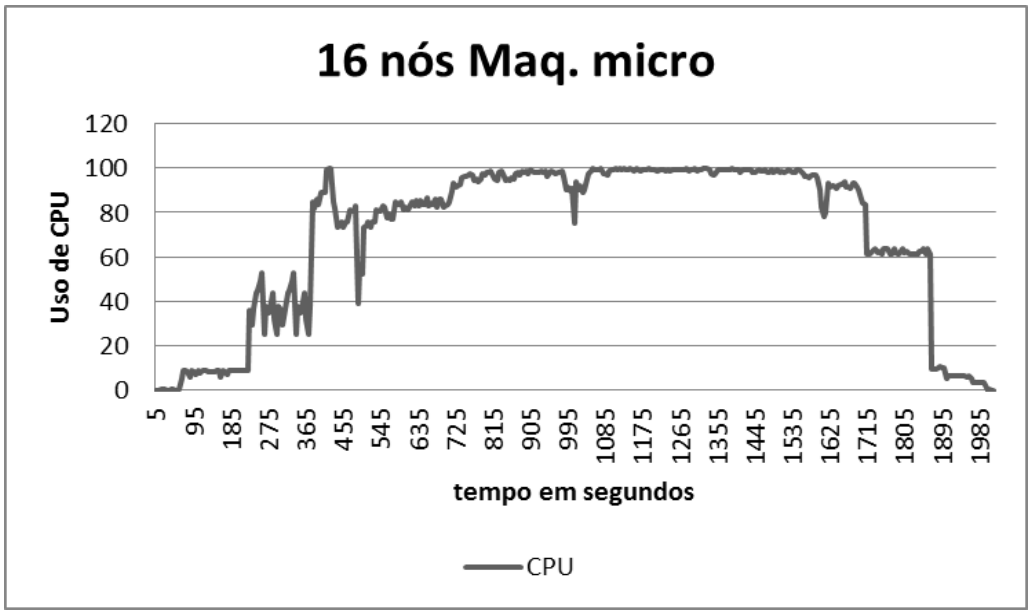


Figura 28. Uso médio de CPU

O mesmo ocorre quando o experimento é executado com 32 nós, conforme pode ser visto na figura Figura 29. Neste gráfico observamos que a máquina do banco de proveniência, durante a execução de experimento, sofreu uma utilização de CPU que sugere um uso acima de sua capacidade, mantendo-se em 100% por quase todo o experimento.

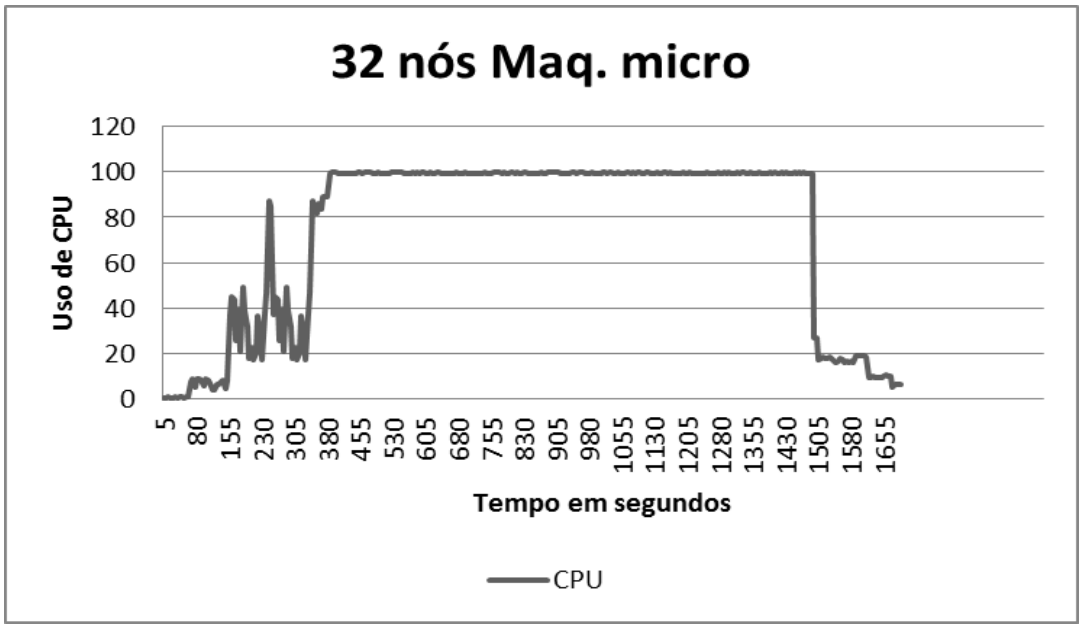
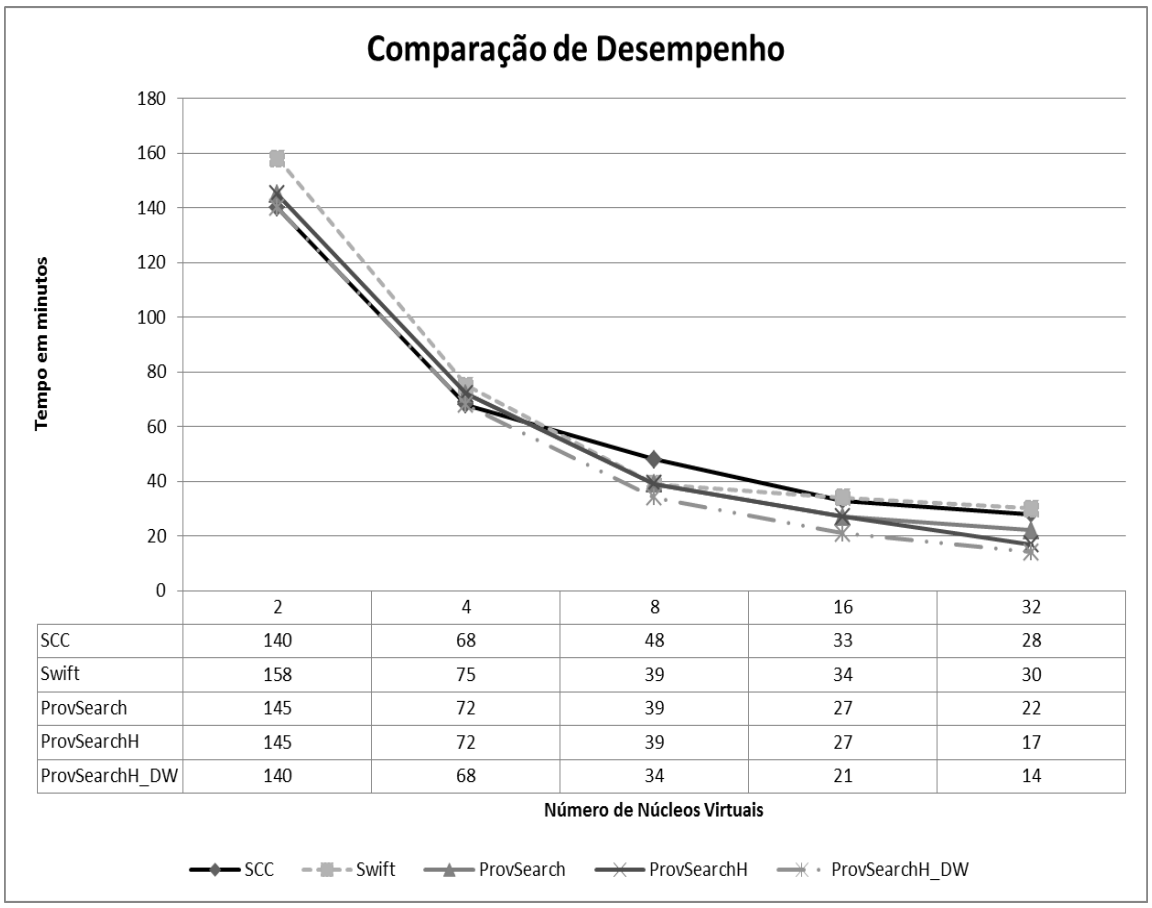


Figura 29. Uso médio de CPU

Esses gráficos sugerem que, dependendo das características do experimento, o ambiente preparado para cuidar de sua proveniência pode não ser suficiente. Porém, precisamos ainda analisar se o gargalo causado pela infraestrutura da base de proveniência impacta negativamente na execução do experimento como um todo. Para isso, comparamos a execução do experimento com a configuração do *hardware* da base de proveniência inalterada, com execuções que utilizaram diferentes estratégias de adaptação do ambiente.



	2	4	8	16	32
SCC	140	68	48	33	28
Swift	158	75	39	34	30
ProvSearch	145	72	39	27	22
ProvSearchH	145	72	39	27	17
ProvSearchH_DW	140	68	34	21	14

Figura 30. Uso médio de CPU

No gráfico da figura Figura 30. Uso médio de CPU vemos a comparação de cinco execuções do mesmo experimento. Primeiro executamos o experimento em dois SGWfCs consolidados no meio científico, o SCC e o Swift, de modo que pudéssemos ter uma linha de base do tempo de execução para diferentes configurações de *hardware.* Subtemos o experimento a cinco cenários distintos, utilizando duas, quatro, oito, dezesseis e por fim trinta e duas máquinas virtuais. Nestas execuções não foi utilizada nenhuma estratégia de otimização do uso da proveniência. Na primeira execução com uso do ProvSearch, habilitamos apenas a estratégia de aumento da capacidade da

máquina (também chamada de aumento vertical), que, para a maioria dos cenários, já foi suficiente para trazer ganhos de tempo para o experimento. Na execução com 32 nós obtivemos ganhos de 22% em relação ao Swift e de 27% em relação ao SCC. Ampliando o uso das estratégias do SCC, rodamos ainda mais dois conjuntos de experimentos, permitindo que a máquina do banco de dados de proveniência pudesse ser também aumentada horizontalmente (com acréscimo de outros nós) e, por fim, que pudesse executar algumas das consultas utilizando a base do DW. Neste novo cenário de testes, obtivemos ganhos ainda mais expressivos, como pode ser visto no gráfico da figuraFigura 30. Uso médio de CPU. Utilizando como exemplo a execução com 32 nós em relação ao Swift e ao SCC, otimizamos a execução do experimento em 54% e 50%, respectivamente, quando utilizamos todas as estratégias possíveis.

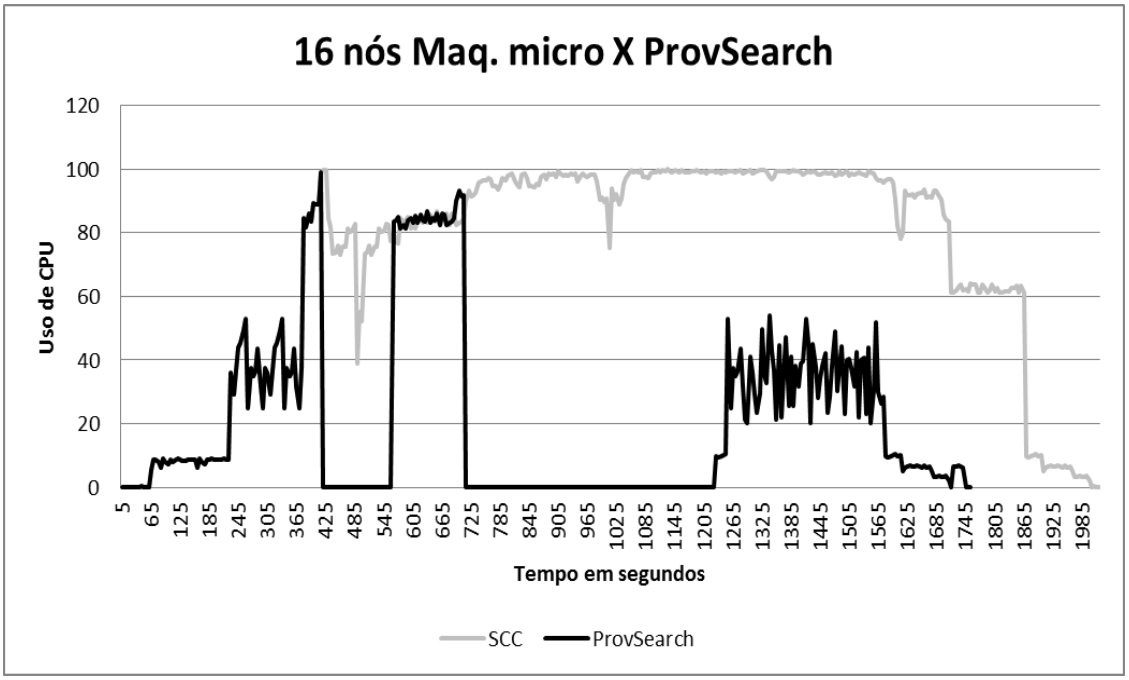


Figura 31. Aplicação das Regras do ProvSearch

Para ilustrar o funcionamento da aplicação das estratégias montamos o gráfico da figura Figura 31. Nele podemos ver dois momentos em que o experimento fica parado. Esses intervalos representam o momento em que o ProvSearch está redimensionando a máquina do banco de proveniência. No tempo 425 é identificado um evento de *flash crowd*, e, então, ProvSearch aplica a primeira regra: o aumento vertical da máquina de proveniência. Feito o ajuste, o experimento retorna a execução para o ponto que havia parado. No entanto, no segundo 725, um novo evento de *flash crowd* é identificado, fazendo com que uma nova estratégia, mais complexa, seja utilizada: o

aumento horizontal. Após realizar todos os passos descritos anteriormente, o experimento é então retomando e segue com essa configuração até o final. Podemos observar no gráfico que, mesmo com essas duas paradas, ainda terminamos a execução do experimento em um tempo anterior à execução que não utilizou nenhum tipo de estratégia.

Apresentamos, por fim, o gráfico da figura Figura 32. Vazão de Escrita, referente à vazão de escrita do experimento. Este experimento possui uma característica de produzir muita proveniência, pois executamos muitas atividades em paralelo que levam pouco tempo para produzir seus resultados. Por conta dessa característica, quando aumentamos o número de execuções paralelas, a demanda por escrita de informações aumenta muito e a máquina de proveniência, se for mal dimensionada, rapidamente se torna um gargalo para a boa execução do experimento como um todo. Isso fica evidenciado quando observamos que a estratégia de aumento horizontal produziu os melhores ganhos. Ao ter um ambiente distribuído de escrita, a sobrecarga passou a ser divida em dois nós, reduzindo a concorrência e aumentando bastante a vazão. A estratégia de aumento vertical da máquina não resolveu o problema do gargalo. Tínhamos uma configuração mais potente de máquina, mas não conseguíamos atender a demanda por escrita na base.

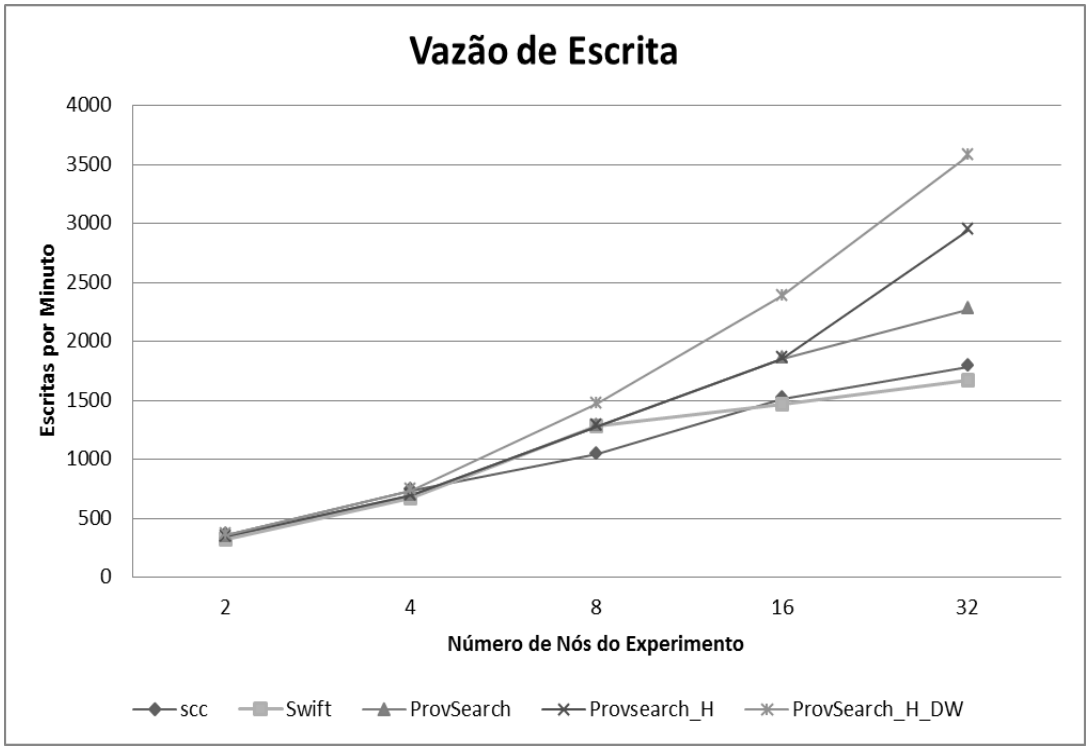


Figura 32. Vazão de Escrita

5.4.2 Análise de Custo

A fim de verificar o custo monetário envolvido na execução do experimento, precisamos analisar os custos envolvidos no uso da infraestrutura utilizada para execução do experimento em si e também para armazenar e acessar os dados de proveniência. Calculamos o custo final utilizando uma adaptação do modelo de cobrança da Amazon EC2. Originalmente ele é baseado na cobrança por janelas de tempo de uma hora. Por simplificação utilizamos uma análise de cobrança por minuto, método de pagamento semelhante ao do GoGrid em que se desconta apenas o tempo utilizado efetivamente por minuto de uso. Na tabela Tabela 3 Configuração de Hardware e Preços de Utilização temos um resumo dos dados de desempenho e precificação dos tipos de máquinas virtuais que utilizamos ao longo dos experimentos.

Tabela 3 Configuração de Hardware e Preços de Utilização

Tipo de Máquina	Memória	Disco	UC[1]	Núcleos	Arquitetura	Preço[2]
Micro	1 GB	--	1 UEC2	1	32 Bits	0,02
Medium	4 GB	--	2 UEC2	1	32 Bits	0,05
Large	7,5 GB	32 GB	2 UEC2	2	64 Bits	0,10
Extra-large	7,5 GB	160 GB	4 UEC2	4	64 Bits	0,21

Assim como no experimento com o workflow sintético, à medida que aplicávamos as estratégias, o custo total do experimento foi diminuindo, mostrando que as estratégias discutidas, além de tornarem os experimentos mais rápidos, representam também uma redução de gastos. É importante destacar que, apesar de nos experimentos rodados a redução absoluta de valores ser pequena, quando projetamos esses ganhos para experimentos de longa duração, chegamos a valores bem expressivos. A redução dos custos financeiros é tão importante quanto a redução de tempo, pois muitos cientistas sofrem com a escassez de recursos financeiros. Para a execução dos experimentos dessa tese sofremos com este problema, a todo tempo havia a preocupação de estourar nossos orçamentos. Mesmo rodando experimentos de tamanho reduzido e com constante monitoramento dos recursos gastamos mais de 1000 dólares.

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

[2] Preço calculado por hora em dólares

5.5 Conclusão

Pelo conjunto dos resultados dos experimentos apresentados atestamos que o uso do conjunto de serviços que desenvolvemos trouxe bons ganhos para a execução de experimentos científicos. Ainda existem lacunas a serem aprimoradas, tais como disponibilizar este conjunto de serviços em um portal ou melhorar a capacidade de decisão do analisador de consultas, para direcionar as consultas para o *data warehouse* ou para a base transacional. Atualmente implementamos a solução baseada em um conjunto de consultas predefinidas, mas já existe uma implementação em andamento, baseada em um conjunto de heurísticas.

Diante dos resultados obtidos, não temos dúvidas que as contribuições apresentadas apontam para um conjunto de funcionalidades que devem ser experimentadas no contexto dos experimentos científicos dependentes de PAD e com uso de proveniência.

Capítulo 6 - Trabalhos Relacionados

Não encontramos, na literatura da área, trabalhos que gerenciassem a proveniência de forma distribuída em nuvens computacionais, tendo o cuidado de explorar sua elasticidade. Muitas soluções implementadas hoje veem na nuvem um recurso virtual, mas o utilizam como se fossem máquinas físicas, sem se preocupar em variar sua configuração, caso haja aumento ou decréscimo de demanda.

O que existem são trabalhos com pontos de interseção com as estratégias aqui apresentadas. Em (Souza e Mattoso 2015) está claro que a gerência com um banco de dados de proveniência de forma distribuída traz vantagens significativas em relação à abordagem centralizada. No entanto, nesse trabalho, tal abordagem não é usada para nuvens computacionais, tendo sido desenvolvida para ambientes de *clusters*. Outro ponto que não é abordado pela solução em questão é o uso de um *data warehouse* para obter ganho de desempenho nas consultas.

Podemos encontrar, ainda, alguns trabalhos que abordam a questão da captura e consulta da proveniência no nível dos sistemas de arquivos paralelos e distribuídos, porém tais estratégias não podem ser utilizadas como forma de acelerar o tempo de execução do experimento, pois a proveniência neste caso é de muito baixo nível. Na abordagem de (Allen *et al.* 2011) eles usam técnicas P2P para descobrir os dados de proveniência distribuídos. Eles adotam uma arquitetura para troca de proveniência multi-organizacional que é baseada em tabelas de *hash* distribuídas (DHT). O DHT é usado apenas para armazenar informações sobre a localização de um repositório de proveniência, sem levar em conta o seu conteúdo. Existem abordagens centradas em sistemas de arquivos distribuídos, que usam metadados para encontrar os recursos, sem considerar o conteúdo dos arquivos. Um exemplo é o que usa informações vetorizadas (Ohnishi *et al.* 2007) ou ainda esquemas especializados, tais como IFS (Lee *et al.* 2002) ou NEMO (Lin e Li 2007). O Distributed Provenance Aware Storage System (DPASS) é usado para capturar a proveniência dos arquivos em sistemas distribuídos por meio da interceptação das operações do sistema de arquivos. Eles trabalham no nível do sistema operacional e a proveniência obtida é, no entanto, armazenada de forma centralizada, como discutimos anteriormente pode não ser a melhor escolha para aplicações distribuídas (Muniswamy-Reddy *et al.* 2006). Muniswamy-Reddy (Muniswamy-Reddy *et al.* 2009) é uma outra solução no nível do sistema operacional. Eles apresentam um

sistema especializado de armazenamento de proveniência que permite sua integração em várias camadas de abstração, variando o armazenamento dos dados, de aplicações Python até aplicações que rodam no nível da camada de rede. Uma boa revisão em dados de proveniência distribuídos, também no nível do sistema de arquivos, é apresentada em (Zhao *et al.* 2013). Sua avaliação explora desempenho e consultas a proveniência para sistemas paralelos de arquivos, utilizando FusionFS e Spade, nos quais a representação de proveniência inclui tanto as operações de arquivo e dados de proveniência tal como entendemos.

Em alguns outros trabalhos, como (Abawajy et al. 2013) e (Zhou et al. 2012), a proveniência mencionada está mais próxima aos registros do histórico das operações realizados no ambiente distribuídos. Os autores descrevem metodologias para que todo o registro distribuído da proveniência possa ser acessado como um repositório único, mas sem se preocupar com o escalonamento dos recursos de *hardware,* por não se tratar de ambientes pagos por uso.

As soluções existentes para gerência de dados de proveniência em sistemas distribuídos de arquivos são altamente escaláveis, mas, em geral, com granularidade muito fina, no nível do sistema operacional, com foco no registro de dados que, para nossa realidade, não são relevantes e ao mesmo tempo são difíceis de serem interpretados. Estas soluções estão fortemente baseadas na manipulação de informação no nível dos arquivos de sistema e sendo assim não poderiam tirar proveito das técnicas de banco de dados distribuídos. Nesta tese visamos a combinar o melhor das soluções de arquivos distribuídos com a gerência distribuída a dados de proveniência de SGWfC.

Capítulo 7 - Conclusões

Ao longo dos últimos anos, *workflows* estão sendo utilizados para a gerência de experimentos científicos com base em simulações computacionais. Como essas experiências se tornam cada vez mais complexas, mais dados de proveniência relacionados com estas execuções são capturados, armazenados e consultados. Além disso, alguns SGWfC são desenvolvidos para trabalhar com proveniência durante ou após a execução do experimento. A necessidade de consultar dados de proveniência em tempo de execução de forma eficiente, nestes casos, é imprescindível. No entanto, devido às características distribuídas do problema, abordagens centralizadas apresentam limitações e não exploram todo potencial das nuvens computacionais. Desta forma, os dados de proveniência precisam ser vistos como um grande repositório distribuído que sofre oscilação no volume de acessos e pode ter os recursos de *hardware* dinamicamente adaptados, para que a infraestrutura não atinja um ponto máximo de utilização, ocasionando uma perda de desempenho.

O ProvSearch gerencia a distribuição dos dados de proveniência com técnicas de sistemas de bancos de dados distribuídos. Consideramos dois níveis de bancos de dados de proveniência, um mais próximo ao da máquina que executa o experimento (ou o conjunto de máquinas envolvidas na execução do *workflow*) e um nível mais elevado que resume e reúne informações específicas dos vários bancos de dados de proveniência, proporcionando, assim, estatísticas que podem ser usadas por SGWfC orientados ao uso de proveniência. ProvSearch é capaz também de dimensionar a infraestrutura das bases de proveniência para atingir um nível otimizado de uso. Sempre que se faz necessário, ele pode aumentar ou diminuir a capacidade do ambiente, evitando assim gastos desnecessários. ProvSearch disponibiliza suas funcionalidades por meio de serviços que podem ser publicados em um portal.

Para que seja viável a utilização do conjunto de serviços disponibilizados, é preciso que o SGWfC seja compatível com as ideias apresentadas, no que tange o uso da proveniência.

7.1 Contribuições Realizadas

Os resultados apresentados nesta tese demonstram que a otimização do uso da proveniência pode trazer uma série de benefícios para o pesquisador, principalmente em

relação à redução de custos e tempo de processamento. Buscamos alternativas à gerência centralizada dos dados de proveniência de experimentos científicos. À medida que os workflows de experimentos se tornam mais complexos e mais detalhes precisam ser armazenados, o modelo centralizado pode ser um gargalo para a execução distribuída e esta abordagem não tira proveito de uma das principais características da nuvem: a elasticidade. Os resultados indicaram que o uso da abordagem proposta nessa tese propiciou uma melhor utilização dos recursos da nuvem, trazendo um ganho na escalabilidade da solução de gerência de proveniência e melhorou o tempo de execução dos *workflows* científicos, oferecendo dados consolidados que são consumidos ao longo da execução do *workflow*.

Apresentamos, ainda, uma proposta de modelagem para dados de proveniência derivada do modelo já consolidado do W3C PROV-DM. Esta metodologia abre portas para ganhos de desempenho em diferentes tipos de consultas executadas sob a base de proveniência e facilita inclusive a integração de bases de proveniência geradas por diferentes SGWfCs. As estratégias apresentadas aceleraram as consultas e escritas à base de dados de proveniência. Em relação às consultas, sejam elas executadas em tempo de execução para escalonamento de recursos, ou *steering*, ou sejam consultas OLAP executadas pelos cientistas. Outra importante contribuição do nosso trabalho foi o ProvSearch: uma solução para gerência da proveniência na nuvem, transparente ao usuário. O conjunto de serviços disponibilizados são responsáveis pela localização do dado de proveniência desejado, gerência de estatísticas e gerência de recursos distribuídos na nuvem. Independente do SGWfC no qual ProvSearch é acoplado, este fica responsável por armazenar e consultar a proveniência gerada pelo experimento científico em execução. Fica ainda responsável, sempre que for preciso, por obter dados históricos de experimentos passados, utilizando uma estratégia eficiente, trazendo redução no tempo total de execução do experimento.

Ao longo dos últimos anos, os resultados parciais obtidos foram sendo publicados e apresentados em eventos da área, destacando-se:

Contribuições sobre o uso da proveniência para a publicação na revista Future Generation Computer Systems, publicação de grande destaque na área, edição de 2015.

i. MATTOSO, M. ; DIAS, J. ; OCAÑA, K. ; OGASAWARA, E. ; **COSTA, F.** ; HORTA, F. ; SILVA, V. ; DE OLIVEIRA, DANIEL . Dynamic steering of HPC

scientific *workflow*s: A survey. Future Generation Computer Systems, v. 46, p. 100-113, 2015.

A apresentação de resultados no IEEE 10th International Conference on e-Science realizado em São Paulo em 2014. Nesta oportunidade descrevemos nosso conjunto de serviços para execução de um experimento na nuvem de forma distribuída.

ii. **COSTA, F.** ; OLIVEIRA, DANIEL DE ; MATTOSO, M. . Towards an Adaptive and Distributed Architecture for Managing *Workflow* Provenance Data. In: 2014 IEEE 10th International Conference on eScience (eScience), 2014, Sao Paulo. 2014 . p. 79.

No Workshop of Provenance Analytics, realizado em Colônia na Alemanha em conjunto com o 5th International Provenance and Annotation Workshop em 2014, Evento tradicional (bianual, desde 2006) do uso de proveniência que apresenta resultados referentes à execução de *workflow*s científicos e coleta de dados de proveniência utilizando mais de uma fonte de dados, inclusive com bancos de dados diferentes.

iii. **COSTA, F.** ; SILVA, V. ; OLIVEIRA, D. ; OCAÑA, K. ; MATTOSO, M. . Towards Supporting Provenance Gathering and Querying in Different Database Approaches. In: 5th International Provenance and Annotation Workshop (IPAW), 2014, Cologne. Proc. of the 5th Intl. Provenance and Annotation Workshop, 2014.

Em 2013 apresentamos no First International Workshop on Managing and Querying Provenance Data at Scale evento realizado em conjunto com o EDBT/ICDT em 2013, em Genova, Itália, nossa modelagem para representação, de forma compatível com o padrão W3C PROV, dos dados da proveniência gerada por equipes dispersas. Uma vez que os dados de proveniência sigam um padrão de representação, a possibilidade de compartilhamento de informação fica facilitada. A modelagem que desenvolvemos serviu de inspiração para o desenvolvimento de um padrão voltado a modelagem da proveniência de

workflows científicos por um dos grupos mais importante importantes na criação de padrões, o DataOne, (Ludascher et al. 2014). Inclusive nossa modelagem continuou compatível com a criada pelo grupo. Nossa publicação sobre esse assunto já foi baixada mais de 200 vezes e referenciada por 35 artigos (de acodo com o Google Scholar).

iv. **COSTA, F.** ; SILVA, V. ; OLIVEIRA, D. ; OCAÑA, K. ; OGASAWARA, E.; DIAS, J. ; MATTOSO, M. . Capturing and querying workflow runtime provenance with PROV. In: First International Workshop on Managing and Querying Provenance Data at Scale, 2013, Genova, Itália. Proceedings of the Joint EDBT/ICDT 2013 Workshops - First International Workshop on Managing and Querying Provenance Data at Scale. New York: ACM Press, 2013. p. 282-289.

Destaco ainda alguns artigos que publicamos no contexto desta tese e que de alguma forma ajudaram no desenvolvimento e validação das ideias aqui apresentadas:

v. OLIVEIRA, D. ; **COSTA, F.** ; SILVA, V. ; OCAÑA, K. ; MATTOSO, M. . Debugging Scientific *Workflow*s with Provenance: Achievements and Lessons Learned. In: XXIX Simpósio Brasileiro de Banco de Dados, 2014, Curitiba, PR. Anais do XXIX Simpósio Brasileiro de Banco de Dados. Porto Alegre: Sociedade Brasileira de Computação, 2014.

vi. SANTOS, E. ; ASSIS, V. ; **COSTA, F.** ; OLIVEIRA, D. ; MATTOSO, M. . Distribuição de Bases de Dados de Proveniência na Nuvem. In: XXVIII Simpósio Brasileiro de Banco de Dados, 2013, Recife. Anais do XXVIII Simpósio Brasileiro de Banco de Dados. Porto Alegre: Sociedade Brasileira de Computação, 2013. p. 1-6.

vii. MATTOSO, M. ; OCAÑA, K. ; HORTA, F. ; DIAS, J. ; OGASAWARA, E. ; SILVA, V. ; OLIVEIRA, D. ; **COSTA, F.** ; ARAUJO, I. . User-steering of HPC *workflow*s. In: 2nd ACM SIGMOD Workshop on Scalable *Workflow* Execution Engines and Technologies, 2013, New York. Proceedings of the 2nd ACM

SIGMOD Workshop on Scalable *Workflow* Execution Engines and Technologies. New York: ACM Press, 2013. p. 4-10.

Referências Bibliográficas

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

Abadio, A. K. R., Kioshima, E. S., Teixeira, M. M., Martins, N. F., Maigret, B., Felipe, M. S. S., (2011), "Comparative genomics allowed the identification of drug targets against human fungal pathogens", *BMC Genomics*, v. 12, p. 75.

Abawajy, J. H., Jami, S. I., Shaikh, Z. A., Hammad, S. A., (2013), "A framework for scalable distributed provenance storage system", *Computer Standards & Interfaces*, v. 35, n. 1 (jan.), p. 179–186.

Ailamaki, A., (2011), "Managing scientific data: lessons, challenges, and opportunities". In: *SIGMOD*, p. 1045–1046, New York, NY, USA.

Allen, M., Chapman, A., Blaustein, B., Seligman, L., (2011), "Getting It Together: Enabling Multi-organization Provenance Exchange". In: *TaPP 2011*, Athens, Greece.

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

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

Amazon EC2, (2014). Amazon Elastic Compute Cloud. *Amazon EC2*. Disponível em: http://aws.amazon.com/ec2/. Acesso em: 1 jul 2014.

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

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

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

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.

de C. Coutinho, R., Drummond, L. M. A., Frota, Y., (2014), "Optimization of a Cloud Resource Management Problem from a Consumer Perspective", In: an Mey, D., Alexander, M., Bientinesi, P., Cannataro, M., Clauss, C., Costan, A., Kecskemeti, G., Morin, C., Ricci, L., Sahuquillo, J., Schulz, M., Scarano, V., Scott, S. L., Weidendorfer, J. [orgs.], Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Kobsa, A., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Terzopoulos, D., Tygar, D., Weikum, G. (eds), *Euro-Par 2013: Parallel Processing Workshops*, , chapter 8374, Berlin, Heidelberg: Springer Berlin Heidelberg, p. 218–227.

Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., Vo, H. T., (2006),

"VisTrails: Visualization Meets Data Management". In: *International Conference on Management of Data (SIGMOD)*, p. 745–747, New York, NY, USA.

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

Carvalho, U., (2015), *Deteccao e Tratamento de Flash Crowd em Nuvens Computacionais*, UFF Disponível em: http://www2.ic.uff.br/PosGraduacao/Teses/abs_684.pdf.

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

Costa, F., Oliveira, D. de, Mattoso, M., (2014), "Towards an Adaptive and Distributed Architecture for Managing Workflow Provenance Data". , p. 79–82

Costa, F., de Oliveira, D., Ocaña, K. A. C. S., Ogasawara, E., Mattoso, M., (2012), "Enabling Re-executions of Parallel Scientific Workflows Using Runtime Provenance Data". In: *Proceedings of the 4th International Conference on Provenance and Annotation of Data and Processes*, p. 229–232, Berlin, Heidelberg.

Costa, F., Silva, V., de Oliveira, D., Ocaña, K., Ogasawara, E., Dias, J., Mattoso, M., (2013), "Capturing and Querying Workflow Runtime Provenance with PROV: A Practical Approach". In: *EDBT/ICDT Workshops*, p. 282–289, New York, NY, USA.

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

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

Davidson, S. B., Freire, J., (2008), "Provenance and Scientific Workflows: Challenges and Opportunities". In: *ACM SIGMOD*, p. 1345–1350, New York, NY, USA.

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

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

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

Deelman, E., Mehta, G., Singh, G., Su, M.-H., Vahi, K., (2007), "Pegasus: Mapping Large-Scale Workflows to Distributed Resources", In: Taylor, I. J., Deelman, E., Gannon, D. B., Shields, M. [orgs.] (eds), *Workflows for e-Science*, Springer, p. 376–394.

Dias, J., Ogasawara, E., Oliveira, D., Porto, F., Coutinho, A., Mattoso, M., (2011), "Supporting Dynamic Parameter Sweep in Adaptive and User-Steered Workflow". In:

6th Workshop on Workflows in Support of Large-Scale Science, p. 31–36, Seattle, WA, USA.

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

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

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

Fowler, M., (2004), *UML distilled: a brief guide to the standard object modeling language*. Boston, Addison-Wesley.

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

Gehani, A., Kim, M., Malik, T., (2010), "Efficient querying of distributed provenance stores". , p. 613

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

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

Goncalez, T. T., Sabino, E. C., Capuani, L., Liu, J., Wright, D. J., Walsh, J. H., Ferreira, J. E., Chamone, D. A., Busch, M. P., Custer, B., (REDS-II), for the N. R. E. D. S.-I., Component, I., (2011), "Blood transfusion utilization and recipient survival at Hospital das Clinicas in São Paulo, Brazil", *Transfusion*, p. no–no.

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

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

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

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

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

Guerra, G., Rochinha, F., Elias, R., Coutinho, A., Braganholo, V., Oliveira, D. de, Ogasawara, E., Chirigati, F., Mattoso, M., (2009), "Scientific Workflow Management System Applied to Uncertainty Quantification in Large Eddy Simulation". In:

Congresso Ibero Americano de Métodos Computacionais em Engenharia, p. 1–13, Búzios, Rio de Janeiro, Brazil.

Hartman, A. L., Riddle, S., McPhillips, T., Ludäscher, B., Eisen, J. A., (2010), "Introducing W.A.T.E.R.S.: a Workflow for the Alignment, Taxonomy, and Ecology of Ribosomal Sequences", *BMC Bioinformatics*, v. 11, n. 1, p. 317.

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

http://postgres-xc.sourceforge.net/docs/1_1/index.htmlERRO. Postgres-XC 1.1 Documentation.

http://www.microsoft.com/en-us/server-cloud/products/sql-server/ERRO. Explore SQL Server 2012-2014 | Microsoft.

http://www.nuodb.com/ERRO. NewSQL | Cloud Database | Distributed Database | Scale-Out | NuoDB.

http://www.pgpool.net/docs/latest/pgpool-en.htmlERRO. pgpool-II User Manual.

http://www.postgresql.org/ERRO. PostgreSQL: The world's most advanced open source database.

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

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

Juristo, N., (2001), *Basics of Software Engineering Experimentation*. 2001 edition ed. Englewood Cliffs, N.J., Springer.

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

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

Kimball, R., Margy Ross, (2002), *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*.

Lee, Y., Oh, C., Park, E. K., (2002), "Intelligent knowledge discovery in peer-to-peer file sharing". In: *Proceedings of the eleventh international conference on Information and knowledge management*, p. 308–315, New York, NY, USA.

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

Lin, Y.-C., Li, L.-S., (2007), "NEMO based P2P file discovery scheme". In: *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, p. 145–150, New York, NY, USA.

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

Ludascher, B., Missier, P., Belhajjame, K., Chirigati, F., Wei, Y., Cuevas-Vicenttín, V., Dey, S., Kianmajd, P., Koop, D., Bowers, S., Altintas, I., (2014). The ProvONE Data Model for Scientific Workflow Provenance. Disponível em: http://vcvcomputing.com/provone/provone.html. Acesso em: 1 ago 2016.

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

Martinho, W., Ogasawara, E., Oliveira, D., Chirigati, F., Santos, I., Travassos, G. H. T., Mattoso, M., (2009), "A conception process for abstract workflows: an example on deep water oil exploitation domain", *5th IEEE International Conference on e-Science*

Mattoso, M., Werner, C., Travassos, G., Braganholo, V., Murta, L., (2008), "Gerenciando Experimentos Científicos em Larga Escala". In: *SEMISH - CSBC*, p. 121–135, Belém, Pará - Brasil.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Ogasawara, E., Oliveira, D., Cruz, S., Martinho, W., Murta, L., (2010), "Towards supporting the life cycle of large scale scientific experiments", *International Journal of Business Process Integration and Management*, v. 5, n. 1, p. 79–92.

Medeiros, C. B., Perez-Alcazar, J., Digiampietri, L., G. Z. Pastorello, J., Santanche, A., Torres, R. S., Madeira, E., Bacarin, E., (2005), "WOODSS and the Web: annotating and reusing scientific workflows", *SIGMOD Record*, v. 34, n. 3, p. 18–23.

Moreau, L., Freire, J., Futrelle, J., McGrath, R. E., Myers, J., Paulson, P., (2008), "The Open Provenance Model: An Overview". In: *International Provenance and Annotation Workshop (IPAW)*, p. 323–326

Moreau, L., Missier, P., (2013), "The PROV Data Model and Abstract Syntax Notation", *W3C Recommendation*

Muniswamy-Reddy, K.-K., Braun, U., Holland, D. A., Macko, P., Maclean, D., Margo, D. W., Seltzer, M. I., Smogor, R., (2009), "Layering in Provenance Systems.". In: *USENIX Annual technical conference*

Muniswamy-Reddy, K.-K., Holland, D. A., Braun, U., Seltzer, M. I., (2006), "Provenance-Aware Storage Systems.". In: *USENIX Annual Technical Conference, General Track*, p. 43–56

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

Ocaña, K. A. C. S., Oliveira, D., Dias, J., Ogasawara, E., Mattoso, M., (2011a), "Optimizing Phylogenetic Analysis Using SciHmm Cloud-based Scientific Workflow". In: *Proceedings of the 7th IEEE International Conference on e-Science (e-Science)*, p. 190–197, Stockholm, Sweden.

Ocaña, K., Oliveira, D. de, Ogasawara, E., Dávila, A., Lima, A., Mattoso, M., (2011b), "SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes". In: *Advances in Bioinformatics and Computational Biology*, p. 66–70

Ogasawara, E., Dias, J., Oliveira, D., Porto, F., Valduriez, P., Mattoso, M., (2011), "An Algebraic Approach for Data-Centric Scientific Workflows", *Proceedings of the International Conference on Very Large Data Bases (PVLDB)*, v. 4, n. 12, p. 1328–1339.

Ogasawara, E., Dias, J., Silva, V., Chirigati, F., Oliveira, D., Porto, F., Valduriez, P., Mattoso, M., (2013), "Chiron: A Parallel Engine for Algebraic Scientific Workflows", *CCPE*, v. 25, n. 16, p. 2327–2341.

Ohnishi, K., Yoshida, K., Oie, Y., (2007), "P2P file sharing networks allowing participants to freely assign structured meta-data to files". In: *Proceedings of the 2nd international conference on Scalable information systems*, p. 5:1–5:8, ICST, Brussels, Belgium, Belgium.

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., Baião, F. A., Mattoso, M., (2010a), "Towards a Taxonomy for Cloud Computing from an e-Science Perspective", In: Antonopoulos, N., Gillam, L. [orgs.] (eds), *Cloud Computing*, London: Springer London, p. 47–62.

Oliveira, D., Cunha, L., Tomaz, L., Pereira, V., Mattoso, M., (2009), "Using Ontologies to Support Deep Water Oil Exploration Scientific Workflows". In: *IEEE International Workshop on Scientific Workflows*, p. 364–367, Los Angeles, California, United States.

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

de Oliveira, D., Ocana, K., Ogasawara, E., Dias, J., Baiao, F., Mattoso, M., (2011), "A performance evaluation of x-ray crystallography scientific workflow using scicumulus". In: *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, p. 708–715

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

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

Oliveira, D., Ogasawara, E., Baião, F., Mattoso, M., (2010c), "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., Ocaña, K., Baião, F., Mattoso, M., (2012c), "An adaptive parallel execution strategy for cloud-based scientific workflows", *Concurrency and Computation: Practice and Experience*, v. 24, n. 13 (set.), p. 1531–1550.

Oliveira, W., Oliveira, D., Braganholo, V., (2014), "Experiencing PROV-Wf for Provenance Interoperability in SWfMSs". In: *IPAW*, p. 294–296, Cologne, German.

Özsu, M. T., Valduriez, P., (2011), *Principles of Distributed Database Systems*. 3 ed. New York, Springer.

Patavino, G. M., de Almeida-Neto, C., Liu, J., Wright, D. J., Mendrone-Junior, A., Ferreira, M. I. L., de Freitas Carneiro, A. B., Custer, B., Ferreira, J. E., Busch, M. P., Sabino, E. C., for the NHLBI Retrovirus Epidemiology Study-II (REDS-II), I. C., (2012), "Number of recent sexual partners among blood donors in Brazil: associations with donor demographics, donation characteristics, and infectious disease markers", *Transfusion*, v. 52, n. 1, p. 151–159.

Pereira, G. C., Ebecken, N. F. F., (2011), "Combining in situ flow cytometry and artificial neural networks for aquatic systems monitoring", *Expert Systems with Applications*, v. 38, n. 8, p. 9626–9632.

Porto, F., Moura, A. M., Silva, F. C., Bassini, A., Palazzi, D. C., Poltosi, M., Castro, L. E. V., Cameron, L. C., (2011), "A metaphoric trajectory data warehouse for Olympic athlete follow-up", *Concurrency and Computation: Practice and Experience*

Sabino, E. C., Gonçalez, T. T., Carneiro-Proietti, A. B., Sarr, M., Ferreira, J. E., Sampaio, D. A., Salles, N. A., Wright, D. J., Custer, B., Busch, M., for the NHLBI Retrovirus Epidemiology Donor Study-II (REDS-II), I. C., (2011), "Human immunodeficiency virus prevalence, incidence, and residual risk of transmission by transfusions at Retrovirus Epidemiology Donor Study-II blood centers in Brazil", *Transfusion*, p. no–no.

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

Souza, R. F. S., Mattoso, M. L. de Q., (2015), *CONTROLLING THE PARALLEL EXECUTION OF WORKFLOWS RELYING ON A DISTRIBUTED DATABASE*, UFRJ/COPPE

Stavrou, A., Rubenstein, D., Sahu, S., (2004), "A lightweight, robust P2P system to handle flash crowds", *IEEE Journal on Selected Areas in Communications*, v. 22, n. 1 (jan.), p. 6–17.

Taylor, I. J., Deelman, E., Gannon, D. B., Shields, M., (2007a), *Workflows for e-Science: Scientific Workflows for Grids*. 1 ed. Springer.

Taylor, I., Shields, M., Wang, I., Harrison, A., (2007b), "The Triana Workflow Environment: Architecture and Applications", *Workflows for e-Science*, Springer, p. 320–339.

Travassos, G. H., Barros, M. O., (2003), "Contributions of In Virtuo and In Silico Experiments for the Future of Empirical Studies in Software Engineering". In: *2nd Workshop on Empirical Software Engineering the Future of Empirical Studies in Software Engineering*, p. 117–130, Rome, Italy.

Vaisman, A., Zimányi, E., (2014), *Data Warehouse Systems*. Berlin, Heidelberg, Springer Berlin Heidelberg.

Valerio, M. D., Sahoo, S. S., Barga, R. S., Jackson, J. J., (2008), "Capturing Workflow Event Data for Monitoring, Performance Analysis, and Management of Scientific Workflows". , p. 626–633

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

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

p. 825–830, Los Alamitos, CA, USA.

Watanabe, S., (1960), "Information theoretical analysis of multivariate correlation", *IBM Journal of research and development*, v. 4, n. 1, p. 66–82.

Wendell, P., Freedman, M. J., (2011), "Going Viral: Flash Crowds in an Open CDN", *ACM SIGCOMM Conference on Internet Measurement (IMC) New York, NY, USA*, p. 549–558.

WfMC, I., (1997), "The WfMC glossary", p. 385–421.

Wozniak, J. M., Armstrong, T. G., Wilde, M., Katz, D. S., Lusk, E., Foster, I. T., (2013), "Swift/T: Large-Scale Application Composition via Distributed-Memory Dataflow Processing". In: *CCGrid*, p. 95–102

Zhao, D., Shou, C., Maliky, T., Raicu, I., (2013), "Distributed data provenance for large-scale data-intensive computing". In: *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, p. 1–8

Zhao, Y., Hategan, M., Clifford, B., Foster, I., von Laszewski, G., Nefedova, V., Raicu, I., Stef-Praun, T., Wilde, M., (2007), "Swift: Fast, Reliable, Loosely Coupled Parallel Computation". In: *3rd IEEE World Congress on Services*, p. 206, 199, Salt Lake City, USA.

Zhou, W., Mapara, S., Ren, Y., Li, Y., Haeberlen, A., Ives, Z., Loo, B. T., Sherr, M., (2012), "Distributed time-aware provenance", *Proceedings of the VLDB Endowment*, v. 6, n. 2 (dez.), p. 49–60.