PROVDOOP: CAPTURA, ARMAZENAMENTO E DISPONIBILIZAÇÃO DE DADOS DE PROVENIÊNCIA EM TEMPO DE EXECUÇÃO DE SISTEMAS SOBRE HADOOP

Vanessa Marques de Assis

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

Orientadoras: Marta Lima de Queirós Mattoso
Kary Ann del Carmen Ocaña Gautherot

Rio de Janeiro
Março de 2016

PROVDOOP: CAPTURA, ARMAZENAMENTO E DISPONIBILIZAÇÃO DE DADOS DE PROVENIÊNCIA EM TEMPO DE EXECUÇÃO DE SISTEMAS SOBRE HADOOP

Vanessa Marques de Assis

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

Examinada por:

_____

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

_____

Prof. Kary Ann del Carmen Ocaña Gautherot, D.Sc.

_____

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

_____

Prof. Alexandre Gonçalves Evsukoff, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2016

Assis, Vanessa Marques de

ProvDoop: captura, armazenamento e disponibilização de dados de proveniência em tempo de execução de sistemas sobre Hadoop / Vanessa Marques de Assis. – Rio de Janeiro: UFRJ/COPPE, 2016.

XI, 70 p.: il.; 29,7 cm.

Orientadoras: Marta Lima de Queirós Mattoso

Kary Ann del Carmen Ocaña Gautherot

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2016.

Referências Bibliográficas: p. 57-70.

1. *Workflows* Científicos. 2. Computação em Nuvem. 3. Proveniência de Dados. 4. Hadoop. I. Mattoso, Marta Lima de Queirós *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Aos meus pais, amigos e familiares,

por tudo que representam em minha vida.

Agradecimentos

Primeiramente e acima de tudo agradeço às pessoas mais importantes da minha vida: meus pais Leila e Francisco (*in memoriam*). À minha mãe por ser meu maior exemplo de vida e meu porto seguro. Ao meu pai (*in memoriam*), por se manter sempre presente em meu coração, e continuar me apoiando nas boas lembranças, nas lições ensinadas e no amor que cultivou.

Às minhas orientadoras, sem as quais esta dissertação não existiria. À professora Marta Mattoso pelas aulas que me motivaram desde a graduação, pela oportunidade de ser orientanda de uma profissional tão inspiradora e pela orientação sempre precisa e eficiente. À professora Kary Ocaña pelo enorme apoio, atenção e dedicação, pelas revisões rápidas e precisas e pelas valiosas orientações que conduziram a escrita desta dissertação.

Aos meus familiares: meus irmãos, meus tios, sobretudo minha amada tia Rosana que sempre esteve ao meu lado, meus primos e meus sobrinhos por sempre fazerem parte da minha torcida.

Aos meus amigos incríveis que estiveram ao meu lado durante toda a minha jornada e continuam a me apoiar e a celebrar comigo em cada uma das minhas conquistas.

Ao meu "muito mais que amigo e namorado" Diego, por ser o meu maior companheiro de vida e por estar ao meu lado mesmo nos momentos mais difíceis, levantando o meu astral, me lembrando do meu potencial e me motivando a seguir adiante.

Aos professores Alexandre Assis e Alexandre Evsukoff pelas aulas inspiradoras e por aceitarem fazer parte da banca do meu mestrado.

Aos meus colegas da Kendoo pela compreensão e pelo enorme apoio, sobretudo nessa reta final.

Ao CNPq, pela concessão de bolsa de mestrado, e à Amazon Web Services, pelos créditos concedidos que me permitiram conduzir os experimentos realizados.

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

Agradeço.

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

PROVDOOP: CAPTURA, ARMAZENAMENTO E DISPONIBILIZAÇÃO DE DADOS DE PROVENIÊNCIA EM TEMPO DE EXECUÇÃO DE SISTEMAS SOBRE HADOOP

Vanessa Marques de Assis

Março/2016

Orientadoras: Marta Lima de Queirós Mattoso

Kary Ann del Carmen Ocaña Gautherot

Programa: Engenharia de Sistemas e Computação

Grande parte dos experimentos científicos em larga escala envolvem execuções de *workflows* computacionalmente intensivos e de longa duração. É fundamental que existam meios para o cientista acompanhar essas execuções enquanto elas estão em andamento. Ao acompanhar o andamento da execução de um *workflow* científico, pode-se tomar decisões e avaliar os resultados em tempo real. Isso permite que o cientista detecte erros mais rapidamente, economizando tempo e recursos financeiros, sobretudo considerando um ambiente como o de nuvem, que possui uma política *pay-per-use*. Nesta dissertação, apresentamos uma solução não intrusiva para execução de *workflows* científicos em nuvens computacionais utilizando Hadoop. A arquitetura proposta, intitulada ProvDoop, captura, armazena e disponibiliza dados de proveniência em tempo de execução para execuções com Hadoop. Experimentos realizados com o ProvDoop usando um caso de estudo real da área de bioinformática, mostram que a utilização do ProvDoop não causa impactos significativos no tempo total da execução dos *workflows* científicos frente aos ganhos que a proveniência em tempo real proporciona. Observou-se que variações no ambiente, como latência da rede, podem causar mais impactos no tempo de execução do que a utilização do ProvDoop, que apresentou melhora de até 4% no tempo total de execução para alguns cenários observados. O ProvDoop viabiliza o acesso aos dados de forma que consultas a um repositório de proveniência possam ser realizadas em tempo de execução, permitindo que o cientista acompanhe as execuções de seus experimentos em tempo real.

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

PROVDOOP: GATHERING, STORING AND PROVIDING RUNTIME PROVENANCE DATA FOR HADOOP BASED SYSTEMS

Vanessa Marques de Assis

March/2016

Advisors: Marta Lima de Queirós Mattoso

Kary Ann del Carmen Ocaña Gautherot

Department: Systems and Computer Engineering

Most of large-scale scientific experiments involve several executions of computationally intensive and long duration scientific workflows. It is vital that there are means for the scientist to monitor those executions while they are still in progress. By monitoring the progress of a scientific workflow execution, it is possible to make decisions and evaluate the results in real time. This allows the scientist to detect errors quicker, saving time and financial resources, especially if we consider an environment like the cloud that has a pay-per-use policy. In this dissertation, we present a non-intrusive solution for scientific workflows executions on computer clouds using Hadoop. The proposed architecture, called ProvDoop, gathers, stores and provides provenance data at run time for executions with Hadoop. Experiments performed with ProvDoop using a real case study from bioinformatics field showed that the use of ProvDoop does not cause a significant impact on the total execution time of scientific workflows, specially considering the gains that real-time provenance provides. It was observed that environment variations, like network latency, can cause more impacts on the execution time than the use of ProvDoop, that showed an improvement up to 4% on the total execution time for some scenarios. ProvDoop provides the access to data in a way that queries can be run on a provenance repository at run time, allowing the scientists to monitor their experiments executions in real time.

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1 - Introdução

Experimento científico consiste na montagem de um protocolo concreto a partir do qual se organizam diversas ações observáveis de forma a corroborar ou refutar uma dada hipótese científica que foca em estabelecer relações entre fenômenos (Beveridge 2004, Jr 2002). A evolução da ciência da computação nas últimas décadas permitiu a exploração de experimentos científicos baseados em simulação (Deelman *et al.* 2009).

Desta forma, os experimentos científicos passaram a depender de recursos computacionais e tecnologias especializadas para simular fenômenos do mundo real em ambientes virtuais. Essas simulações são baseadas em modelos computacionais complexos que normalmente são representados por um conjunto de programas utilizados durante as simulações que podem consumir um grande volume de dados.

Diante da potencial complexidade dos modelos computacionais utilizados e do volume de dados a ser consumido, surgiram técnicas e abordagens para apoiar a execução de experimentos científicos, como os *workflows* científicos (Deelman *et al.* 2009, Taylor *et al.* 2007a). Os *workflows* científicos permitem modelar e executar experimentos científicos através de simulações computacionais de larga escala. Um *workflow* pode ser definido como uma abstração que permite a composição de diversas atividades em um fluxo estruturado (Taylor *et al.* 2007a), onde cada atividade corresponde à invocação de um programa e as dependências representam o fluxo de dados entre as atividades envolvidas na execução.

Em alguns casos, esses *workflows* científicos podem ser compostos por milhares de atividades que por sua vez podem processar milhares de entradas. Além disso, podem ser necessárias diversas de execuções para que um cientista finalize seu experimento, gerando um cenário em que ocorrem centenas de milhares de execuções de atividades, gerando uma enorme quantidade de dados.

Baseado nessas e em outras aplicações que processam um grande volume de dados, surgiu o conceito denominado "*Big Data*" (Bertino *et al*. 2011). Esse termo faz menção não apenas ao volume de dados, mas também à sua variedade e acurácia e a velocidade necessária para o seu processamento (Borkar *et al.* 2012). Como existem diversos recursos geradores para esses dados, surge uma enorme variedade de formatos, alguns estruturados e outros não estruturados (Goldman *et al.* 2012). Entende-se como dados

não estruturados aqueles que não seguem um modelo rígido como o de um sistema de banco de dados, por exemplo, o modelo relacional (Heradotou *et al.* 2011).

As aplicações "*Big Data*" fazem da computação o mecanismo para criar soluções capazes de analisar grandes bases de dados, processar seus pesados cálculos, identificar comportamentos e disponibilizar serviços especializados em seus domínios, porém, quase sempre esbarram no poder computacional das máquinas atuais (Goldman *et al.* 2012). Execuções chegam a consumir horas ou dias de processamento nas arquiteturas convencionais.

Diante desse cenário, a utilização de ambientes de processamento de alto desempenho (PAD) se tornou uma necessidade real, pois esses ambientes permitem aumentar o poder computacional envolvido nas execuções a fim de impedir que o tempo de execução seja uma barreira para a realização dos experimentos científicos. Como exemplos destes ambientes, podemos citar os *clusters* (Dantas 2005a), as grades computacionais (Dantas 2005b, Foster e Kesselman 2004), os ambientes de computação voluntária (Anderson *et al.* 2002, Cirne *et al.* 2006, Beberg *et al.* 2009,), as redes ponto a ponto, do inglês *peer-to-peer* ou P2P, (Valduriez e Pacitti 2005) e mais recentemente as nuvens computacionais (D. Oliveira, Baião e Mattoso 2010, Vaquero *et al.* 2009).

As nuvens computacionais vêm se mostrando um ambiente alternativo e viável para muitos tipos de aplicações (Hashem *et al.* 2015) à medida que oferecem *hardware* e *software* de forma virtualizada e sob demanda para os usuários, que são cobrados apenas pelos recursos que foram de fato utilizados, eliminando a necessidade de custos com recursos de *hardware* e *software* dedicados e permitindo a construção de soluções escaláveis e adaptáveis de maneira rápida e eficiente.

No entanto, realizar execuções distribuídas em ambientes de alto desempenho, como as nuvens computacionais, não é uma tarefa trivial pois é necessário dividir o trabalho em diversas unidades de processamento, levando em consideração preocupações como dependência entre os dados, balanceamento de carga e escalonamento de tarefas para um uso eficiente dos recursos computacionais, além da recuperação de falhas caso um ou mais nós envolvidos na execução falhem.

Foi nesse contexto que foi desenvolvido o Apache Hadoop, um arcabouço para o processamento de grandes quantidades de dados em ambientes distribuídos. Nesse

arcabouço, problemas como integridade dos dados, disponibilidade dos nós, escalabilidade da aplicação e recuperação de falhas ocorrem de forma transparente ao usuário (White 2012). Além disto, seu modelo de programação funcional e paralelo e seu sistema de armazenamento dos dados promovem um rápido processamento.

O arcabouço Apache Hadoop se consolidou no ambiente empresarial e também tem obtido apoio da comunidade acadêmica, sendo utilizado para a execução de *workflows* científicos (Wang *et al*. 2009, Islam *et al*. 2012, Dias *et al*. 2013, Islam e Srinivasan 2015, Dalman *et al*. 2015, Bux *et al*. 2015, Liu *et al*. 2015, Gugnani e Kiss 2015).

A execução, a especificação e o monitoramento de *workflows* científicos são normalmente realizados por sistemas chamados de Sistemas de Gerência de *Workflows* Científicos (SGWfC) (Altintas *et al.* 2004, Callahan *et al.* 2006) que possuem um motor de execução acoplado, responsável por invocar cada um dos programas envolvidos no fluxo.

Um dos principais diferenciais dos SGWfC em relação ao Hadoop é a gerência da dependência do fluxo de dados do *workflow* sendo executado (Silva *et al.* 2014). Essa gerência de fluxo de dados é realizada pelos SGWfC através da captura e armazenamento de dados de proveniência (Freire *et al.* 2008) produzidos pela execução do *workflow*.

Por dados de proveniência pode-se entender todos os dados que são armazenados com o objetivo de descrever determinado experimento. Por exemplo, os dados de entrada, a configuração do ambiente e o tempo de execução de cada atividade. A proveniência é fundamental para a confiabilidade e reprodutibilidade de um experimento científico, assim como para a detecção e correção de erros na execução de um *workflow* científico (Davidson *et al.* 2008).

Tentativas iniciais em associar proveniência ao Hadoop foram realizadas de maneira preliminar e intrusiva (Glavic 2015). Trabalhos publicados anteriormente (Amsterdamer *et al*. 2011, Park *et al*. 2011, Crawl *et al*. 2011, Ikeda *et al.* 2011, Oliveira *et al.* 2013, Akoush *et al.* 2013) capturam dados básicos de proveniência durante a execução do Hadoop, porém, só permitem que os dados de proveniência sejam consultados ao fim da execução.

Ao disponibilizar dados de proveniência somente ao fim da execução, a análise dos experimentos se torna ineficiente, sobretudo para experimentos em larga escala, *i.e.* que executam por diversas horas, dias ou até semanas.

Dados de proveniência disponíveis em tempo de execução ajudam o cientista à medida que permitem que o mesmo tome decisões e realize ações avaliando os dados gerados pela aplicação em tempo real. Por meio da análise dos dados de proveniência pode-se re-executar uma atividade com problema tão logo se detecte a existência do mesmo, evitando assim que uma série de erros ocorram na cadeia de programas. Sobretudo em um ambiente de nuvem, onde os recursos são pagos à medida que são usados, esse é um aspecto fundamental de economia de tempo e recursos financeiros para o cientista.

Existem muitos benefícios na utilização dos dados de proveniência gerados em tempo de execução, tais como, possibilidade de escalonamento adaptativo de *workflows* (Oliveira *et al*. 2011b), possíveis estratégias de supervisão (do inglês *steering*) de *workflows* (Dias *et al*. 2011) e gerenciamento das falhas das atividades com disparo de re-execuções (Costa 2012).

No entanto, não há atualmente solução que insira proveniência de dados em execuções de *workflows* científicos com Hadoop na nuvem computacional de forma não intrusiva e em tempo de execução.

Nesta dissertação apresentamos o ProvDoop, uma solução computacional que funciona em conjunto com o ecossistema do Hadoop de forma não intrusiva. O ProvDoop permite que o Hadoop realize execuções paralelas e distribuídas de *workflows* científicos, ao mesmo tempo em que realiza a coleta, armazenamento e disponibilização de dados de proveniência de maneira que eles possam ser acessados, em tempo de execução, através de consultas estruturadas a uma base de dados de proveniência.

O ProvDoop permite a disponibilização de dados de proveniência em tempo real para o cientista, garantindo um monitoramento mais preciso, eficiente e eficaz de execuções de *workflows* realizadas com Hadoop e podendo representar significativa economia de tempo e recursos financeiros.

1.1 Caracterização do Problema

De acordo com o que foi explicitado na subseção anterior, o problema que esta dissertação busca solucionar é:

"Como modelar e desenvolver uma solução não intrusiva que permita capturar, armazenar e disponibilizar dados de proveniência em execuções de workflows científicos em nuvens computacionais utilizando Hadoop, de maneira que os dados possam ser acessados, em tempo de execução, através de consultas estruturadas realizadas a uma base de dados de proveniência."

1.2 Hipótese

A hipótese geral desta dissertação é que é possível capturar, armazenar e disponibilizar dados de proveniência para execuções com *workflows* científicos em nuvens computacionais utilizando Hadoop através de soluções não intrusivas, ou seja, que não alterem o código-fonte do Hadoop. Tais soluções devem permitir que os dados possam ser acessados, em tempo de execução, através de consultas estruturadas a uma base de dados de proveniência, gerando ganhos para os cientistas em termos de reprodutibilidade de experimentos e detecção e correção de erros precocemente, economizando tempo e recursos financeiros.

Desse modo, o objetivo principal desta dissertação é propor tal solução de forma que os possíveis impactos gerados no tempo de execução do *workflow* científico sejam pequenos frente aos ganhos gerados pela disponibilização de dados de proveniência em tempo de execução.

1.3 ProvDoop

O ProvDoop é uma solução computacional, acoplada ao ecossistema do Hadoop de maneira não intrusiva, que captura, armazena e disponibiliza em tempo de execução dados de proveniência de execuções de *workflows* científicos realizados com Hadoop. O ProvDoop foi desenvolvido de forma a gerenciar as execuções com Hadoop e os dados de proveniência capturados mesmo em ambientes paralelos e distribuídos, como é a nuvem computacional.

Com a proveniência coletada pelo ProvDoop é possível obter, em tempo de execução, informações sobre os arquivos utilizados e gerados pelo experimento, as máquinas utilizadas na execução, as atividades executadas juntamente com detalhes sobre cada uma delas, dentre outras informações.

A arquitetura proposta pelo ProvDoop cobre não só a execução do *workflow* científico com Hadoop, mas também a gerência dos arquivos de entrada e saída em um sistema de arquivos compartilhado. Além disso, o ProvDoop realiza a coleta, armazenamento e disponibilização dos dados de proveniência em um repositório acessível em tempo de execução para o cientista através de consultas estruturadas a um banco de dados relacional.

1.4 Organização da Dissertação

Além desta introdução, esta dissertação é organizada como segue. O Capítulo 2 apresenta conceitos importantes para a compreensão desta dissertação, como experimentos científicos, *workflows* científicos, nuvem computacional, proveniência de dados e Hadoop. O Capítulo 3 apresenta o ProvDoop, uma solução computacional para captura, armazenamento e disponibilização em tempo de execução de dados de proveniência para execuções de *workflows* científicos no Hadoop. O Capítulo 4 apresenta as avaliações experimentais realizadas com o ProvDoop, descrevendo o experimento e o ambiente utilizado e detalhando os resultados obtidos. Por fim, o Capítulo 5 conclui esta dissertação, apresentando os resultados alcançados e os desdobramentos possíveis para trabalhos futuros.

Capítulo 2 - Fundamentação Teórica

Nas próximas seções apresentamos a fundamentação teórica de experimentos científicos em larga escala, de maneira a apresentar conceitos importantes para a compreensão desta dissertação. O Capítulo está organizado como segue. A Seção 2.1 apresenta o conceito de experimento científico baseado em simulação, que é o objeto de estudo desta dissertação. A Seção 2.2 define o ciclo de vida de um experimento científico. A Seção 2.3 caracteriza o conceito de *workflows* científicos. A Seção 2.4 apresenta o conceito de proveniência de dados no contexto de *workflows* científicos. A Seção 2.5 apresenta os conceitos de nuvem computacional, seguida pela Seção 2.6 que apresenta o ciclo de vida do *workflow* científico na nuvem. Por fim, a seção 2.7 apresenta o Hadoop.

2.1 Experimentos Científicos Baseados em Simulação

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). Por condições controladas entende-se que há esforços para eliminar, ou reduzir tanto quanto possível, os erros durante uma observação planejada (Juristo e Moreno 2001). A partir dessas definições, podemos concluir que um experimento científico está associado a um conjunto de ações controladas. Essas ações controladas incluem variações de testes, e seus resultados são geralmente comparados entre si para aceitar ou refutar uma hipótese científica. Os experimentos científicos são a maior preocupação da comunidade científica (Mattoso *et al*. 2010c).

Existem diversos tipos de experimentos científicos, são eles: *in vivo, in vitro, in virtuo* e *in silico* (Travassos e Barros 2003). Nesta dissertação, tratamos dos experimentos *in silico* ou baseados em simulação (Travassos e Barros 2003). Sendo assim, no contexto desta dissertação o termo "experimento científico" será usado para referenciar somente experimentos científicos baseados em simulação. Esse tipo de experimento é utilizado nos mais diversos domínios científicos, tais como bioinformática (Ocaña *et al*. 2011a,

Ocaña *et al*. 2011b, Lemos *et al*. 2004), estudos na área de saúde (de Almeida-Neto *et al*. 2011, Goncalez *et al*. 2011, Patavino *et al*. 2012, Sabino *et al*. 2011), prospecção de petróleo em águas profundas (Carvalho 2009, Martinho *et al*. 2009, Ogasawara *et al*. 2011, Oliveira *et al*. 2009a), mapeamento dos corpos celestes (Hey *et al*. 2012), ecologia (Hartman *et al*. 2010), agricultura (Fileto *et al*. 2003), busca de genes ortólogos dos tripanosomas causadores de doenças tropicais negligenciadas (Coutinho *et al*. 2011, Dávila *et al*. 2008), dinâmica de fluidos computacional (Guerra e Rochinha 2009, Guerra *et al*. 2009, 2012, Lins *et al*. 2009), estudos fisiológicos (Porto *et al*. 2011), previsão de precipitação (Evsukoff *et al*. 2011), monitoramento aquático (Pereira e Ebecken 2011) e pesquisa sobre energia escura (Governato *et al*. 2010). Todos esses exemplos podem ser considerados de larga escala por consumirem e produzirem um grande volume de dados.

O desenvolvimento destes tipos de experimentos, todos considerados de larga escala, requer muitos recursos computacionais e tempo de processamento, o que torna necessária a utilização de um ambiente de Processamento de Alto Desempenho (PAD). São considerados ambientes de PAD os *clusters* e supercomputadores, grades computacionais, ambientes de computação voluntária e, mais recentemente, as nuvens de computadores.

Os experimentos científicos baseados em simulação são criados a partir de modelos computacionais complexos que normalmente são representados por um conjunto de programas utilizados durante as simulações. Esses programas devem ser executados de forma encadeada, produzindo e consumindo uma grande quantidade de dados. Cada programa pode ser executado consumindo um grupo específico de parâmetros e dados, cada qual com sua própria semântica e sintaxe. A saída de um programa é normalmente utilizada como entrada para outro programa (Cavalcanti *et* al. 2005).

Diante da potencial complexidade dos modelos computacionais utilizados e do volume de dados a ser consumido, surgiu a necessidade de técnicas e abordagens para apoiar a execução de experimentos científicos. Os *workflows* científicos apresentam uma estratégia interessante para apoiar essas execuções através do controle do encadeamento de programas e podem ser definidos como uma abstração para modelar o fluxo de atividades e de dados em um experimento. Em *workflows* científicos, essas atividades são geralmente programas ou serviços que representam algoritmos e métodos computacionais sólidos (Barker e van Hemert 2008).

Esses *workflows* são controlados e executados pelos Sistemas de Gerência de Workflows Científicos (SGWfC), que são mecanismos complexos que visam apoiar a configuração e execução dos workflows. Há muitos SGWfC disponíveis, como o VisTrails (Callahan *et al*. 2006), o Taverna (Missier *et al*. 2010), o Swift/T (Wozniak *et al*. 2013), o Kepler (Ludascher *et al*. 2006), o Pegasus (Deelman *et al*. 2007), o Chiron (Ogasawara *et al*. 2013) e o Galaxy (Goecks *et al*. 2010), cada um com suas próprias características, vantagens e desvantagens.

Nessa dissertação, o conceito de experimento científico engloba o conceito de *workflow*, e não podem ser tratados como sinônimos. 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 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 de experimentos científicos, tratado com detalhes na próxima seção, envolve várias fases, sendo uma delas a execução dos ensaios.

2.2 Ciclo de Vida do Experimento Científico

Nesta seção será apresentado um modelo de ciclo de vida para um experimento cientifico proposto em Mattoso *et al*. (2010).

A Figura 1 apresenta o ciclo de vida de um experimento científico. É possível observar na figura a existência de algumas etapas que são percorridas diversas vezes pelos cientistas ao longo do seu experimento científico, de acordo com as fases existentes, que são as seguintes: execução, composição e análise. Cada fase possui um subciclo independente que pode ser percorrido em momentos distintos do experimento.

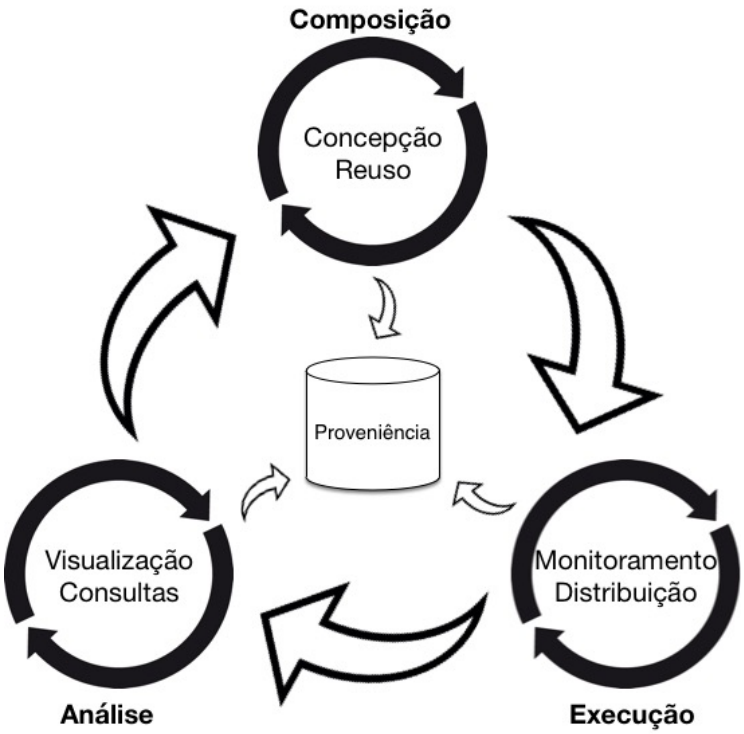

Figura 1 Ciclo de Vida do Experimento Científico Adaptado de Mattoso *et al.* (2010)

A fase de composição tem o objetivo de criar e estruturar o experimento científico, definindo uma sequência lógica de atividades e os tipos de entrada e saída de cada uma delas. Essa fase pode se decompor em outras duas subfases: concepção e reuso. Enquanto a concepção tem por finalidade a criação do experimento, o reuso tem por propósito recuperar um experimento existente e adaptá-lo para uma nova finalidade.

Já na fase de execução, as especificações do *workflow* se tornam concretas e executáveis permitindo que o mesmo seja executado por um SGWfC. A fase de execução possui uma maior dependência computacional e por isso é a fase em que maiores esforços costumam ser concentrados. Essa fase também se subdivide em outras duas: distribuição e monitoramento. A subfase de distribuição está relacionada com a necessidade da execução de atividades do workflow em ambientes de PAD, principalmente devido a necessidades de desempenho. A subfase de monitoramento está relacionada à necessidade de verificar periodicamente o estado atual da execução do *workflow*, uma vez que este pode executar por longos períodos.

Por fim, a fase de análise tem como objetivo principal estudar os dados gerados nas etapas de composição e execução. Esta fase é altamente dependente dos dados de

proveniência (Freire *et al.* 2008) que foram gerados nas fases anteriores. Esta fase também pode ser decomposta em duas subfases: visualização e consulta.

Na fase de análise, os cientistas podem se deparar com a situação em que a hipótese analisada é provavelmente correta ou que a hipótese é refutada. Em ambos os casos, os cientistas precisam executar novamente o *workflow* a fim de validar efetivamente a hipótese ou criar uma nova. Todas essas execuções, mesmo que com diferentes parâmetros e conjuntos de dados, devem ser associada a um mesmo experimento científico.

2.3 *Workflows* Científicos

À medida que um experimento se torna cada vez mais complexo, seja pela quantidade de execuções ou pelo volume de dados a ser processado, gerenciar tais simulações torna-se um desafio. Como solução, os *workflows* científicos são usados para apoiar a gerência de recursos envolvidos em simulações computacionais de larga escala.

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

Atividades são os programas envolvidos na execução de um *workflow* cientifico que são responsáveis por consumir dados de entrada e produzir dados de saída. Uma ativação é a menor unidade de dados autocontida para a execução de uma atividade, sendo composta de um conjunto de valores para realizar o processamento necessário da atividade. Para isso, uma ativação é definida em função de três procedimentos: instrumentação dos dados de entrada (substituições de identificadores de atributos por valores específicos), invocação do programa associado à atividade e a extração dos dados para a relação de saída.

2.4 Proveniência de Dados

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

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

E é a partir da análise desse histórico, ou seja, da sequência de passos que levou a um determinado resultado, é que se pode avaliar os procedimentos utilizados, analisar os parâmetros de entrada e garantir a reprodutibilidade dos resultados obtidos.

Goble *et al.* (2003) resumem as diversas funções 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.
(iv) Informacional: permite realizar consultas baseadas nos descritores de origem para a descoberta de dados, além de prover o contexto necessário para interpretar os mesmos.

De maneira geral, a proveniência pode ser caracterizada em duas formas: prospectiva e retrospectiva (Cruz *et al.* 2009, Freire *et al.* 2008). A proveniência prospectiva captura a especificação do que levou à geração de um determinado produto. No contexto de *workflows* científicos, isso representa os dados relativos à estrutura do *workflow*, bem como as configurações de ambiente utilizadas para executá-lo. Já a proveniência retrospectiva foca em capturar dados, juntamente com os descritores gerados a partir da execução de um determinado processo, ou seja, de um determinado *workflow*

científico. Entre os dados de proveniência que englobam a proveniência retrospectiva estão: tempos de início e fim de execução, arquivos produzidos, erros que ocorreram, informações de desempenho de atividades, entre outros.

Para armazenar a proveniência retrospectiva devem ser utilizados, preferencialmente, modelos de dados que se baseiem na recomendação do *Open Provenance Model* (OPM) (Moreau *et al.* 2008), ou de sua evolução: o modelo PROV (Moreau *et al*. 2011), o qual também propõe uma representação genérica de proveniência.

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

2.5 Nuvem Computacional

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

Vaquero *et al.* (2009) definem nuvem como uma grande gama de recursos virtualizados que são facilmente usáveis e acessíveis. Eles podem ser reconfigurados para se ajustar a uma demanda variável, permitindo uma utilização ótima de recursos. Esse conjunto de recursos normalmente é disponibilizado por um provedor em um modelo onde se paga somente pelo uso (do inglês *pay-per-use model*).

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

Como a própria definição apresentada indica, a computação em nuvem representa um grande potencial para apoiar experimentos científicos que necessitem de ambientes de PAD. A natureza das necessidades da computação científica se encaixa bem com a flexibilidade e a elasticidade, sob demanda, oferecida pelas nuvens. De acordo com Simmhan *et al.* (2010) o uso efetivo de nuvens pode reduzir o custo real com

equipamentos e com consequente manutenção dos mesmos, além de questões como atualizações constantes de *software* e *hardware*.

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

Em relação ao Modelo de Negócio, as abordagens são geralmente classificadas em três categorias (Mell e Grance 2011): *Software* como Serviço (do inglês *Software as a Service* ou SaaS), Plataforma como Serviço (do inglês *Platform as a Service* ou PaaS) e Infraestrutura como Serviço (do inglês *Infrastructure as a Service* ou IaaS), criando um modelo chamado SPI (do inglês *Service-Platform Infrastructure*) (Youseff *et al.* 2008, Zhu e Wang 2008).

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

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

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

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

2.6 Ciclo de Vida do *Workflow* Científico em Nuvem

Um *workflow* científico que é executado em um ambiente de nuvens de computadores segue as mesmas etapas do ciclo de vida de um experimento, porém com algumas fases adicionais. A Figura 2 representa as sete principais etapas que compõem o ciclo de vida de um *workflow* científico executado em nuvens.

Figura 2 Ciclo de Vida de um *Workflow* Científico em Nuvens de Computadores Adaptado de Oliveira (2012)

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

Na fase de configuração do ambiente é realizada a transferência de dados da máquina local para a nuvem, a criação das imagens com os programas que fazem parte do *workflow* e a criação do *cluster* virtual, onde o *workflow* será executado em paralelo. Esta fase possui um subciclo, pois a configuração do cluster virtual é uma tarefa que não termina enquanto o *workflow* não finaliza sua execução. Isso porque o tamanho do *cluster* pode aumentar ou diminuir dependendo da demanda de capacidade de processamento do *workflow*.

Na fase de execução, as tarefas do *workflow* são de fato despachadas e executadas nas diversas máquinas virtuais que foram instanciadas na fase de configuração. Após o despacho do *workflow* para execução ocorre a fase de monitoramento do *workflow*.

Na fase de análise os dados são efetivamente baixados para a máquina do cientista para enfim serem analisados. Esta fase inclui uma subfase de descoberta, onde os cientistas desenharão conclusões baseados nos dados e verificarão se a hipótese de pesquisa original foi corroborada ou refutada. No caso de ter sido refutada, o ciclo se repete novamente, mudando-se alguns parâmetros de entrada. E no caso de ter sido corroborada, o ciclo normalmente se repete a fim de validar a hipótese.

2.7 Hadoop

Um dos grandes desafios computacionais da atualidade é armazenar, manipular e analisar, de forma inteligente, a grande quantidade de dados existente. Baseado em aplicações que possuem um volume gigantesco de dados, surgiu o conceito denominado "Big Data" (Bertino *et al*. 2011). Esse termo se refere não apenas ao volume dos dados, mas também à sua variedade e velocidade necessária para o seu processamento.

Os ambientes de PAD permitem que o processamento de tais volumes de dados seja realizado, pois sem eles o tempo de processamento se torna impraticável. No entanto, dividir uma tarefa em subtarefas e então executá-las paralelamente em diversas unidades de processamento não é algo trivial. E, caso essa divisão não seja realizada adequadamente, o desempenho da execução pode ser afetado de maneira significativa. Além disso, é necessário extrair a dependência entre os dados da aplicação, determinar um algoritmo de balanceamento de carga e de escalonamento para as tarefas e garantir a recuperação da execução da aplicação caso uma máquina falhe.

Foi nesse contexto que foi desenvolvido o Apache Hadoop. O Hadoop é um arcabouço de código aberto, implementado em Java e utilizado para o processamento e armazenamento em larga escala, para alta demanda de dados. Nesse arcabouço, problemas como integridade dos dados, disponibilidade dos nós, escalabilidade da aplicação e recuperação de falhas ocorrem de forma transparente ao usuário. Os elementos chave do Hadoop são o modelo de programação MapReduce e o sistema de arquivos distribuído HDFS (White 2012).

O paradigma de programação MapReduce (Dean e Ghemawat 2008) utilizado pelo Hadoop se inspira em duas funções simples (*Map* e *Reduce*) presentes em diversas linguagens de programação funcionais.

A função *Map* recebe uma lista de tuplas <chave, valor> como entrada e, aplicando uma função dada, gera uma nova lista de tuplas como saída. A função é aplicada a todos os elementos da lista de entrada. Logo, cada iteração na lista vai gerar um elemento da lista de saída.

A função *Reduce*, similarmente à função *Map*, vai receber como entrada uma lista de tuplas e, em geral, aplicará uma função para que a entrada seja reduzida a uma única tupla na saída. Por exemplo, uma possível função *Reduce* seria uma função de média, que retorna a média dos valores da lista de entrada.

No paradigma MapReduce, as funções *Map* e *Reduce* são utilizadas em conjunto e, normalmente, as saídas produzidas pela execução das funções *Map* são utilizadas como entrada para as funções *Reduce*.

O programador é responsável por criar as funções *Map* e *Reduce* de cada aplicação. O MapReduce abstrai a complexidade do paralelismo das aplicações, pois esconde do programador a grande complexidade da distribuição e gerenciamento de dados. No entanto, é necessário que o programador consiga abstrair todo o problema em funções *Map* e *Reduce*, o que, dependendo da aplicação, pode não ser trivial.

No Hadoop, uma máquina mestre que gerencia as demais máquinas escravas. Dessa forma, o nó mestre distribui as funções desenvolvidas utilizando o paradigma MapReduce para todos os nós escravos. O nó mestre gerencia a execução de forma a garantir o balanceamento de carga e a tolerância a falhas.

O *Hadoop Distributed File System* (HDFS) é um sistema de arquivos distribuído de código aberto integrado ao arcabouço Hadoop. O HDFS oferece suporte ao armazenamento e ao processamento de grandes volumes de dados em um agrupamento de computadores heterogêneos de baixo custo. Essa quantidade de dados pode chegar à ordem de *petabytes*, quantidade que não seria possível armazenar em um sistema de arquivos tradicional.

Um sistema de arquivos distribuído possui as mesmas características que um sistema de arquivos convencional, entretanto, deve permitir o armazenamento e o compartilhamento desses arquivos em diversos *hardwares* diferentes, que normalmente estão interconectados por meio de uma rede. O HDFS divide os arquivos em uma sequência de blocos de tamanho fixo. O tamanho padrão definido no arcabouço é 64 Mb, podendo ser alterado se necessário. O HDFS também é implementado com a arquitetura mestre/escravo. Enquanto o nó mestre é o responsável por armazenar os metadados dos arquivos, os nós escravos são os responsáveis pelo armazenamento físico dos dados.

2.8 Trabalhos Relacionados

Existem vários estudos que se relacionam de alguma forma com o trabalho apresentado nesta dissertação. Serão apresentados aqueles trabalhos que forneceram conceitos importantes para o desenvolvimento desta dissertação e aqueles que se assemelham de alguma forma com a abordagem aqui apresentada. Nesse último caso, será feita a devida comparação para mostrar em quais aspectos a abordagem se diferencia da aqui apresentada.

Em Oliveira *et al*. (2013), foi desenvolvido um módulo de integração entre o VisTrails e o Hadoop, de forma a realizar a execução de *workflows* científicos utilizando o Hadoop a partir do VisTrails. O módulo desenvolvido captura dados de proveniência durante a execução, mesmo em execuções realizada em ambiente de nuvem. No entanto, diferentemente dos dados capturados pela solução proposta nesta dissertação, os dados não estão disponíveis em tempo de execução. De forma que é necessário que a execução finalize para que os dados possam ser acessados. Além disso, a solução depende de um SGWfC, o VisTrails, diferentemente da solução proposta nesta dissertação, que funciona diretamente no Hadoop.

A mesma limitação em relação a disponibilização dos dados de proveniência somente ao fim da execução pode ser encontrada em diversas outras abordagens (*e.g.* Ikeda *et al*. 2011, Amsterdamer *et al*. 2011, Park *et al*. 2011, Crawl *et al*. 2011 e Akoush *et al.* 2013). Além disso, essas mesmas abordagens tratam de soluções intrusivas, ou seja, que alteram o código fonte do Hadoop. A abordagem proposta nesta dissertação visa inserir proveniência utilizando somente as implementações *Map* e *Reduce* presentes em qualquer distribuição do Hadoop, de forma a não alterar seu código fonte.

Foge do escopo desta dissertação realizar comparações entre o desempenho de execuções de *workflows* utilizando Hadoop com execuções utilizando outras soluções existentes, como o SciCumulus (Oliveira *et al.* 2010). Estudos como este podem ser encontrados em Oliveira 2012, Oliveira *et al.* 2012, Dias *et al.* 2013 e Ferreira 2014.

Outro ponto que foge do escopo desta dissertação é a análise de ajustes finos realizados na configuração do Hadoop a fim de melhorar seu desempenho. Esse tipo de análise pode ser encontrada em Herodotou *et al.* 2011, Joshi 2012, Heger 2013, Li *et al.* 2014 e Shi *et al.* 2015.

Diferentemente do que se observa nas publicações apresentadas anteriormente, as avaliações experimentais realizadas nesta dissertação têm por objetivo comparar o desempenho observado em execuções de *workflows* científicos com Hadoop sem proveniência de dados e comparar com o desempenho obtido em execuções utilizando a solução proposta nesta dissertação. O objetivo desta comparação é avaliar se a adição de proveniência em tempo real prejudicará o desempenho da execução dos *workflows* científicos de maneira significativa, analisando assim a viabilidade da solução.

Capítulo 3 - ProvDoop

Este capítulo apresenta o ProvDoop, uma solução computacional composta por componentes e arquitetura desenvolvidos, que captura, armazena e disponibiliza dados de proveniência em tempo de execução para *workflows* científicos executados no Hadoop. A motivação para o desenvolvimento do ProvDoop foi baseada na necessidade que os cientistas enfrentam de acompanhar a execução de *workflows* científicos, sobretudo quando essas execuções duram por longos períodos. Esse acompanhamento é fundamental para que possam tomar decisões e realizar a análise dos resultados gerados de maneira mais rápida, eficiente e embasada. Isso ainda não era possível para cientistas que executam seus *workflows* com a utilização do Hadoop, já que este não apresenta qualquer mecanismo de disponibilização de dados de proveniência em tempo de execução.

Neste capítulo, passaremos por todos os passos da concepção e implementação do ProvDoop, detalhando sua arquitetura, seu modelo de proveniência e as particularidades envolvidas no seu desenvolvimento. A seção 3.1 apresenta a arquitetura conceitual do ProvDoop e o funcionamento de cada um de seus componentes. Na seção 3.2 o modelo de proveniência do ProvDoop é apresentado. Por fim, a seção 3.3 apresenta os principais aspectos do processo de implementação de cada componente.

3.1 Arquitetura do ProvDoop

O ProvDoop é uma solução computacional composta por arquitetura e componentes desenvolvidos, que permite a execução de *workflows* científicos no Hadoop ao mesmo tempo em que coleta, armazena e disponibiliza dados de proveniência em tempo de execução para o cientista. A arquitetura do ProvDoop pode ser dividida em seis componentes:

(i) **Sistema de arquivos compartilhados**: contém todos os arquivos de entrada que serão consumidos na execução do *workflow* científico e todos os arquivos de saída gerados por ele.

(ii) **Agente de execução:** responsável pela execução do *workflow* científico no Hadoop.

(iii) **Agente de captura:** responsável por capturar dados de proveniência, este componente funciona em conjunto com o agente de execução, capturando dados durante a execução do *workflow*.

(iv) **Fila de mensagens:** contém mensagens relacionadas aos dados de proveniência capturados durante a execução do *workflow* científico. A fila controla o fluxo de dados entre o agente de captura dos nós escravos e o agente de armazenamento.

(v) **Agente de armazenamento:** responsável por armazenar os dados coletados pelo agente de captura no repositório de proveniência, retirando mensagens da fila, validando a consistência dos dados extraídos da mensagem e armazenando-os.

(vi) **Repositório de proveniência:** este repositório contém todos os dados de proveniência capturados durante a execução dos *workflows*. Ele está sempre acessível ao cientista, mesmo durante as execuções realizadas.

A seguir detalhamos o funcionamento de cada um dos componentes envolvidos na arquitetura do ProvDoop. A Figura 3 apresenta a arquitetura conceitual do ProvDoop, seus seis componentes e as interações entre eles.

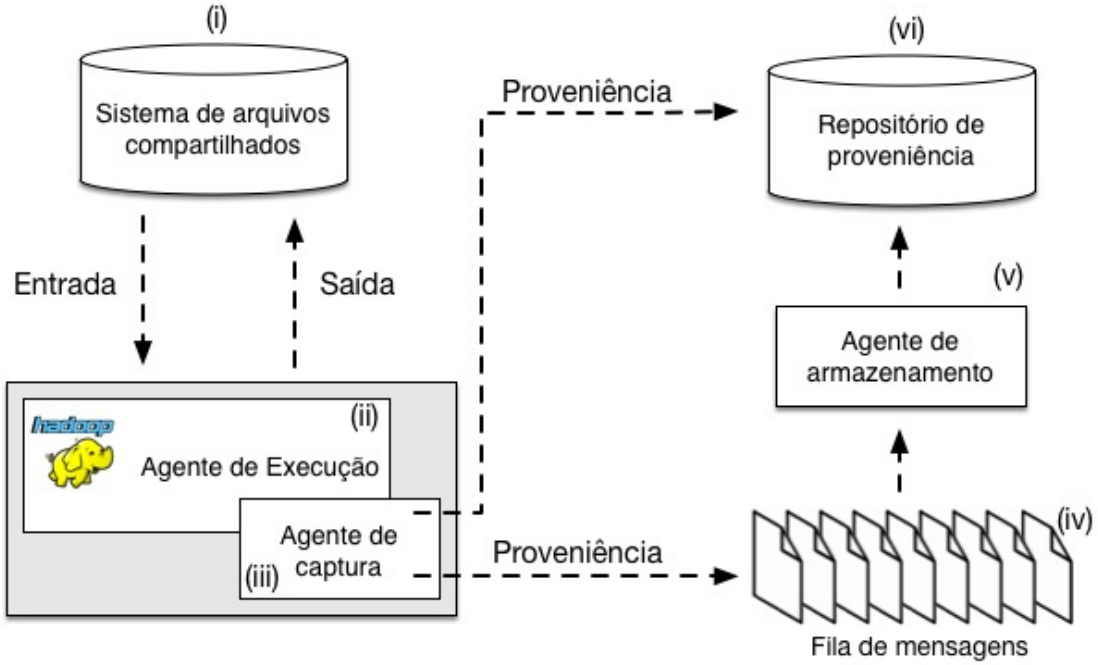

Figura 3 Arquitetura Conceitual do ProvDoop

O **sistema de arquivos compartilhados** contém todos os arquivos de entrada que são utilizados como insumos na execução dos *workflows* científicos, bem como todas as

saídas geradas a partir das execuções realizadas. Saídas intermediárias, ou seja, saídas geradas por todas as atividades ao longo da execução do *workflow* também são armazenadas no sistema de arquivos compartilhados, sendo disponibilizadas não só para os cientistas mas também para todas as máquinas envolvidas na execução do *workflow*.

Como as execuções realizadas com Hadoop tendem a ser distribuídas, é fundamental que todos os arquivos necessários estejam acessíveis para todos os nós envolvidos na execução e que esses nós possam armazenar novos arquivos gerados nesse sistema de arquivos compartilhados para que sejam utilizados por outras atividades, pois em uma execução distribuída é comum que um arquivo gerado por um nó seja utilizado por outro nó envolvido na execução. Além disso, é fundamental que os cientistas tenham acesso a esses arquivos para que possam avaliar os resultados gerados.

O **agente de execução** foi desenvolvido de forma a permitir a utilização do Hadoop como motor de execução de *workflows* científicos, mantendo o fluxo de atividades do mesmo e invocando cada um dos programas envolvidos na execução. No entanto, como dito anteriormente, o Hadoop descreve suas operações apenas por meio de funções de mapeamento (*Map*) e de junção (*Reduce*). Sendo assim, foi necessário realizar uma abstração do problema de execução de *workflows* científicos para que este pudesse ser processado utilizando o modelo de programação MapReduce.

O agente de execução é constituído por funções *Maps* e funções *Reduce*, que foram desenvolvidas para cada atividade envolvida na execução do *workflow* científico e por um controlador que será executado no nó mestre e que coordenará como e quando cada um desses *Maps* e *Reduces* deve ser chamado. Os *Maps*, os *Reduces* e o controlador do agente de execução foram desenvolvidos de forma a serem executados no Hadoop tanto na nuvem quanto localmente, para testes.

A Figura 4 mostra a arquitetura desenvolvida para o agente de execução em termos do fluxo de dados. Observa-se cada um dos *Maps* envolvidos na execução de um *workflow* científico com N entradas e M atividades, onde cada *Map* XY representa a execução de um *Map* da atividade X com uma entrada Y de forma que a saída gerada por sua execução sirva de entrada para uma execução *Map* ZY, sendo Z = X+1. Ou seja, cada saída de uma atividade é utilizada como entrada na atividade subsequente e assim por

diante, até que se passe pela função *Reduce*, que reúne todos os resultados obtidos pela execução da atividade M para todas as N entradas, gerando o resultado final.

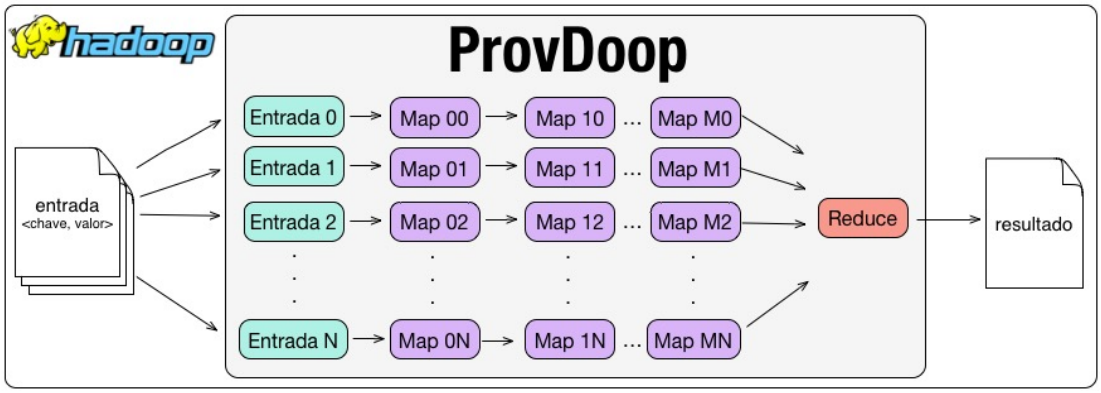

Figura 4 Arquitetura do Agente de Execução do ProvDoop

A entrada para a execução do *workflow* científico é um conjunto de tuplas <chave, valor> onde a chave identifica a entrada a ser executada e o valor é o caminho completo para o arquivo de entrada. A partir da execução do arquivo contido nesse caminho, a primeira atividade gerará os resultados iniciais, que serão consumidos pela segunda atividade e assim por diante.

Além disso, para que a execução ocorra de maneira a garantir a ordem correta da chamada de cada atividade, bem como o fornecimento das entradas correspondentes para cada uma delas, parte do agente de execução se dá por um controlador. Esse controlador é executado no nó mestre de maneira a garantir que uma atividade só será chamada quando suas entradas, produzidas pelas atividades anteriores ou fornecidas pelos cientistas, estiverem prontas para serem processadas e acessíveis a cada um dos nós responsáveis por sua execução.

A execução se dá então da seguinte maneira: o nó mestre prepara o conjunto de entradas a serem consumidas pela primeira atividade do *workflow*, garantindo que tais entradas estejam acessíveis aos nós escravos pelo sistema de arquivos compartilhados apresentado anteriormente. Os nós escravos executam a primeira atividade, processando os arquivos de entrada, gerando arquivos de saída e colocando-os no sistema de arquivos compartilhados para que possam ser consumidas pela atividade seguinte. O nó mestre precisa garantir que os arquivos de saída a serem consumidos pela próxima atividade estejam prontos e disponíveis e só então poderá invocar a chamada da próxima atividade. Esse processo se repete até o fim da execução do

workflow. Todos os arquivos gerados ficam disponíveis no sistema de arquivos compartilhados para que o cientista acesse-os.

O **agente de captura** é responsável por capturar todos os dados de proveniência durante a execução do *workflow*. Ele se acopla ao agente de execução e, em pontos específicos da execução, realiza a coleta dos dados, enviando-os diretamente para a base de proveniência ou para a fila de mensagens, conforme mostrado adiante, para que sejam processados e armazenados pelo agente de armazenamento.

O agente de captura está presente tanto nos nós escravos quanto no nó mestre. Quando o agente de captura coleta dados de execução do nó mestre, ele pode armazená-los diretamente na base de dados, pois o nó mestre é um único nó centralizado, descartando problemas de concorrência, que coordena toda a execução do *workflow*, estando ciente do estado atual do mesmo, de forma que os dados coletados pelo nó mestre não apresentem problemas de inconsistência. Além disso, o nó mestre não realiza o processamento das atividades de fato, fazendo apenas a distribuição das funções e recursos entre os nós escravos.

Sendo assim, o nó mestre não exige tanto processamento quanto os nós escravos e o tempo necessário para armazenar os dados na base de dados não tende a impactar o tempo de execução do *workflow*. O agente de captura do nó mestre armazena somente dados referentes à execução de cada atividade e do *workflow* como um todo, pois os detalhes da execução de cada ativação nos nós escravos não estão acessíveis a ele e, caso estivessem, seria oneroso mapear esses detalhes para cada um dos nós escravos existentes.

Já os dados capturados nos nós escravos poderiam apresentar problemas de inconsistência, pois os nós escravos não estão cientes do andamento da execução do *workflow* como um todo, tendo somente as informações necessárias para a execução em seu nó. Além disso, uma mesma execução pode apresentar milhares de nós escravos e a necessidade de abrir uma conexão com a base de dados de proveniência para cada um deles pode causar um grande impacto no tempo de execução de cada entrada, sobretudo em atividades que costumam ser executadas rapidamente, além de gerar uma série de preocupações em relação à concorrência das escritas na base de dados.

A **fila de mensagens** foi criada para solucionar esse problema. Todos os agentes de captura presentes nos nós escravos mandam mensagens contendo dados de

proveniência coletados para a fila de mensagens. Dessa forma, não é necessário que os nós escravos abram conexão com a base de dados, gerando possíveis problemas de concorrência, inconsistência e atraso na execução das atividades. Como todos os nós escravos precisam mandar mensagens para essa fila, ela precisa estar acessível para todas as máquinas envolvidas na execução do *workflow*, ou seja, ela precisa ser uma fila escalável, podendo lidar tanto com execuções envolvendo poucos nós quanto execuções envolvendo milhares de nós.

As mensagens presentes na fila são então coletadas e processadas pelo **agente de armazenamento**, que analisa cada uma das mensagens recebidas e as compara com o estado atual da base de dados de proveniência, garantindo que não serão armazenados dados inconsistentes ou desatualizados.

Sendo assim, só dois componentes realizam comunicação direta com o repositório de proveniência: o agente de captura, que armazena informações coletadas no nó mestre sobre a execução das atividades e do *workflow*, e o agente de armazenamento, que armazena informações sobre a execução de cada uma das entradas nos nós escravos, através do processamento das mensagens da fila.

Por fim, **o repositório de proveniência** contém dados coletados referentes às execuções em andamento, às execuções passadas e aos ambientes de execução utilizados, permitindo que o cientista realize consultas à base de dados em tempo real e possa tomar decisões rápidas e embasadas em relação ao seu experimento científico.

3.2 Modelo de Proveniência do ProvDoop

O ProvDoop foi projetado e implementado para armazenar dados de proveniência em tempo de execução para *workflows* científicos executados com Hadoop. Ele armazena informações sobre o experimento que está sendo executado e sobre o ambiente onde a execução está ocorrendo.

O modelo de proveniência construído para o ProvDoop é baseado no *Open Provenance Model* (OPM) (Moreau *et al.* 2008a, 2008b). O OPM é uma recomendação aberta e focada em permitir que informações sejam trocadas entre diversos sistemas, através da utilização de um modelo compartilhado que permita a compatibilidade entre eles, em possibilitar que desenvolvedores construam e compartilhem ferramentas que operem em um modelo unificado de proveniência e em fazer com que a proveniência seja

definida de forma independente da tecnologia utilizada, sendo ela gerada por sistemas de computadores ou não (Moreau *et al.* 2011). O OPM não é diretamente instanciável em um banco de dados, mas é uma representação padrão de proveniência de dados para a maioria dos sistemas de gerenciamento de *workflows* científicos (Altintas *et al.* 2004b, Callahan *et al.* 2006, Deelman *et al.* 2007, Fahringer *et al.* 2005, Hull *et al.* 2006b, Taylor *et al.* 2007c, Zhao *et al.* 2007).

A Figura 5 representa o esquema lógico projetado para o repositório de proveniência do ProvDoop. Todas as informações relacionadas a execuções anteriores de *workflows* científicos bem como a execuções em andamento são recuperadas a partir do repositório de proveniência do ProvDoop. Este esquema é representado por meio de um diagrama de entidade-relacionamento (DER) e foi modelado com base em requisitos levantados juntamente com cientistas e seguindo as recomendações do OPM.

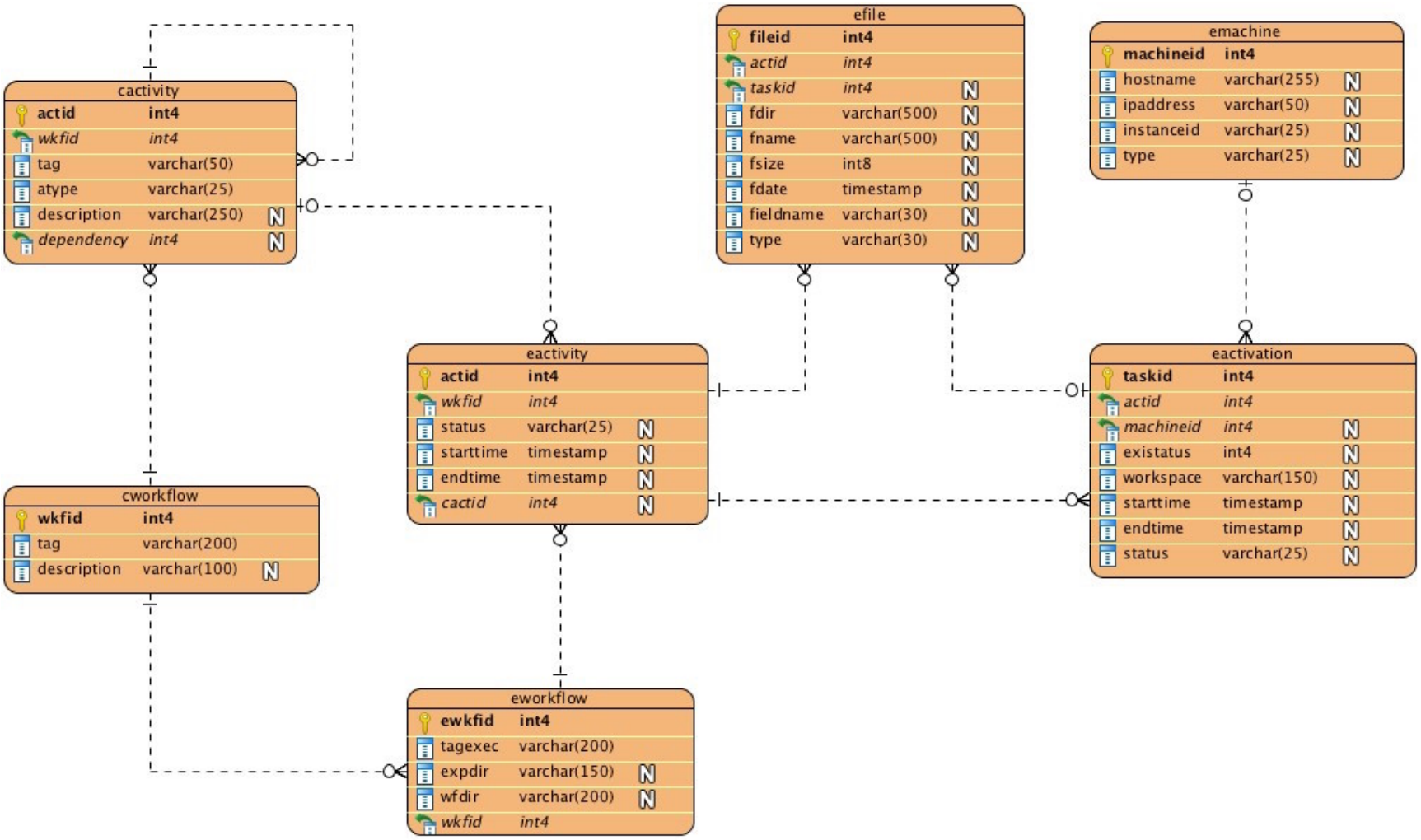

Figura 5 Esquema Lógico de Proveniência do ProvDoop

Na Figura 5, pode-se observar que os dados referentes aos *workflows* são divididos entre as tabelas *cworkflow* e *eworkflow*, assim como os dados referentes às atividades são divididos em *cactivity* e *eactivity*. As tabelas que possuem nome iniciando com a letra *C* referem-se a tabelas com dados de composição, enquanto tabelas que possuem nome iniciando com a letra *E* referem-se a dados de execução. Dados de composição tratam de informações mais genéricas referentes à estrutura do *workflow* ou da atividade. Enquanto dados de execução tratam de informações mais específicas referentes a cada uma das execuções realizadas. As diferenças entre composição e execução são apresentadas com mais detalhes na Seção 2.2.

Na tabela *cworkflow*, o atributo *tag* serve para classificar um determinado experimento (*e.g.* bioinformática, prospecção de petróleo em águas profundas, etc). Na tabela *cactivity*, de maneira análoga, o atributo *tag* identifica qual é aquela atividade no *workflow*. O atributo *dependency* indica a dependência daquela atividade em relação a alguma outra. Ou seja, caso uma atividade B dependa da saída gerada por uma atividade A para ser executada, então B é dependente de A e isso será refletido por esse atributo. É esse atributo que é levado em consideração pelo nó mestre para determinar a ordem da execução das atividades e que atividades devem esperar o fim da execução da anterior para que possam prosseguir com sua execução.

A tabela *eworkflow* possui o atributo *tagExec*, responsável por identificar unicamente uma tentativa de execução, e os atributos *expDir* e *wfDir* que representam respectivamente o diretório onde serão armazenados os arquivos referentes ao resultado da execução com Hadoop e o diretório onde serão armazenados os arquivos resultantes da execução do *workflow*.

A tabela *eactivity* representa a execução de uma atividade em uma execução de *workflow* específica. Seu atributo *status* pode apresentar os estados *BLOCKED*, indicando que a atividade está bloqueada aguardando o fim da execução de alguma atividade da qual ela é dependente, *READY*, indicando que a atividade está pronta para ser executada e os recursos estão sendo alocados para que a execução seja iniciada, *RUNNING*, indicando que a atividade está em execução e *FINISHED*, indicando que a execução da atividade já foi finalizada. Os atributos *starttime* e *endtime* indicam respectivamente o tempo inicial e final da execução da atividade, ou seja, esses tempos consideram o período em que a atividade se manteve no estado *RUNNING*.

A tabela *eactivation* possui estados análogos aos da tabela *eactivity* e os atributos *starttime* e *endtime* também funcionam da mesma maneira. O atributo *workspace* indica onde serão armazenados os resultados referentes àquela ativação em particular. E o atributo *machineid* indica em qual máquina da tabela e*machine* aquela ativação foi executada.

Na tabela *emachine* são guardadas informações referentes às máquinas onde cada uma das ativações foram executadas, como nome do servidor (*hostname)*, endereço de IP (*ipaddress)*, identificador da instância (*instanceid)* e tipo da instância (*type)*. Esses valores são especialmente úteis em execuções distribuídas na nuvem, onde diversas máquinas são instanciadas e cada uma delas fica responsável por um conjunto de ativações. Através dessa tabela, é possível descobrir exatamente qual máquina ficou responsável por qual ativação e quais as características dela. Além disso, na análise de desempenho é importante considerar que tipos de máquina foram instanciadas em cada experimento. Dessa forma é possível fazer uma análise posterior comparando as máquinas utilizadas na execução e os desempenhos obtidos, permitindo que se possa mapear que máquinas são mais adequadas para cada execução. Essa tabela permite esse tipo de análise, tornando as execuções de experimentos futuros mais adequadas e otimizadas.

Por fim, a tabela *efile* representa cada um dos arquivos utilizados no *workflow*. Através dessa tabela, podemos saber qual o nome de cada arquivo, o diretório em que eles se encontram no sistema de arquivos compartilhados, seu tamanho, sua data de criação, o seu formato e seu tipo, que pode ser de entrada (*input)* ou de saída (*ouput)*.

3.3 Detalhes de implementação

Embora o ProvDoop possa ser executado em ambientes fora da nuvem computacional para execuções não distribuídas, o ambiente de nuvem foi utilizado nesta dissertação devido à flexibilidade que a nuvem proporciona em termos de escalabilidade, elasticidade e variedade de recursos. O provedor de nuvem escolhido foi o Amazon Web Service (AWS), pois ele é atualmente o líder neste segmento (Gartner 2015), sendo o ambiente de computação em nuvem mais popular, onde muitas aplicação científicas e comerciais foram implantadas (Armbrust *et al.* 2010, Hey *et al.* 2009, Hoffa *et al.* 2008, Matsunaga *et al.* 2008, Ocaña *et al.* 2011b, 2011a).

Para detalhar a implementação do ProvDoop, é preciso considerar cada um dos seis (6) componentes envolvidos na sua arquitetura, apresentados na Seção 3.1, são eles: o sistema de arquivos compartilhados, o agente de execução, o agente de captura, a fila de mensagens, o agente de armazenamento e o repositório de proveniência.

A única restrição a ser levada em conta para o funcionamento do **sistema de arquivos compartilhados**, é que ele seja acessível para todos os nós envolvidos na execução, permitindo leitura e escrita por parte destes nós. A latência dessas operações também é um ponto importante a ser considerado pois em execuções de *workflows* em larga escala, essa latência pode causar grandes impactos no desempenho.

Levando essa restrição em consideração, utilizamos o *Amazon Simple Storage Service* (Amazon S3) como sistema de arquivos compartilhados do ProvDoop. O Amazon S3 é um serviço de armazenamento de objetos escalável e que pode ser acessado de qualquer lugar da *Web*, desde que configurado adequadamente. Todos os nós envolvidos na execução do *workflow* podem se comunicar com o Amazon S3, acessando arquivos que são utilizados como entrada e salvando arquivos produzidos na saída, de maneira distribuída e com baixa latência na comunicação com outros recursos da AWS que estejam presentes na mesma região (Varia 2011).

Os **agentes de execução**, **captura** e **armazenamento** foram todos desenvolvidos utilizando a linguagem Java Versão 8 *Update* 25. Para a implementação do agente de execução, tanto do controlador presente no nó mestre quanto dos *Maps* e *Reduces* presentes nos nós escravos, foram necessárias as bibliotecas do Hadoop. A versão 2.6.0 do Hadoop foi escolhida tanto para as bibliotecas utilizadas quanto para a configuração do ambiente de desenvolvimento.

O **agente de captura** e o **agente de armazenamento** utilizam bibliotecas para manipular arquivos no formato JSON (*JavaScript Object Notation*), pois as mensagens da fila foram construídas nesse formato. O formato JSON foi escolhido por ser considerado um formato leve e amplamente utilizado para troca de mensagens. Além disso, o Hibernate foi utilizado para mapear as tabelas do repositório de proveniência nas classes do projeto e o conector JDBC do PostgreSQL para realizar a conexão com a base de dados. Todos os três agentes desenvolvidos utilizam Maven para gerenciar suas dependências, JUnit para o desenvolvimento de testes unitários e AWS SDK para Java para realizar a comunicação com todos os recursos utilizados da AWS. Os três agentes foram testados em execuções

distribuídas realizadas em ambientes configurados em instâncias do *Amazon Elastic Cloud Compute* (Amazon EC2) e localmente, em execuções não distribuídas. As execuções não distribuídas foram realizadas apenas para fins de testes durante o desenvolvimento, pois o Hadoop só se torna realmente vantajoso em ambientes distribuídos (White 2012).

O **agente de armazenamento** foi implementado através do desenvolvimento de uma *thread* que fica em execução checando se há mensagens na fila a serem processadas. Em caso positivo, o agente de armazenamento recolhe as mensagens da fila e processa cada uma delas, de forma a verificar se a mensagem está atualizada e com informações consistentes com aquelas armazenadas no repositório de proveniência. Por fim, os dados são armazenados na base de dados e passam a estar disponíveis para consulta.

A *thread* do agente de armazenamento foi programada de forma a ser executada a cada segundo em busca de mensagens na fila e, caso existam mensagens, o processamento só encerra quando todas as mensagens forem lidas e processadas e a fila estiver vazia novamente. Além disso, a *thread* foi programa de forma a não executar concorrentemente com outra já em execução.

A **fila de mensagens** precisa ser escalável e distribuída, pois ela precisa atender desde execuções com poucos nós até execuções com milhares de nós, e é fundamental que todos os nós envolvidos na execução tenham acesso a ela, já que os agentes de captura estão presentes também nos nós escravos coletando dados de proveniência e enviando-os para a fila de mensagens durante toda a execução. Diante de tais restrições, decidimos utilizar o *Amazon Simple Queue Service* (Amazon SQS), um serviço de fila oferecido pela AWS que é rápido, escalável e completamente gerenciável (Varia 2011). Desta forma, todas as medidas para garantir a disponibilidade e escalabilidade do serviço e a não perda de mensagens são tomadas pela própria AWS.

As mensagens da fila foram construídas de forma que, mesmo que sejam processadas mais de uma vez, elas manterão o resultado mais atualizado possível e não gerarão resultados inconsistentes no banco. Essa é uma recomendação da AWS devido ao caráter distribuído da fila, que não garante que uma mensagem só será recebida uma única vez.

A fila de mensagens foi configurada com as configurações padrões de fila distribuída, que seguem: uma mensagem recebida por um componente ficará 30 segundos invisível até que possa ser recebida novamente, uma mensagem fica na fila por 4 dias caso não seja

apagada antes, o tamanho máximo de uma mensagem é de 256KB e não há qualquer atraso programado para a entrega da mensagem, de forma que uma mensagem se torna disponível no momento em que for inserida. A mensagem é construída no formato JSON e possui todos os dados relevantes para que o estado de uma ativação possa ser atualizado.

A Figura 6 mostra um exemplo de mensagem disparada no início da execução de uma ativação. O agente de captura de um nó escravo verifica que uma ativação começou a ser executada e então recolhe informações referentes ao início da execução, constrói uma mensagem no formato JSON e então insere essa mensagem na fila para que ela seja posteriormente processada e armazenada pelo agente de armazenamento.


```
{
   "messageType":"TASK_STARTED",
   "taskId":201,
   "startTime":1454794366745,
   "workspace":"/exp/sciphy.output/provenance-on/SciPhyTestTag/ORTHOMCL256/",
   "instanceType":"t2.micro",
   "instanceId":"i-2040e5a0",
   "hostName":"ec2-52-23-247-211.compute-1.amazonaws.com",
   "ipAddress":"52.23.247.211"
}
```


Figura 6 Exemplo de Mensagem Disparada para a Fila no Início da Execução de uma Ativação

A mensagem de exemplo mostra que a ativação iniciou a execução e passa o identificador da ativação, o momento em que a execução iniciou, a pasta em que os dados de saída estão sendo armazenados no sistema de arquivos compartilhados, o tipo de máquina que está sendo utilizado na execução, o identificador da máquina em questão, o nome do servidor e o endereço IP do mesmo.

A Figura 7 mostra uma mensagem indicando o fim da execução dessa mesma ativação. Essa mensagem contém todos os dados que a mensagem de início da execução contém, mais os dados obtidos ao fim da execução. Essa redundância de informações visa garantir que, mesmo que as mensagens sejam recebidas fora de ordem, todos os dados poderão ser armazenados corretamente. Essa também é uma recomendação da AWS, pois devido ao caráter distribuído da fila não é possível garantir que todas as mensagens chegarão na mesma ordem em que foram inseridas. Além disso, o agente de armazenamento foi desenvolvido de forma a lidar com possíveis mensagens fora de ordem ou mensagens duplicadas, garantindo sempre a consistência dos dados antes de armazená-los.

A mensagem disparada ao fim da execução apresenta, além das informações já mostradas na mensagem de início, o momento em que a execução encerrou e os arquivos que foram gerados durante a execução. Cada um dos arquivos possui informações sobre o diretório onde o mesmo se encontra, a sua data de criação, o seu tamanho, o seu formato e o seu nome.


```
{
   "messageType":"TASK_FINISHED",
   "taskId":201,
   "startTime":1454794366745,
   "endTime":1454794368629,
   "instanceType":"t2.micro",
   "instanceId":"i-2040e5a0",
   "hostName":"ec2-52-23-247-211.compute-1.amazonaws.com",
   "ipAddress":"52.23.247.211",
   "workspace":"/exp/sciphy.output/provenance-on/SciPhyTestTag/ORTHOMCL256/",
   "outputFiles":[
      {
         "fdir":"/exp/sciphy.output/provenance-on/SciPhyTestTag/ORTHOMCL256/",
         "fdate":1454794368555,
         "fsize":2646,
         "fieldName":"FASTA_NUMBERED",
         "fname":"ORTHOMCL256.fastaNumbered"
      },
      {
         "fdir":"/exp/sciphy.output/provenance-on/SciPhyTestTag/ORTHOMCL256/",
         "fdate":1454794368629,
         "fsize":2635,
         "fieldName":"MAFFT",
         "fname":"ORTHOMCL256.mafft"
      }
   ]
}
```


Figura 7 Exemplo de Mensagem Disparada para a Fila ao Fim da Execução de uma Ativação

Por fim, o **repositório de proveniência** é mantido utilizando um banco de dados relacional PostgreSQL versão 9.3.1 que foi configurado em uma máquina dedicada no *Amazon Relational Database Service* (Amazon RDS), um serviço que facilita a configuração, a operação e a escalabilidade de bancos de dados relacionais na nuvem.

A Figura 8 ilustra uma visão geral de como a arquitetura do ProvDoop foi implementada, mostrando os componentes da arquitetura e os serviços da nuvem utilizados para a implantação de cada um dos componentes.

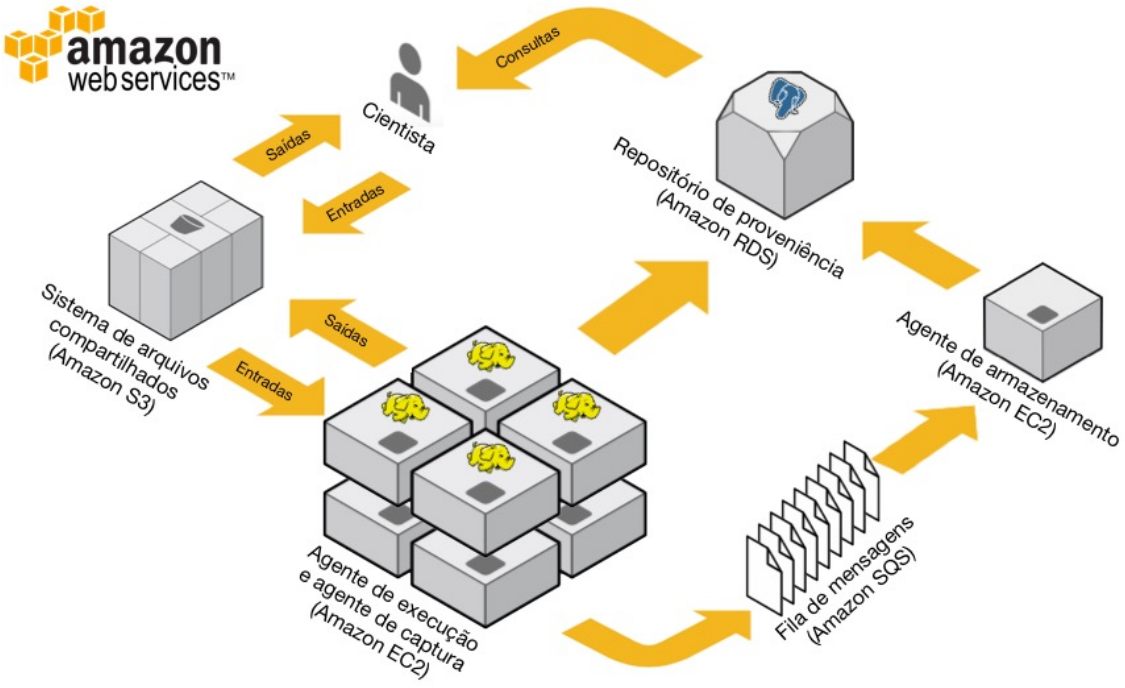

Figura 8 Arquitetura do ProvDoop e Serviços da Nuvem Utilizados na Implantação

Na Figura 8 é possível observar todos os componentes apresentados na primeira seção deste capítulo, mas agora cada um dos elementos de nuvem utilizados na implementação da arquitetura estão representados na figura, de forma que pode-se observar como cada um dos componente foi implementado na nuvem. Nessa arquitetura em particular, utilizam-se recursos da *Amazon Web Services*, mas estes poderiam ser adaptados de acordo com o provedor de nuvem utilizado. Utilizamos o Amazon S3 para o sistema de arquivos compartilhado. Os agentes de execução, captura e armazenamento foram executados em instância da Amazon EC2. O serviço Amazon SQS foi utilizado para a fila de mensagens. Por fim, uma instância criada a partir do Amazon RDS com PostgreSQL foi utilizada para o repositório de proveniência.

Capítulo 4 - Avaliação Experimental

Este capítulo apresenta os resultados experimentais obtidos através de execuções com o ProvDoop e as configurações utilizadas para executar o *workflow* científico SciPhy, caso de estudo utilizado nesta dissertação.

A ideia central deste capítulo é analisar o desempenho do ProvDoop, avaliando os resultados obtidos com execuções com proveniência de dados e comparando-as com execuções sem proveniência de dados. Dessa forma será possível analisar se há perda no desempenho ao adicionar proveniência em execuções realizadas com o ProvDoop em relação a execuções realizadas com Hadoop sem proveniência em tempo de execução. E, caso existam tais perdas, se elas são significativas frente aos ganhos que a proveniência de dados em tempo de execução oferece. Além disso, neste capítulo propomos uma série de consultas que visam testar a cobertura do repositório de proveniência, validando a hipótese de que os dados podem ser consultados em tempo real, gerando ganhos para os cientistas.

Este capítulo está organizado como segue. Na seção 4.1 descrevemos as características do *workflow* científico utilizado como estudo de casos que foi modelado e executado com ProvDoop. Nas seções 4.2 e 4.3 apresentamos respectivamente a configuração do ambiente e do experimento utilizado na avaliação. A seção 4.4 contém a análise de desempenho realizada a partir do experimento e, por fim, a seção 4.5 apresenta uma análise das consultas de proveniência propostas.

4.1 O SciPhy

Bioinformática e Biologia Computacional são termos utilizados para descrever a área interdisciplinar que une a tecnologia da informação com a biologia molecular (Lengauer 2002). De acordo com a definição *do National Institutes of Health (*NIH), a bioinformática é "pesquisa, desenvolvimento ou aplicação de ferramentas e abordagens computacionais para a expansão do uso de dados biológicos, médicos, comportamentais ou de saúde, incluindo aqueles usados para adquirir, armazenar, organizar, analisar ou visualizar esses dados." A disciplina relacionada de biologia computacional é "o desenvolvimento e aplicação de métodos dado-analíticos e teóricos, modelagem matemática e técnicas de simulação computacional para o estudo de sistemas biológicos, comportamentais e sociais".

Uma das áreas de bioinformática é a filogenia. Filogenia é o termo comumente utilizado para hipóteses de relações evolutivas, ou relações filogenéticas, de um grupo de organismos, isto é, determinar as relações ancestrais entre espécies conhecidas. Uma árvore filogenética, também chamada de árvore da vida (Zvelebil e Baum 2007), é uma representação gráfica, em forma de uma árvore, das relações evolutivas entre várias espécies. Resultados obtidos a partir de tecnologias filogenéticas podem contribuir para outras áreas de bioinformática como o desenvolvimento de novas drogas (Anderson 2003).

Durante a última década, tem havido um aumento sem precedentes no volume de dados com o objetivo de sequenciamento para categorizar todos os genes de genomas de diversos organismos, por exemplo, o genoma humano (Lander 2001). Esse aumento no volume de dados envolvidos em experimentos de bioinformática traz consigo uma maior necessidade de infraestrutura computacional de alto desempenho.

Dentre os *workflows* que necessitam de ambientes computacionais de alto desempenho, nesta dissertação trabalharemos com o SciPhy. O SciPhy (Ocaña *et al.* 2011b) é um *workflow* científico que foi desenvolvido para gerar árvores filogenéticas com máxima verossimilhança (Yang 1994) a partir de uma grande coleção de arquivos multi-fasta de sequências biológicas de diversos organismos.

O *workflow* SciPhy é composto por quatro atividades principais que são: construção do alinhamento múltiplo de sequências (AMS), conversão de formato do alinhamento, pesquisa e eleição do melhor modelo evolutivo a ser usado e, por fim, a construção da árvore filogenética. A Figura 9 mostra as fases de execução apresentadas acima e as aplicações de bioinformática responsáveis por executar cada uma dessas etapas, são elas: o MAFFT (Katoh e Toh 2008) como programa de AMS, o ReadSeq (Gilbert 2003) na etapa de conversão de formato, o ModelGenerator (Keane *et al.* 2006) na etapa de pesquisa e eleição do modelo evolutivo e o RAxML (Stamatakis 2006) na construção da árvore filogenética.

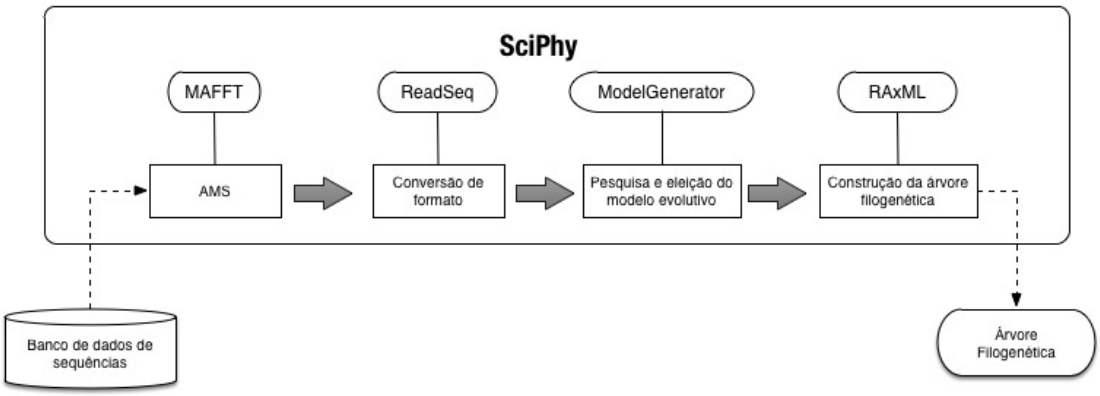

Figura 9 Atividades Envolvidas na Execução do SciPhy

Embora o número de atividades encadeadas no SciPhy não seja um número alto, na prática este *workflow* pode ser demasiadamente complexo de ser gerenciado devido ao volume de dados trabalhados e a quantidade de parâmetros que podem ser explorados, uma vez que o cientista não conhece *a priori* qual a configuração que levará à melhor árvore filogenética.

Um experimento típico pode analisar centenas ou milhares de arquivos multi-fasta, cada um contendo centenas ou milhares de sequências biológicas. Como o SciPhy foi projetado de forma que cada arquivo seja processado independentemente dos demais, ele se torna um bom candidato para a exploração paralela de suas tarefas.

Mais informações sobre o SciPhy podem ser encontradas em Ocaña *et al.* (2011).

4.2 Configuração do Ambiente

Para os experimentos executados nesta dissertação, implantamos o ProvDoop no ambiente de nuvem Amazon EC2, fornecido pela Amazon Web Services (AWS 2015).

O Amazon EC2 fornece vários tipos diferentes de máquinas virtuais, com diferentes capacidades de CPU, memória, armazenamento e capacidade de rede. O processo de escolha de um tipo de instância envolve algumas etapas:

(i) **Escolha da família de instâncias:** Essa primeira etapa se refere a qual grupo de instâncias será considerado para uso. Existem as famílias de instâncias de uso geral (famílias T e M), otimizadas para computação (família C), otimizadas para memória (família R), otimizadas para processamentos gráficos (família G) e otimizadas para armazenamento (família I e D).

(ii) **Escolha do tamanho das instâncias:** O tamanho de cada instância, dependendo da família escolhida, pode variar entre *nano*, *micro*, *small*, *medium*, *large*, *xlarge*, *2xlarge*, *4xlarge*, *8xlarge* e *10xlarge*. Esses tamanhos influenciam na quantidade de memória disponível, podendo ir de 0,5 a 244 GB, na capacidade de armazenamento, no desempenho de rede e no processador físico utilizado.

(iii) **Escolha do sistema operacional:** Todos os sistemas operacionais mais comuns estão disponíveis para escolha.

Sendo assim, seguindo as etapas apresentadas, primeiro foi decidido o uso de uma família de instâncias de uso geral. Essa decisão foi tomada pois ela apresenta instâncias que costumam ter desempenho adequado à maior parte das aplicações, sem possuir otimizações que poderiam enviesar os resultados obtidos no experimento.

Foi escolhida a família M1 pois ela é descrita como uma família que oferece recursos equilibrados de computação, memória e rede, se mostrando uma boa opção para diversas aplicações. Além disso, essa família não possui desempenho com capacidade de intermitência (AWS 2015), o que poderia inviabilizar uma avaliação precisa dos resultados obtidos. O tamanho de instância escolhido foi o *large*, que possui dois núcleos de processamento Intel Xeon, 7,2 GB de memória RAM e desempenho de rede moderado.

O sistema operacional utilizado é o Amazon Linux 64 bits, que é baseado no RHEL 5.x e RHEL6, sendo compatível com CentOS5.x. Esse sistema operacional já inclui as ferramentas de linha de comando da AWS, Python, Ruby, Perl e Java.

Foi necessário instalar e configurar as aplicações de bioinformática utilizadas e o Hadoop 2.6.0 em todas as máquinas envolvidas na execução do *workflow*. Todos os comandos necessários para as instalações e configurações foram inseridos em um *script* no formato *.sh* que é executado no momento em que a instância é criada, fazendo com que todas as máquinas criadas tenham o mesmo ambiente configurado. Todos os programas e arquivos necessários na instalação se encontram disponíveis no Amazon S3, de forma que o *script* possa baixá-los rapidamente, graças à baixa latência na comunicação entre Amazon EC2 e Amazon S3 presentes numa mesma região. Essas máquinas são utilizadas pelos agentes de execução e de captura, que são disponibilizados através de um arquivo no formato *.jar* baixado automaticamente pelo *script* mencionado.

Uma única instância Amazon EC2 foi criada para o agente de armazenamento, com as mesmas configurações de *hardware* das máquinas utilizadas pelos agentes de captura e

execução, mas nessa máquina não houve a necessidade de instalar quaisquer aplicações extras, pois o agente de armazenamento só precisa do Java, já instalado na instância por padrão.

As máquinas foram configuradas de forma a serem acessadas via SSH somente com uma chave no formato *.pem* que só pode ser baixada uma vez e deve ser guardada para todos os acessos necessários posteriormente. Essa é uma recomendação de segurança da AWS.

Outra recomendação de segurança é que os nós escravos estejam acessíveis somente através do nó mestre. Sendo assim, para acessar um nó escravo é necessário primeiro acessar o nó mestre utilizando a chave *.pem*. O acesso ao painel de execução do Hadoop é configurado através da utilização de um *proxy* instalado no navegador, também seguindo as devidas recomendações de segurança.

As configurações utilizadas na fila de mensagens são as configurações padrão do serviço Amazon SQS e refletem o comportamento descrito na Seção 3.3.

Todas as máquinas, juntamente com a fila, foram instanciadas na região leste dos Estados Unidos, no norte de Virgínia. E todas as máquinas foram inseridas em uma mesma zona de disponibilidade, na tentativa de diminuir a latência de comunicação entre as instâncias (Varia 2011).

Ajustes finos na configuração do Hadoop a fim de melhorar o desempenho fogem do escopo desta dissertação, embora este seja um assunto bastante explorado na academia (Herodotou *et al.* 2011, Joshi 2012, Heger 2013, Li *et al.* 2014, Shi *et al.* 2015). No entanto, foi necessário realizar algumas mudanças nas configurações padrões do Hadoop para que os testes pudessem ser realizados.

A primeira alteração necessária foi no arquivo de configuração *mapred-site.xml*, onde aumentamos o tempo máximo de execução de uma tarefa. Essa necessidade surgiu graças ao programa ModelGenerator, que leva vários minutos para ser executado para uma única entrada. Então, mesmo para experimentos com poucas entradas, é recomendável que se faça esse ajuste para garantir que a execução do *workflow* científico não será interrompido por ter atingido o tempo máximo de processamento para uma entrada.

Além disso, foram realizadas alterações nos arquivos *capacity-scheduler.xml* e *yarn-site.xml* a fim de ajustar o número de núcleos disponíveis para o processamento das tarefas, fazendo com que esse número refletisse o número de núcleos físicos da máquina, seguindo recomendações de melhores práticas de configuração do Hadoop (White 2012).

4.3 Configuração do Experimento

Para executar o SciPhy em paralelo no ProvDoop, o experimento foi configurado de forma que o conjunto de dados de entrada fosse constituído por 200 arquivos multi-fasta com sequências de proteínas extraídas do banco de dados biológico RedSeq *release* 48 (Pruitt *et al.* 2009).

Esse conjunto de dados é formado por 200 arquivos multi-fasta de aminoácidos e cada arquivo multi-fasta é constituído por uma média de 10 sequências biológicas, sendo o menor arquivo com 9 sequências e o maior arquivo com 11 sequências.

Na execução do SciPhy, cada arquivo multi-fasta é alinhado para obter um AMS utilizando o programa MAFFT versão 7.271. Em seguida, a conversão do formato do alinhamento é realizado utilizando o programa ReadSeq versão 2.1.30 e a pesquisa e eleição do melhor modelo evolutivo são feitas pelo programa ModelGenerator versão 0.84. Por fim, a geração da árvore filogenética é feita com o programa RAxML versão 8.0.20.

Cada um dos programas envolvidos na execução do SciPhy foram colocados no Amazon S3 para que os *scripts* de instalação pudessem baixá-los e configurá-los no momento de inicialização de cada máquina utilizada na execução.

A fim de executar cada uma das atividades do SciPhy no Hadoop foi preciso implementar funções *Map* e *Reduce* específicas. Cada tarefa *Map* baixa os arquivos multi-fasta de entrada, e transfere essa entrada de forma a executar o programa associado. As próximas atividades no *workflow* seguem a mesma abordagem, mas essas novas atividades vão consumir os dados produzidos a partir da atividade anterior.

4.4 Análise de Desempenho

Como apresentado nos capítulos anteriores, o foco desta dissertação é inserir proveniência de dados em tempo de execução no Hadoop em nuvens computacionais e avaliar os possíveis impactos gerados no desempenho das execuções, analisando se os impactos causados são significativos, frente aos ganhos que a proveniência em tempo real proporciona.

A análise do desempenho do ProvDoop foi feita da seguinte maneira: executamos o experimento inicialmente sem proveniência para todas as 200 entradas variando o número

de núcleos de processamento de 2 a 128. Em seguida, executamos o mesmo experimento, com as mesmas entradas e mesmos ambientes e configurações, agora coletando a proveniência em tempo de execução e armazenando os dados coletados no repositório de proveniência.

No entanto, sem algum tipo de proveniência de dados não é possível avaliar o tempo de execução da atividade e do *workflow* para que a comparação entre as duas abordagens possa ser feita. Quaisquer ferramentas externas que poderiam ser utilizadas para capturar esses tempos poderiam afetar o desempenho, enviesando os resultados. E o painel de monitoramento do Hadoop se torna indisponível assim que as máquinas envolvidas na execução, que podem ser muitas, são terminadas.

Para contornar esse problema, nas execuções sem proveniência de dados inserimos no nó mestre algumas linhas de código que apenas capturam o tempo de início e de fim do *workflow* e de cada atividade, e então escreve essas durações em um arquivo no sistema de arquivos compartilhados.

Pode-se dizer que essa é uma forma básica de proveniência de dados, mas sem ela não seria possível sequer realizar a análise da execução, mostrando quão importante é ter esse tipo de informação sobre as execuções. A Figura 10 mostra um exemplo de arquivo gerado para as execuções sem proveniência de dados.

```
WORKFLOW: SciPhy
TAG: 200x32Cores

MAFFT: 170.34 seconds
READSEQ: 157.986 seconds
MODELGENERATOR: 3593.624 seconds
RAXML: 194.399 seconds
TOTAL: 4116.349 seconds
```

Figura 10 Exemplo de Arquivo Gerado pela Execução sem Proveniência de Dados

O arquivo permitiu então obter os tempos de cada execução sem o ProvDoop. E o repositório de proveniência foi usado para avaliar os resultados de cada execução com ProvDoop. Dessa forma, foi possível realizar comparações entre as duas abordagens.

A seguir, mostramos os resultados obtidos a partir das avaliações experimentais realizadas. A Tabela 1 mostra o tempo total de execução do *workflow* **em horas**, dividindo os resultados de acordo com o número de núcleos de processamento utilizado e com a presença ou não de proveniência de dados. Os resultados sem o ProvDoop estão

na coluna Hadoop e os resultados com a utilização do ProvDoop estão na coluna ProvDoop.

	Hadoop	ProvDoop
2 Núcleos	19,18	19,28
4 Núcleos	8,51	8,64
8 Núcleos	4,37	4,18
16 Núcleos	2,33	2,17
32 Núcleos	1,14	1,34
64 Núcleos	0,81	0,87
128 Núcleos	0,42	0,44

Tabela 1 Tempos Totais das Execuções em Horas

É possível observar que nas execuções com 2, 4, 32, 64 e 128 núcleos, o desempenho sem proveniência (coluna Hadoop) foi melhor do que com proveniência (coluna ProvDoop) diferente das execuções com 8 e 16 núcleos.

Uma das maiores diferenças observadas foi com 16 núcleos de processamento, onde a abordagem com ProvDoop foi cerca de 11,08 minutos mais rápida do que a execução com Hadoop, representando cerca de 4% do tempo de toda a execução do *workflow* para 16 núcleos. Por outro lado, com 32 núcleos de processamento a abordagem com Hadoop se mostrou cerca de 11,85 minutos mais rápida do que a abordagem com ProvDoop, representando cerca de 14% de toda a execução do *workflow* com 32 núcleos.

A Figura 11 mostra o gráfico gerado a partir dos dados da Tabela 1. É possível observar que as execuções com e sem proveniência apresentam o mesmo comportamento.

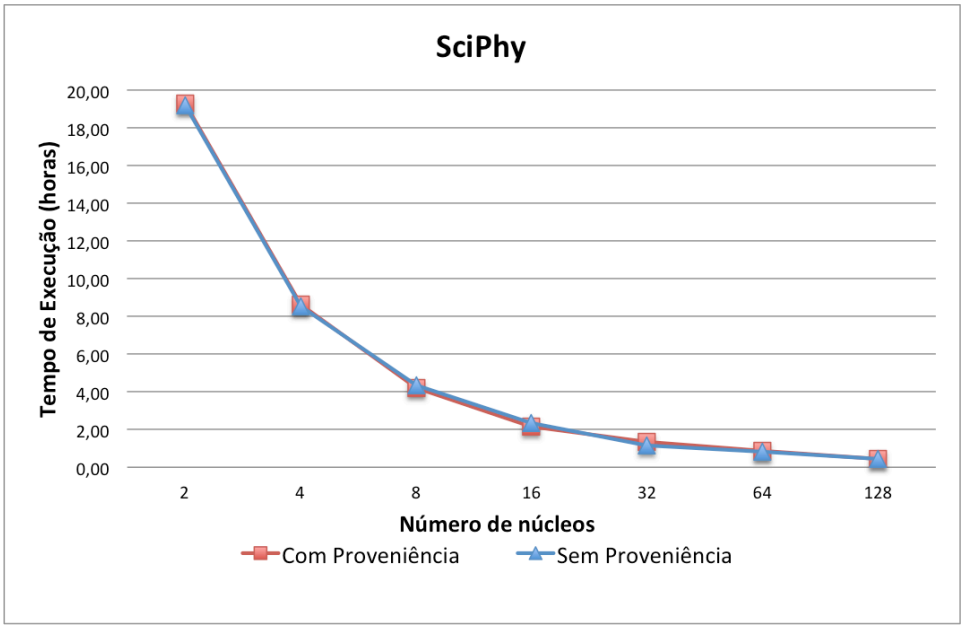

Figura 11 Gráfico com Tempos Totais das Execuções em Horas

Se levarmos em consideração que, como dito anteriormente, as maiores diferenças de tempo são de cerca de 11 minutos de execução, então o gráfico se comporta como esperado, tendo em vista que 11 minutos não são facilmente observáveis em um gráfico com uma escala de quase 20 horas, tempo da primeira execução. O que faz com que os gráficos com proveniência e sem proveniência pareçam completamente sobrepostos.

A Tabela 2 mostra os tempos de execução, **em minutos**, obtidos nos experimentos para cada uma das atividades do *workflow* científico SciPhy e o tempo total obtido, Os tempos são divididos para cada atividade de acordo com o número de núcleos de processamento e se a execução foi realizada sem proveniência ou com proveniência, de maneira análoga à tabela anterior.

	MAFFT		ReadSeq		ModelGenerator		RAxML	
	Hadoop	ProvDoop	Hadoop	ProvDoop	Hadoop	ProvDoop	Hadoop	ProvDoop
2	31,25	33,26	30,88	32,22	984,27	985,92	104,17	105,25
4	**19,64**	**16,23**	15,45	15,89	427,86	438,27	47,64	48,05
8	9,98	9,14	**8,71**	**10,04**	**221,23**	**211,33**	22,04	20,37
16	6,12	4,76	5,70	4,68	122,51	110,67	**5,36**	**10,03**
32	2,84	2,98	2,63	2,70	59,89	68,74	3,24	6,04
64	1,57	2,00	1,83	1,95	43,36	45,21	2,08	3,17
128	2,48	2,59	2,46	2,54	18,83	19,01	1,72	2,07

Tabela 2 Tempos de Execuções para cada Atividade em Minutos

Na tabela estão destacados em negritos os valores onde se observou maior diferença entre as duas abordagens, para cada atividade. Na atividade MAFFT a maior diferença entre os tempos favorece a execução com ProvDoop, com 3,41 minutos a menos para 4 núcleos, representando cerca de 20,10% do tempo de execução da atividade para esse número de núcleos. De maneira análoga, nas atividades ReadSeq, ModelGenerator e RAxML, essas diferenças são, respectivamente, de cerca de 1,34 minutos favorecendo o Hadoop, 11,85 minutos favorecendo o ProvDoop e 4,67 minutos favorecendo o Hadoop. Esses tempos representam 4,15% para o ReadSeq com 8 núcleos, 9,67% para o ModelGenerator com 8 núcleos e 46,56% para o RAxML com 16 núcleos.

A partir dos valores da Tabela 2, a Figura 12 foi criada com os gráficos para cada uma das atividades, mostrando a relação entre o tempo de execução e o número de núcleos de processamento utilizado. Todos os gráficos mostram os resultados com proveniência e sem proveniência, e o tempo de execução é representado em minutos em todos eles.

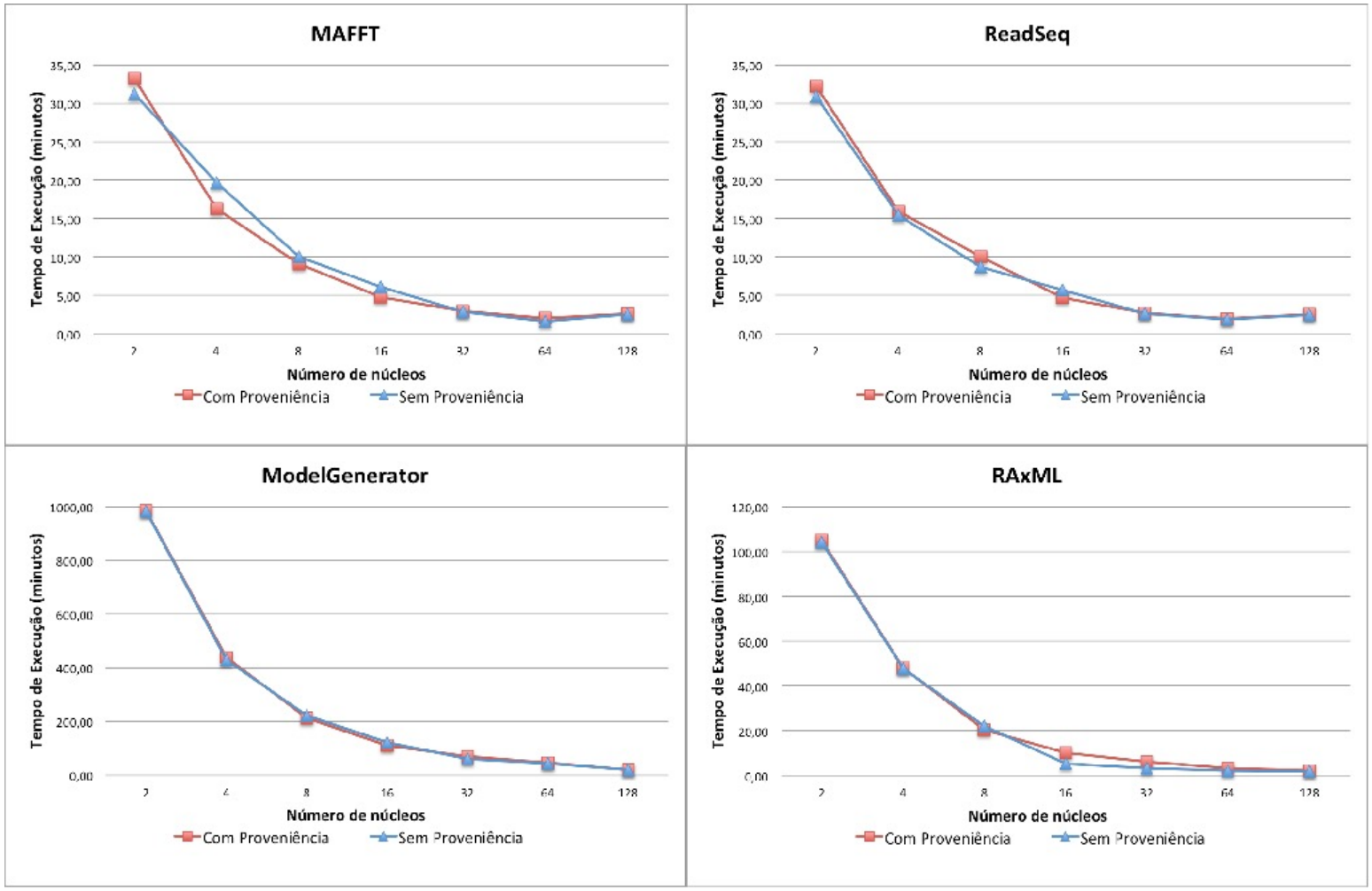

Figura 12 Gráficos com Tempos de Execuções para cada Atividade em Minutos

É possível observar em todas as atividades um comportamento semelhante ao observado na execução do *workflow* como um todo: o comportamento do gráfico com proveniência e sem proveniência segue o mesmo padrão. No entanto, ao analisar cada atividade individualmente, é possível detectar diferenças mais significativas entre as execuções mais curtas, como MAFFT e ReadSeq, de forma que os gráficos não se mostram tão sobrepostos quanto o gráfico do *workflow*. Isso se deve ao fato de que, como essas atividades são consideravelmente mais curtas, uma pequena diferença de tempo se torna significativa o suficiente para ser observada na escala do gráfico. Este é um comportamento que não é observado no ModelGenerator, atividade mais longa do *workflow*.

Observando os dados coletados e os gráficos gerados de cada uma das atividades e do *workflow* completo é possível afirmar que a abordagem com melhor desempenho pode variar, dependendo do cenário selecionado. Em alguns momentos a execução com proveniência é mais rápida e em outros momentos a execução sem proveniência é mais rápida. Ou seja, a adição de proveniência não é decisiva para que o desempenho da execução do *workflow* seja afetado.

Esse comportamento indica que ao inserir proveniência de dados em tempo de execução no Hadoop, o impacto no desempenho não é significante pois as oscilações no tempo de execução devido a variações do ambiente impactam mais no tempo final da execução do que as mudanças causadas pela captura, armazenamento e disponibilização dos dados de proveniência em tempo real. Por variações do ambiente, pode-se entender alocações variáveis de recursos da máquina ou possíveis variações na velocidade de comunicação entre os nós. Essas são variações que todo sistema distribuído e heterogêneo está sujeito a sofrer.

Por fim, a Figura 13 mostra o gráfico de aceleração das execuções do *workflow* para cada uma das abordagens, para cada cenário analisado. É mostrada também a curva que representa a aceleração ideal.

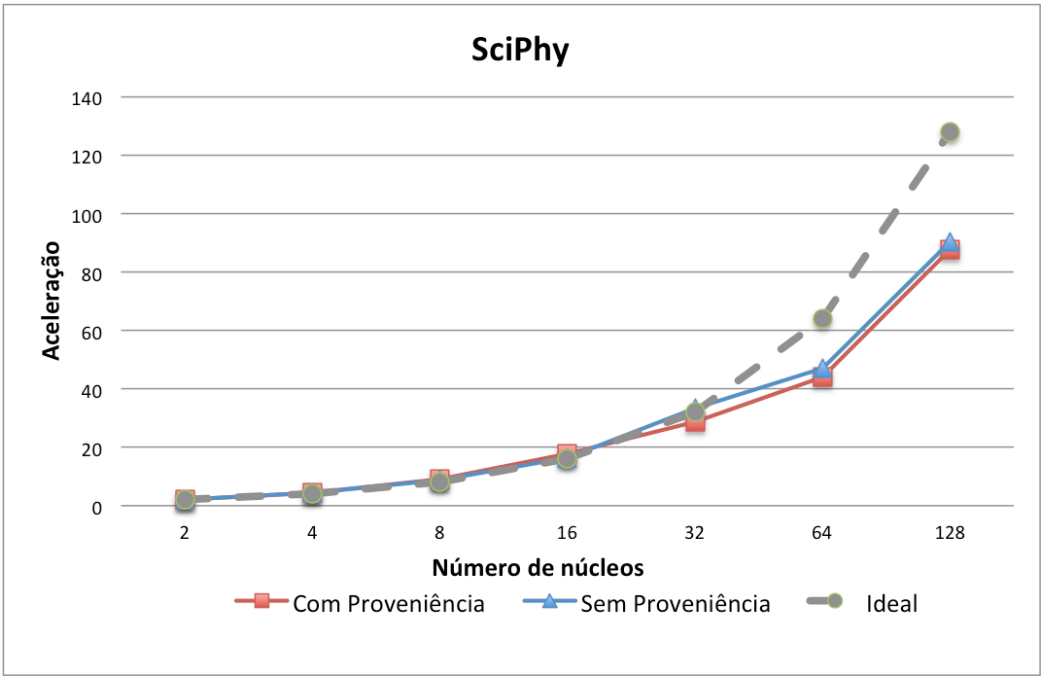

Figura 13 Aceleração do *Workflow* SciPhy

É possível observar que a aceleração é quase ideal entre 2 e 32 núcleos de processamento, e sofreu uma pequena degradação de 64 até 128 núcleos. Além disso, a partir de 32 núcleos de processamento, a aceleração da abordagem com proveniência passa a se manter menor do que a aceleração sem proveniência, com a diferença entre elas sendo de cerca de 14,7% para 32 núcleos, onde observa-se o ponto de maior diferença. Essa diferença pode ser atribuída ao *overhead* de coleta de proveniência, que tende a se tornar mais significativo quanto mais aumentarmos os o número de nós e diminuirmos o tempo de execução.

4.5 Análise de Consultas de Proveniência

Esta seção discute os resultados das diversas execuções do ProvDoop usando o caso de estudos SciPhy materializados sob a forma de consultas baseadas no repositório de proveniência proposto. Isto representa uma das maiores contribuições desta dissertação, tendo em vista que execuções realizadas com Hadoop até então não apresentavam dados de proveniência em tempo de execução acessíveis através de consultas estruturadas a um repositório de proveniência.

As consultas têm como função recuperar dados de proveniência que foram previamente coletados pelo ProvDoop, além de mostrar resultados de consultas em tempo de execução, ou seja, enquanto existem *workflows* em execução no ProvDoop.

As consultas consideradas nesta seção representam apenas uma fração do subconjunto original de consultas que podem ser realizadas utilizando a base de proveniências do ProvDoop. Todas as consultas foram executadas na base de proveniência do ProvDoop e são realizadas como parte dos testes da cobertura do repositório.

As consultas apresentadas a seguir foram elaboradas a partir da necessidade da Dra. Kary Ocaña, especialista em bioinformática que acompanhou a execução dos experimentos apresentados nesta dissertação. Todas as consultas foram formuladas em SQL, uma vez que o repositório de proveniência do ProvDoop é um banco de dados relacional.

Consulta 1: "Recuperar as informações de composição do *workflow* com nome SciPhy e das atividades associadas a ele por ordem crescente de identificador das atividades."

```
SELECT w.tag AS wftag,
       w.description AS wfdescription,
       a.tag AS acttag,
       a.atype AS acttype,
       a.description AS actdescription
FROM cworkflow w INNER JOIN cactivity a
ON w.wkfid = a.wkfid
WHERE w.tag='SciPhy'
ORDER BY a.actid;
```

wftag character varying(200)	wfdescription character varying(100)	acttag character varying(5	acttype character	actdescription character varying(250)
SciPhy	Phylogeny using RAxML -mafft	MAFFT	MAP	Align with MAFFT
SciPhy	Phylogeny using RAxML -mafft	ReadSeq	MAP	Change format with ReadSeq
SciPhy	Phylogeny using RAxML -mafft	ModelGenerator	MAP	Elect model with ModelGenerator
SciPhy	Phylogeny using RAxML -mafft	RAxML	MAP	Construct trees with RAxML

Figura 14 Resultado da Execução da Consulta 1

Essa é possivelmente a consulta mais básica dentre todas as consultas de proveniência em relação à composição, pois ela descreve toda a estrutura que compõe o *workflow*. Essa consulta tem como objetivo a descoberta do conhecimento, tendo em vista que ela auxilia o cientista na descoberta da composição de *workflows* já existentes e que possam ser reaproveitados, ou seja, fomenta o reuso de *workflows* conforme discutido por Mattoso *et al.* (2008, 2009, 2010b).

Consulta 2: "Recuperar a *tag* e a descrição dos *workflows* que tenham atividades em execução no momento, juntamente com informações sobre o estado, o tempo inicial e final e a duração, caso esteja disponível, da execução de cada atividade associada a ele, em ordem crescente de identificador das atividades."

```
SELECT w.tagexec AS wfTag,
ca.tag AS activity,
```

```
a.status, a.starttime, a.endtime,
EXTRACT ('EPOCH' FROM (a.endtime - a.starttime)) AS duration
FROM eworkflow w INNER JOIN eactivity a
ON w.ewkfid = a.wkfid
INNER JOIN cactivity ca
ON ca.actid = a.cactid
WHERE w.ewkfid IN (SELECT w.ewkfid FROM eworkflow w
                   INNER JOIN eactivity a
                   ON w.ewkfid = a.wkfid
                   WHERE a.status='RUNNING') ORDER BY cactid;
```

wftag character vary	activity character varying(5	status character v	starttime timestamp without time zone	endtime timestamp without time zone	duration double pre
200x2Cores	MAFFT	FINISHED	2016-02-08 02:27:52.145	2016-02-08 03:01:07.841	1995.696
200x2Cores	ReadSeq	FINISHED	2016-02-08 03:01:09.071	2016-02-08 03:33:22.06	1932.989
200x2Cores	ModelGenerator	RUNNING	2016-02-08 03:33:22.757		
200x2Cores	RAxML	BLOCKED			

Figura 15 Resultado da Execução da Consulta 2

Essa é uma consulta importante, que a proveniência em tempo de execução permite realizar. Ela mostra quais atividades já terminaram a execução, quais ainda estão em andamento e quais estão bloqueadas, esperando a finalização de alguma outra.

Na Figura 15 observamos que as atividades MAFFT e ReadSeq já terminaram de ser executadas e a atividade ModelGenerator ainda está em execução. O cientista pode então avaliar se a duração de cada uma das atividades executadas está compatível com o esperado e se a atividade em execução não está demorando mais do que deveria.

Esse monitoramento em tempo real permite que o cientista possa interromper a execução caso perceba algum comportamento inesperado, economizando tempo e recursos financeiros.

Consulta 3: "Recuperar todas as ativações geradas pelas atividades em execução, mostrando a *tag* de execução do *workflow*, a *tag* da atividade que está em execução, o diretório onde as saídas de cada ativação estão sendo colocadas, o código de saída da ativação, seu estado, seu momento de início e de fim, caso já esteja disponível, e a duração daquelas ativações que já encerraram a execução."

```
SELECT w.tagexec, ca.tag AS activity,
t.workspace, t.exitstatus, t.status,
t.starttime, t.endtime,
EXTRACT ('EPOCH' FROM (t.endtime - t.starttime)) AS duration
FROM cactivity ca INNER JOIN eactivity a
ON ca.actid = a.cactid
INNER JOIN eworkflow w
ON w.ewkfid = a.wkfid
INNER JOIN eactivation t
```

```
ON t.actid = a.actid
WHERE a.status = 'RUNNING';
```

tagexec character var	activity character varying	workspace character varying(150)	exitstatus integer	status character v	starttime timestamp without time zone	endtime timestamp without time zone	duration double pre
200x2Cores	ModelGenerator	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1378/	0	FINISHED	2016-02-08 09:12:20.821	2016-02-08 09:20:33.979	493.158
200x2Cores	ModelGenerator	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL465/		RUNNING	2016-02-08 16:59:25.621		
200x2Cores	ModelGenerator	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL780/		READY			
200x2Cores	ModelGenerator	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1772/	0	FINISHED	2016-02-08 12:10:20.563	2016-02-08 12:16:54.061	393.498
200x2Cores	ModelGenerator	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1126/	0	FINISHED	2016-02-08 06:29:00.141	2016-02-08 06:34:11.358	311.217
200x2Cores	ModelGenerator	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL874/		READY			
200x2Cores	ModelGenerator	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL659/		READY			
200x2Cores	ModelGenerator	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1147/	0	FINISHED	2016-02-08 07:23:29.103	2016-02-08 07:30:19.527	410.424
200x2Cores	ModelGenerator	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1940/	0	FINISHED	2016-02-08 14:48:02.508	2016-02-08 14:55:17.275	434.767
200x2Cores	ModelGenerator	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1882/	0	FINISHED	2016-02-08 13:49:08.278	2016-02-08 13:57:35.084	506.806

Figura 16 Resultado da Execução da Consulta 3

Essa é uma consulta importante, sobretudo para atividades que tem um tempo de execução maior, como a ModelGenerator. Ela permite que o cientista avalie quais ativações já foram processadas, quais ainda estão aguardando processamento e quais estão em andamento.

Ela é um complemento da consulta anterior, pois permite que o cientista avalie quantas ativações ainda precisam ser executadas e quantas já foram executadas. Com a ajuda da consulta anterior, o cientista pode inferir qual a previsão para que a atividade termine sua execução, assumindo que todas as ativações tenham durações que não divergem muito entre si.

Da mesma forma que na consulta anterior, esse monitoramento em tempo real permite que o cientista avalie o experimento antes mesmo do término dele, podendo analisar os dados de forma mais eficiente e tomar decisões mais rapidamente.

Consulta 4: “Recuperar por ordem crescente de execuções dos *workflows* as *tags* das execuções, o nome de todas as atividades associadas, os momentos de início e término de cada ativação gerada e sua duração, para todas as execuções de atividades já encerradas.”

```
SELECT w.tagexec, ca.tag AS activity,
t.workspace, t.exitstatus, t.status,
t.starttime, t.endtime,
EXTRACT ('EPOCH' FROM (t.endtime - t.starttime)) AS duration
FROM cactivity ca INNER JOIN eactivity a
ON ca.actid = a.cactid
INNER JOIN eworkflow w
ON w.ewkfid = a.wkfid
INNER JOIN eactivation t
ON t.actid = a.actid
WHERE a.status = 'FINISHED'
ORDER BY w.wkfid;
```

tagexec character varying(200	activity charact	workspace character varying(150)	exitstatus integer	status character v	starttime timestamp without time zone	endtime timestamp without time zone	duration double pre
200x32-2Cores	MAFFT	/exp/sciphy.output/provenance-on/200x32-2Cores/ORTHOMCL1009/	0	FINISHED	2016-02-05 02:34:34.239	2016-02-05 02:34:36.521	2.282
200x32-2Cores	MAFFT	/exp/sciphy.output/provenance-on/200x32-2Cores/ORTHOMCL1952/	0	FINISHED	2016-02-05 02:34:32.425	2016-02-05 02:34:34.038	1.613
200x32-2Cores	MAFFT	/exp/sciphy.output/provenance-on/200x32-2Cores/ORTHOMCL918/	0	FINISHED	2016-02-05 02:34:26.974	2016-02-05 02:34:29.32	2.346
200x32-2Cores	MAFFT	/exp/sciphy.output/provenance-on/200x32-2Cores/ORTHOMCL659/	0	FINISHED	2016-02-05 02:34:21.499	2016-02-05 02:34:23.139	1.64
200x32-2Cores	MAFFT	/exp/sciphy.output/provenance-on/200x32-2Cores/ORTHOMCL874/	0	FINISHED	2016-02-05 02:34:15.31	2016-02-05 02:34:17.371	2.061
200x32-2Cores	MAFFT	/exp/sciphy.output/provenance-on/200x32-2Cores/ORTHOMCL787/	0	FINISHED	2016-02-05 02:34:13.844	2016-02-05 02:34:16.012	2.168
200x32-2Cores	MAFFT	/exp/sciphy.output/provenance-on/200x32-2Cores/ORTHOMCL862/	0	FINISHED	2016-02-05 02:34:04.684	2016-02-05 02:34:06.375	1.691
200x32-2Cores	MAFFT	/exp/sciphy.output/provenance-on/200x32-2Cores/ORTHOMCL741/	0	FINISHED	2016-02-05 02:34:00.76	2016-02-05 02:34:02.978	2.218
200x32-2Cores	MAFFT	/exp/sciphy.output/provenance-on/200x32-2Cores/ORTHOMCL885/	0	FINISHED	2016-02-05 02:33:52.972	2016-02-05 02:33:55.111	2.139
200x32-2Cores	MAFFT	/exp/sciphy.output/provenance-on/200x32-2Cores/ORTHOMCL652/	0	FINISHED	2016-02-05 02:33:48.372	2016-02-05 02:33:50.868	2.496

Figura 17 Resultado da Execução da Consulta 4

Essa consulta pode ser considerada análoga à consulta anterior, mas ao considerar atividades que já foram finalizadas ela acaba por fornecer outra informação importante: um histórico de execuções. Através desse histórico, o cientista pode avaliar execuções passadas de atividades e comparar com execuções que estejam em andamento. Isso permite que o cientista já consiga fazer uma análise dos dados sendo gerados e comparar com dados históricos enquanto um *workflow* ainda está sendo executado, tornando a avaliação do experimento mais rápida e mais eficiente.

Consulta 5: "Recuperar as *tags* dos *workflows* em execução, bem como os nomes de suas atividades e seus estados de execução, e as informações referentes a todos os arquivos de saída gerados: seus diretórios, nomes, tamanhos, datas de criação e formatos. Ordenar os resultados por identificador de atividade."

```
SELECT w.tagexec AS wfTag,
ca.tag AS activity, a.status,
f.fdir, f.fname, f.fsize, f.fdate, f.fieldname
FROM eworkflow w INNER JOIN eactivity a
ON w.ewkfid = a.wkfid
INNER JOIN cactivity ca
ON ca.actid = a.cactid
INNER JOIN efile f
ON a.actid = f.actid
WHERE w.ewkfid IN (SELECT w.ewkfid FROM eworkflow w
                   INNER JOIN eactivity a
                   ON w.ewkfid = a.wkfid
                   WHERE a.status='RUNNING')
AND f.type='output'
ORDER BY cactid;
```

wftag character vary	activity charact	status character v	fdir character varying(500)	fname character varying(500)	fsize bigint	fdate timestamp without time zone	fieldname character varying(30)
200x2Cores	MAFFT	FINISHED	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1000/	ORTHOMCL1000.fastaNumbered	4358	2016-02-08 02:28:17.61	FASTA_NUMBERED
200x2Cores	MAFFT	FINISHED	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1000/	ORTHOMCL1000.mafft	5149	2016-02-08 02:28:17.868	MAFFT
200x2Cores	MAFFT	FINISHED	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1021/	ORTHOMCL1021.fastaNumbered	2553	2016-02-08 02:28:27.358	FASTA_NUMBERED
200x2Cores	MAFFT	FINISHED	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1021/	ORTHOMCL1021.mafft	3491	2016-02-08 02:28:27.602	MAFFT
200x2Cores	MAFFT	FINISHED	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1996/	ORTHOMCL1996.fastaNumbered	3475	2016-02-08 02:28:37.436	FASTA_NUMBERED
200x2Cores	MAFFT	FINISHED	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1996/	ORTHOMCL1996.mafft	4041	2016-02-08 02:28:37.692	MAFFT
200x2Cores	MAFFT	FINISHED	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1997/	ORTHOMCL1997.fastaNumbered	3805	2016-02-08 02:28:47.942	FASTA_NUMBERED
200x2Cores	MAFFT	FINISHED	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL1997/	ORTHOMCL1997.mafft	4467	2016-02-08 02:28:48.194	MAFFT
200x2Cores	MAFFT	FINISHED	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL2001/	ORTHOMCL2001.fastaNumbered	3663	2016-02-08 02:28:57.411	FASTA_NUMBERED
200x2Cores	MAFFT	FINISHED	/exp/sciphy.output/provenance-on/200x2Cores/ORTHOMCL2001/	ORTHOMCL2001.mafft	3639	2016-02-08 02:28:57.663	MAFFT

Figura 18 Resultado da Execução da Consulta 5

Essa consulta permite que um cientista saiba exatamente quais arquivos já foram produzidos por um *workflow* que está em andamento. O resultado dessa consulta já permite que o cientista verifique diretamente, por exemplo, se os arquivos gerados são os arquivos esperados e se o tamanho de cada arquivo está compatível com um tamanho de arquivo padrão para aquele formato e para aquele experimento.

Além disso, o caminho completo para o arquivo no sistema de arquivos compartilhados está disponível no resultado da consulta, permitindo que o cientista baixe o arquivo e analise seu conteúdo. Caso algum arquivo fuja do esperado, o cientista conseguirá detectar o problema ainda em tempo de execução, podendo interromper a mesma caso julgue necessário.

Consulta 6: "Recuperar todas as máquinas envolvidas na execução do *workflow* de identificador 32, mostrando o identificador das instâncias, seu tipo, seu nome de servidor e seu endereço de IP."

```
SELECT distinct m.machineid, m.instanceid,
m.type, m.hostname, m.ipaddress
FROM eworkflow w
INNER JOIN eactivity a
ON w.ewkfid = a.wkfid
INNER JOIN eactivation t
ON t.actid = a.actid
INNER JOIN emachine m
ON m.machineid = t.machineid
WHERE w.ewkfid = 32;
```

instanceid character varying(25)	type character v	hostname character varying(255)	ipaddress character varying(
i-778caafe	m1.large	ec2-52-23-199-110.compute-1.amazonaws.com	52.23.199.110
i-86f8a30f	m1.large	ec2-54-209-163-38.compute-1.amazonaws.com	54.209.163.38
i-748caafd	m1.large	ec2-52-23-230-121.compute-1.amazonaws.com	52.23.230.121
i-a797b32e	m1.large	ec2-54-210-226-213.compute-1.amazonaws.com	54.210.226.213
i-768caaff	m1.large	ec2-54-209-11-38.compute-1.amazonaws.com	54.209.11.38
i-b897b331	m1.large	ec2-54-164-140-219.compute-1.amazonaws.com	54.164.140.219
i-87f8a30e	m1.large	ec2-52-90-222-182.compute-1.amazonaws.com	52.90.222.182
i-758caafc	m1.large	ec2-52-23-198-75.compute-1.amazonaws.com	52.23.198.75

Figura 19 Resultado da Execução da Consulta 6

Essa é uma consulta importante, sobretudo no contexto de processamento distribuído na nuvem, onde existem diversas máquinas executando um mesmo *workflow* e cada uma delas fica responsável por uma parte da execução.

Com essa consulta, é possível mapear que máquinas estão executando um *workflow* específico e, caso seja necessário, acessar alguma máquina diretamente, através do nome do servidor ou do endereço IP, para avaliar qualquer problema que esteja ocorrendo com as entradas processadas por aquela máquina.

Além disso, essa consulta, em conjunto com as consultas 3 e 4, permite que o cientista faça um paralelo entre a máquina que executou um conjunto de entradas e o tempo de execução dele, de forma a avaliar de maneira mais embasada que tipo de máquina é mais adequada para cada um dos seus conjuntos de entrada e dos seus experimentos.

Capítulo 5 - Conclusões

Experimentos científicos em larga escala envolvem diversas execuções de *workflows* científicos computacionalmente intensivos. No entanto, uma vez que a execução de um *workflow* pode levar horas, semanas ou até meses, é fundamental que existam meios de o cientista acompanhar a execução enquanto ela está em andamento. Dados de proveniência disponíveis em tempo de execução ajudam o cientista a medida que permitem que o mesmo tome decisões e realize ações avaliando os dados gerados pela aplicação em tempo real, permitindo, por exemplo, interromper a execução tão logo se detecte um erro que prejudicaria todo o andamento do experimento. Sobretudo em um ambiente de nuvem, onde os recursos são pagos a medida que são usados, esse é um aspecto fundamental de economia de tempo e recursos financeiros para o cientista. No entanto, nenhuma das abordagens existentes de execução de *workflows* científicos com Hadoop permite a consulta de dados de proveniência em tempo de execução. Nos melhores cenários, é necessário esperar até o fim da execução para que se possa analisar os dados gerados.

O objetivo principal desta dissertação foi apresentar e avaliar o ProvDoop, uma solução computacional acoplável ao ecossistema do Hadoop de maneira não intrusiva, que captura, armazena e disponibiliza dados de proveniência em tempo de execução para execuções de *workflows* científicos no Hadoop em nuvens computacionais. Esta abordagem utiliza uma arquitetura proposta e componentes desenvolvidos de forma que os dados possam ser acessados pelo cientista em tempo de execução através de consultas a um repositório de proveniência.

No entanto, uma preocupação importante é garantir que ao inserir a proveniência de dados, o desempenho não seja afetado de maneira a tornar impeditiva a execução de *workflows* científicos com proveniência de dados em tempo de execução no Hadoop. Sendo assim, foram realizados diversos experimentos envolvendo um *workflow* real da área de filogenia a fim de avaliar os impactos da adição de proveniência em tempo de execução.

Os resultados mostraram que não há impactos significativos no tempo de execução de um *workflow* científico ao comparar execuções com Hadoop sem proveniência e execuções com ProvDoop. Nos resultados obtidos, pôde-se observar que, em alguns cenários, o ProvDoop realizou a execução do *workflow* em até 4% menos tempo do que o Hadoop,

indicando que as oscilações no tempo de execução devido a variações do ambiente e comunicação entre os nós, podem representar um impacto maior no tempo de execução do que a solução proposta.

A arquitetura do ProvDoop permite a validação e armazenamento dos dados de proveniência de maneira independente de sua execução, através da utilização de um agente de armazenamento que é executado paralelamente ao agente de execução. Este agente de armazenamento permite que os diversos nós escravos envolvidos na execução possam ter seus dados armazenados sem causar impactos significativos ao tempo total de execução de um *workflow* científico.

Diante dos resultados obtidos, pode-se concluir que o ProvDoop atingiu seu objetivo, pois ele se apresenta como uma solução não intrusiva capaz de executar *workflows* científicos com Hadoop na nuvem computacional ao mesmo tempo em que captura, armazena e disponibiliza dados de proveniência em tempo de execução sem causar impactos no tempo total de execução do *workflow*.

5.1 Contribuições

As principais contribuições realizadas pela pesquisa desenvolvida ao longo desta dissertações são:

(i) A concepção e implementação de uma arquitetura acoplável ao Hadoop, de forma não intrusiva, capaz de permitir a execução de *workflows* científicos com Hadoop em nuvens computacionais, ao mesmo tempo em que disponibiliza dados de proveniência em tempo real.

(ii) A concepção e implementação de agentes de execução, de captura e de armazenamento que, em conjunto com a arquitetura proposta, caracterizam o ProvDoop, uma solução capaz de capturar, armazenar e disponibilizar dados de proveniência em tempo real com Hadoop em nuvens computacionais.

(iii) Um modelo de proveniência que permite que o cientista realize consultas e acompanhe a execução de seus *workflows* com Hadoop. Este modelo armazena dados sobre o *workflow* e suas atividades e ativações, bem como sobre as máquinas utilizadas na nuvem e os arquivos consumidos e produzidos na execução.

(iv) Avaliação da solução proposta, por meio de execuções realizadas com um *workflow* real de filogenia, que mostrou que o ProvDoop é capaz de inserir proveniência de dados em tempo real no Hadoop sem impactar significativamente o tempo de execução do *workflow*.

Os resultados obtidos indicaram que o uso da solução proposta nesta dissertação melhora a confiabilidade e reprodutibilidade de experimentos científicos pelo uso da proveniência em tempo real, sem causar impactos negativos no tempo de execução do mesmo. Pelo contrário, os impactos causados são positivos, tendo em vista que passa a ser possível detectar e corrigir erros na execução de um *workflow* científico com Hadoop mais rapidamente e eficientemente, diminuindo o tempo total do experimento e, consequentemente, os custos atrelados a utilização de recursos na nuvem.

5.2 Limitações

Toda pesquisa científica possui determinadas limitações. Desta maneira, algumas limitações foram identificadas, principalmente no que tange o desenvolvimento do ProvDoop, a partir de uma análise criteriosa da prova de conceito implementada, dos *workflows* executados e dos dados de proveniência:

(i) Por questões de simplificação na implementação, diversos componentes do ProvDoop estão hoje com implementação dependente da API da *Amazon Web Services*, bem como de seus componentes. Idealmente, os componentes devem ser independentes do ambiente, mas este cenário ainda está distante. O ambiente da *Amazon Web* Services se mostrou o mais estável, completo e confiável para que pudéssemos realizar os experimentos.

(ii) Para executar diferentes *workflows* científicos utilizando o ProvDoop, é necessário realizar alterações nas funções *Map* e *Reduce* para refletirem as atividades envolvidas no novo *workflow*. Para que isso não seja mais necessário, é preciso construir um módulo de especificação do *workflow*, que deve ler e interpretar um arquivo que representa o *workflow* científico a ser executado e então preparar os *Maps* e os *Reduces* de acordo com os valores obtidos do arquivo.

(iii) Como observado na análise do desempenho, quando o número de nós envolvidos na execução aumenta muito, diminuindo o tempo de execução para apenas alguns minutos, o *overhead* causado pela inicialização dos componentes e pela comunicação dos mesmos começa a impactar o tempo de execução do *workflow*

científico. Portanto, pode-se concluir que o ProvDoop é uma solução ideal para *workflows* científicos que demore ao menos algumas horas para ser executado e esses de fato são os *workflows* científicos que mais se beneficiam da proveniência de dados em tempo de execução.

5.3 Trabalhos Futuros

A abordagem utilizada nesta dissertação representa ganhos para a comunidade científica e abre caminho para trabalhos futuros relacionadas à pesquisa realizada, mas que fogem do escopo tratado.

Para realizar a execução de diferentes *workflows* com ProvDoop, é necessário que o cientista realize pequenas alterações nos *Maps* e *Reduces* desenvolvidos para cada atividade. Então um trabalho futuro importante é realizar uma abstração dos *workflows* científicos de maneira que o cientista consiga elaborar uma especificação de mais alto nível, informando as particularidades do *workflow* de maneira simples e o ProvDoop consiga, a partir de tal especificação, realizar a execução.

Podem-se também realizar diferentes pesquisas relacionadas à proveniência de dados com Hadoop, principalmente no que tange a captura de dados específicos de domínio. Idealmente deve existir uma forma de analisar os produtos de dados das diversas ativações do *workflow* e extrair dos arquivos produzidos informações importantes referentes a conhecimento específico do domínio do *workflow*, levando em consideração a complexidade adicionada pela execução em nuvens computacionais.

Outros trabalhos futuros importantes são abordagens em relação à detecção e identificação de erros, a fim de realizar re-execuções de maneira automática com ProvDoop, sem a necessidade de intervenção do cientista.

Referências Bibliográficas

AKOUSH, S.; SOHAN, R.; HOPPER, A. HadoopProv: Towards Provenance As a First Class Citizen in MapReduce. Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance., TaPP '13. p.11:1–11:4, 2013. Berkeley, CA, USA: USENIX Association.

DE ALMEIDA-NETO, C.; LIU, J.; WRIGHT, D. J.; et al. 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, 2011.

ALTINTAS, I.; BERKLEY, C.; JAEGER, E.; et al. Kepler: an extensible system for design and execution of scientific workflows. 16th International Conference on Scientific and Statistical Database Management, 2004. Proceedings. . p.423–424, 2004.

Amazon Web Services (AWS) − Serviços de computação em nuvem. .

AMSTERDAMER, Y.; DAVIDSON, S. B.; DEUTCH, D.; et al. Putting Lipstick on Pig: Enabling Database-style Workflow Provenance. **Proc. VLDB Endow.**, v. 5, n. 4, p. 346–357, 2011.

ANDERSON, A. C. The Process of Structure-Based Drug Design. **Chemistry & Biology**, v. 10, n. 9, p. 787–797, 2003.

ANDERSON, D. P.; COBB, J.; KORPELA, E.; LEBOFSKY, M.; WERTHIMER, D. SETI@Home: An Experiment in Public-resource Computing. **Commun. ACM**, v. 45, n. 11, p. 56–61, 2002.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; et al. A View of Cloud Computing. **Commun. ACM**, v. 53, n. 4, p. 50–58, 2010.

BARKER, A.; VAN HEMERT, J. Scientific workflow: a survey and research directions. **Parallel Processing and Applied Mathematics**. p.746–753, 2007. Springer.

BEBERG, A. L.; ENSIGN, D. L.; JAYACHANDRAN, G.; KHALIQ, S.; PANDE, V. S. Folding@home: Lessons from eight years of volunteer distributed computing. IEEE International Symposium on Parallel Distributed Processing, 2009. IPDPS 2009. . p.1–8, 2009.

BERTINO, E.; BERNSTEIN, P.; AGRAWAL, D.; et al. Challenges and Opportunities with Big Data. , Cyber Center Publications., 2011.

BEVERIDGE, W. I. B. The art of scientific investigation. , p. xii+171 pp., 1950.

BORKAR, V.; CAREY, M. J.; LI, C. Inside "Big Data Management": Ogres, Onions, or Parfaits? Proceedings of the 15th International Conference on Extending Database Technology. , EDBT '12. p.3–14, 2012. New York, NY, USA: ACM.

BRANDT, J.; BUX, M.; LESER, U. Cuneiform: a Functional Language for Large Scale Scientific Data Analysis. EDBT/ICDT Workshops. . p.7–16, 2015.

BUX, M.; BRANDT, J.; LIPKA, C.; et al. SAASFEE: Scalable Scientific Workflow Execution Engine. **Proc. VLDB Endow.**, v. 8, n. 12, p. 1892–1895, 2015.

CALLAHAN, S. P.; FREIRE, J.; SANTOS, E.; et al. VisTrails: Visualization Meets Data Management. Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. , SIGMOD '06. p.745–747, 2006. New York, NY, USA: ACM.

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

CAVALCANTI, M. C.; TARGINO, R.; BAIÃO, F.; et al. Managing structural genomic workflows using web services. **Data & Knowledge Engineering**, v. 53, n. 1, p. 45–74, 2005.

CIRNE, W.; BRASILEIRO, F.; ANDRADE, N.; et al. Labs of the World, Unite!!! **Journal of Grid Computing**, v. 4, n. 3, p. 225–246, 2006.

COSTA, F. **Heurísticas para Controle de Re-Execução Paralela de Workflows Científicos em Nuvens de Computadores**, 2012. Universidade Federal do Rio de Janeiro.

COSTA, F.; OLIVEIRA, D. DE; OCAÑA, K. A. C. S.; OGASAWARA, E.; MATTOSO, M. Enabling Re-executions of Parallel Scientific Workflows Using Runtime Provenance Data. In: P. Groth; J. Frew (Orgs.); **Provenance and Annotation of Data and Processes**, Lecture Notes in Computer Science. p.229–232, 2012. Springer Berlin Heidelberg.

COSTA, F.; SILVA, V.; DE OLIVEIRA, D.; et al. Capturing and Querying Workflow Runtime Provenance with PROV: A Practical Approach. Proceedings of the Joint

EDBT/ICDT 2013 Workshops. , EDBT '13. p.282–289, 2013. New York, NY, USA: ACM.

COUTINHO, F.; OGASAWARA, E.; OLIVEIRA, D.; et al. Many task computing for orthologous genes identification in protozoan genomes using Hydra. **Concurrency and Computation: Practice and Experience**, v. 23, n. 17, p. 2326–2337, 2011.

CRAWL, D.; WANG, J.; ALTINTAS, I. Provenance for MapReduce-based Data-intensive Workflows. Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science. , WORKS '11. p.21–30, 2011. New York, NY, USA: ACM.

CRUZ, S. M. S. **Uma estratégia de apoio à gerência de dados de proveniência em experimentos científicos**, 2011. Ph. D. Thesis, Federal University of Rio de Janeiro-COPPE, Brazil.

CRUZ, S. M. S. DA; CAMPOS, M. L. M.; MATTOSO, M. Towards a taxonomy of provenance in scientific workflow management systems. Services-I, 2009 World Conference on. . p.259–266, 2009. IEEE.

DALMAN, T.; WIECHERT, W.; NÖH, K. A scientific workflow framework for 13C metabolic flux analysis. **Journal of Biotechnology**.

DANTAS, M. Clusters Computacionais. **Computação Distribuída de Alto Desempenho: Redes, Clusters e Grids Computacionais**, v. 1, p. 145–180, 2005a.

DANTAS, M. Grids Computacionais. **Computação Distribuída de Alto Desempenho: Redes, Clusters e Grids Computacionais**, v. 1, p. 201–238, 2005b.

DAVIDSON, S. B.; FREIRE, J. Provenance and Scientific Workflows: Challenges and Opportunities. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. , SIGMOD '08. p.1345–1350, 2008. New York, NY, USA: ACM.

DÁVILA, A. M.; MENDES, P. N.; WAGNER, G.; et al. ProtozoaDB: dynamic visualization and exploration of protozoan genomes. **Nucleic acids research**, v. 36, n. suppl 1, p. D547–D552, 2008.

DEAN, J.; GHEMAWAT, S. MapReduce: Simplified Data Processing on Large Clusters. **Commun. ACM**, v. 51, n. 1, p. 107–113, 2008.

DEAN, J.; GHEMAWAT, S. MapReduce: A Flexible Data Processing Tool. **Commun. ACM**, v. 53, n. 1, p. 72–77, 2010.

DEELMAN, E.; GANNON, D.; SHIELDS, M.; TAYLOR, I. Workflows and e-Science: An overview of workflow system features and capabilities. **Future Generation Computer Systems**, v. 25, n. 5, p. 528–540, 2009.

DEELMAN, E.; MEHTA, G.; SINGH, G.; SU, M.-H.; VAHI, K. Pegasus: Mapping Large-Scale Workflows to Distributed Resources. In: I. J. Taylor; E. Deelman; D. B. Gannon; M. Shields (Orgs.); **Workflows for e-Science**. p.376–394, 2007. Springer London.

DEELMAN, E.; VAHI, K.; JUVE, G.; et al. Pegasus, a Workflow Management System for Science Automation. **Future Gener. Comput. Syst.**, v. 46, n. C, p. 17–35, 2015.

DE OLIVEIRA, D.; CUNHA, L.; TOMAZ, L.; PEREIRA, V.; MATTOSO, M. Using ontologies to support deep water oil exploration scientific workflows. Services-I, 2009 World Conference on. . p.364–367, 2009. IEEE.

DIAS, J. F. **Execução Interativa de Experimentos Científicos Computacionais em Larga Escala**, 2013. Universidade Federal do Rio de Janeiro.

DIAS, J.; OGASAWARA, E.; DE OLIVEIRA, D.; et al. Algebraic dataflows for big data analysis. 2013 IEEE International Conference on Big Data. . p.150–155, 2013.

DIAS, J.; OGASAWARA, E.; DE OLIVEIRA, D.; et al. Supporting Dynamic Parameter Sweep in Adaptive and User-steered Workflow. Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science. , WORKS '11. p.31–36, 2011. New York, NY, USA: ACM.

DITTRICH, J.; QUIANÉ-RUIZ, J.-A.; JINDAL, A.; et al. Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing). **Proc. VLDB Endow.**, v. 3, n. 1-2, p. 515–529, 2010.

DITTRICH, J.; QUIANÉ-RUIZ, J.-A.; RICHTER, S.; et al. Only Aggressive Elephants are Fast Elephants. **arXiv:1208.0287 [cs]**, 2012.

EVSUKOFF, A. G.; DE LIMA, B. S.; EBECKEN, N. F. Long-term runoff modeling using rainfall forecasts with application to the Iguaçu River Basin. **Water resources management**, v. 25, n. 3, p. 963–985, 2011.

FAHRINGER, T.; PRODAN, R.; DUAN, R.; et al. ASKALON: A Grid Application Development and Computing Environment. Proceedings of the 6th IEEE/ACM

International Workshop on Grid Computing. , GRID ’05. p.122–131, 2005. Washington, DC, USA: IEEE Computer Society.

FERREIRA, J. L. R. **Migração do Chiron para ambiente de processamento paralelo com memória distribuida**, 2014. Universidade Federal do Rio de Janeiro.

FILETO, R.; LIU, L.; PU, C.; ASSAD, E. D.; MEDEIROS, C. B. POESIA: An ontological workflow approach for composing Web services in agriculture. **The VLDB Journal—The International Journal on Very Large Data Bases**, v. 12, n. 4, p. 352–367, 2003.

FLORATOU, A.; TELETIA, N.; DEWITT, D. J.; PATEL, J. M.; ZHANG, D. Can the Elephants Handle the NoSQL Onslaught? **Proc. VLDB Endow.**, v. 5, n. 12, p. 1712–1723, 2012.

FOSTER, I.; KESSELMAN, C. **The Grid 2: Blueprint for a New Computing Infrastructure**. Elsevier, 2003.

FREIRE, J.; KOOP, D.; SANTOS, E.; SILVA, C. T. Provenance for Computational Tasks: A Survey. **Computing in Science & Engineering**, v. 10, n. 3, p. 11–21, 2008.

GARTNER. **Magic Quadrant for Public Cloud Storage Services, Worldwide**. 2015.

GILBERT, D. Sequence File Format Conversion with Command-Line Readseq. **Current Protocols in Bioinformatics**, 2002. John Wiley & Sons, Inc.

GLAVIC, B. Big Data Provenance: Challenges and Implications for Benchmarking. In: T. Rabl; M. Poess; C. Baru; H.-A. Jacobsen (Orgs.); **Specifying Big Data Benchmarks**, Lecture Notes in Computer Science. p.72–80, 2014. Springer Berlin Heidelberg.

GOBLE, C.; WROE, C.; STEVENS, R. The myGrid project: services, architecture and demonstrator. Proc. of the UK e-Science All Hands Meeting. . p.595–602, 2003.

GOECKS, J.; NEKRUTENKO, A.; TAYLOR, J.; OTHERS. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. **Genome Biol**, v. 11, n. 8, p. R86, 2010.

GOLDMAN, A.; KON, F.; JUNIOR, F. P.; POLATO, I.; DE FÁTIMA PEREIRA, R. Apache Hadoop: conceitos teóricos e práticos, evolução e novas possibilidades. **XXXI Jornadas de atualizaçoes em informatica**, 2012.

GONCALEZ, T. T.; SABINO, E. C.; CAPUANI, L.; et al. Blood transfusion utilization and recipient survival at Hospital das Clinicas in Sao Paulo, Brazil. **Transfusion**, v. 52, n. 4, p. 729–738, 2012.

GOVERNATO, F.; BROOK, C.; MAYER, L.; et al. Bulgeless dwarf galaxies and dark matter cores from supernova-driven outflows. **Nature**, v. 463, n. 7278, p. 203–206, 2010.

GUERRA, G. M.; ROCHINHA, F. A Sparse Grid Method Applied to Stochastic Fluid-Structure Interaction. **COBEM, 2009, Gramado, Brazil**, 2009.

GUERRA, G.; ROCHINHA, F. A.; ELIAS, R.; et al. Uncertainty quantification in computational predictive models for fluid dynamics using a workflow management engine. **International Journal for Uncertainty Quantification**, v. 2, n. 1, 2012.

GUERRA, G.; ROCHINHA, F.; ELIAS, R.; et al. Scientific workflow management system applied to uncertainty quantification in large eddy simulation. Congresso Ibero Americano de Métodos Computacionais em Engenharia. . p.1–13, 2009.

GUGNANI, S.; KISS, T. Extending Scientific Workflow Systems to Support MapReduce Based Applications in the Cloud. 2015 7th International Workshop on Science Gateways (IWSG). . p.16–21, 2015.

HARTMAN, A. L.; RIDDLE, S.; MCPHILLIPS, T.; LUDÄSCHER, B.; EISEN, J. A. Introducing WATERS: a workflow for the alignment, taxonomy, and ecology of ribosomal sequences. **BMC bioinformatics**, v. 11, n. 1, p. 1, 2010.

HASHEM, I. A. T.; YAQOOB, I.; ANUAR, N. B.; et al. The rise of "big data" on cloud computing: Review and open research issues. **Information Systems**, v. 47, p. 98–115, 2015.

HEGER, D. Hadoop performance tuning-a pragmatic & iterative approach. **CMG Journal**, v. 4, p. 97–113, 2013.

HERODOTOU, H.; LIM, H.; LUO, G.; et al. Starfish: A Self-tuning System for Big Data Analytics. CIDR. . v. 11, p.261–272, 2011.

HEY, T. The Fourth Paradigm – Data-Intensive Scientific Discovery. In: S. Kurbanoğlu; U. Al; P. L. Erdoğan; Y. Tonta; N. Uçak (Orgs.); **E-Science and Information Management**, Communications in Computer and Information Science. p.1–1, 2012. Springer Berlin Heidelberg.

HOFFA, C.; MEHTA, G.; FREEMAN, T.; et al. On the Use of Cloud Computing for Scientific Workflows. IEEE Fourth International Conference on eScience, 2008. eScience ’08. . p.640–645, 2008.

HULL, D.; WOLSTENCROFT, K.; STEVENS, R.; et al. Taverna: a tool for building and running workflows of services. **Nucleic Acids Research**, v. 34, n. suppl 2, p. W729–W732, 2006.

IKEDA, R.; PARK, H.; WIDOM, J. Provenance for Generalized Map and Reduce Workflows. , 2010. Stanford InfoLab.

INTERLANDI, M.; SHAH, K.; TETALI, S. D.; et al. Titian: Data Provenance Support in Spark. **Proc. VLDB Endow.**, v. 9, n. 3, p. 216–227, 2015.

ISLAM, M.; HUANG, A. K.; BATTISHA, M.; et al. Oozie: Towards a Scalable Workflow Management System for Hadoop. Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies. , SWEET ’12. p.4:1–4:10, 2012. New York, NY, USA: ACM.

ISLAM, M. K.; SRINIVASAN, A. **Apache Oozie: The Workflow Scheduler for Hadoop**. O’Reilly Media, Inc., 2015.

JARRARD, R. D. Scientific methods. **Online book, URL: http:// emotionalcompetency. com/ sci/ booktoc. html**, 2001.

JINDAL, A.; QUIANÉ-RUIZ, J.-A.; DITTRICH, J. Trojan data layouts: right shoes for a running elephant. Proceedings of the 2nd ACM Symposium on Cloud Computing. . p.21, 2011. ACM.

JOSHI, S. B. Apache Hadoop Performance-tuning Methodologies and Best Practices. Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering. , ICPE ’12. p.241–242, 2012. New York, NY, USA: ACM.

JR, H. G. G. **Scientific Method in Practice**. Cambridge University Press, 2003.

JURISTO, N.; MORENO, A. M. **Basics of software engineering experimentation**. Springer Science & Business Media, 2013.

KEANE, T. M.; CREEVEY, C. J.; PENTONY, M. M.; NAUGHTON, T. J.; MCLNERNEY, J. O. Assessment of methods for amino acid matrix selection and their

use on empirical data shows that ad hoc assumptions for choice of matrix are not justified. **BMC Evolutionary Biology**, v. 6, p. 29, 2006.

KIM, W.; KIM, S. D.; LEE, E.; LEE, S. Adoption issues for cloud computing. Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia. . p.2–5, 2009. ACM.

LANDER, E. S.; LINTON, L. M.; BIRREN, B.; et al. Initial sequencing and analysis of the human genome. **Nature**, v. 409, n. 6822, p. 860–921, 2001.

LEMOS, M.; CASANOVA, M. A.; SEIBEL, L. F. B.; DE MACEDO, J. A. F.; DE MIRANDA, A. B. Ontology-driven workflow management for biosequence processing systems. Database and Expert Systems Applications. . p.781–790, 2004. Springer.

LENGAUER, T.; MANNHOLD, R.; KUBINYI, H.; TIMMERMAN, H. Bioinformatics–from genomes to drugs. , 2002.

LI, M.; ZENG, L.; MENG, S.; et al. MRONLINE: MapReduce Online Performance Tuning. Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing. , HPDC '14. p.165–176, 2014. New York, NY, USA: ACM.

LINS, E. F.; ELIAS, R. N.; GUERRA, G. M.; ROCHINHA, F. A.; COUTINHO, A. L. 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, 2009.

LIU, J.; PACITTI, E.; VALDURIEZ, P.; MATTOSO, M. A Survey of Data-Intensive Scientific Workflow Management. **Journal of Grid Computing**, v. 13, n. 4, p. 457–493, 2015.

LUDÄSCHER, B.; ALTINTAS, I.; BERKLEY, C.; et al. Scientific workflow management and the Kepler system. **Concurrency and Computation: Practice and Experience**, v. 18, n. 10, p. 1039–1065, 2006.

MARINOS, A.; BRISCOE, G. Community cloud computing. **Cloud Computing**. p.472–484, 2009. Springer.

MARTINHO, W.; OGASAWARA, E.; OLIVEIRA, D.; et al. A conception process for abstract workflows: an example on deep water oil exploitation domain. 5th IEEE International Conference on e-Science. , 2009.

MATSUNAGA, A.; TSUGAWA, M.; FORTES, J. CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications. IEEE Fourth International Conference on eScience, 2008. eScience ’08. . p.222–229, 2008.

MATTOSO, M.; OCAÑA, K.; HORTA, F.; et al. User-steering of HPC workflows: state-of-the-art and future directions. Proceedings of the 2nd ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies. . p.4, 2013. ACM.

MATTOSO, M.; WERNER, C.; TRAVASSOS, G. H.; et al. 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, 2010.

MELL, P.; GRANCE, T. The NIST definition of cloud computing. , 2011.

MISSIER, P.; SOILAND-REYES, S.; OWEN, S.; et al. Taverna, reloaded. Scientific and Statistical Database Management. . p.471–481, 2010. Springer.

MOREAU, L.; CLIFFORD, B.; FREIRE, J.; et al. The Open Provenance Model core specification (v1.1). **Future Generation Computer Systems**, v. 27, n. 6, p. 743–756, 2011.

MOREAU, L.; FREIRE, J.; FUTRELLE, J.; et al. The Open Provenance Model: An Overview. In: J. Freire; D. Koop; L. Moreau (Orgs.); **Provenance and Annotation of Data and Processes**, Lecture Notes in Computer Science. p.323–326, 2008. Springer Berlin Heidelberg.

MOREAU, L.; LUDÄSCHER, B.; ALTINTAS, I.; et al. Special Issue: The First Provenance Challenge. **Concurrency and Computation: Practice and Experience**, v. 20, n. 5, p. 409–418, 2008.

MOREAU, L.; MISSIER, P.; BELHAJJAME, K.; et al. The PROV data model and abstract syntax notation. **W3C Working Draft.(Work in progress.)**, 2011.

MURTA, L.; BRAGANHOLO, V.; CHIRIGATI, F.; KOOP, D.; FREIRE, J. noWorkflow: Capturing and Analyzing Provenance of Scripts. In: B. Ludäscher; B. Plale (Orgs.); **Provenance and Annotation of Data and Processes**, Lecture Notes in Computer Science. p.71–83, 2014. Springer International Publishing.

NAPPER, J.; BIENTINESI, P. Can cloud computing reach the top500? Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop. . p.17–20, 2009. ACM.

OCANA, K. A. C. S.; DE OLIVEIRA, D.; DIAS, J.; OGASAWARA, E.; MATTOSO, M. Optimizing Phylogenetic Analysis Using SciHmm Cloud-based Scientific Workflow. 2011 IEEE 7th International Conference on E-Science (e-Science). . p.62–69, 2011.

OCAÑA, K. A. C. S.; OLIVEIRA, D. DE; OGASAWARA, E.; et al. SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes. In: O. N. de Souza; G. P. Telles; M. Palakal (Orgs.); **Advances in Bioinformatics and Computational Biology**, Lecture Notes in Computer Science. p.66–70, 2011. Springer Berlin Heidelberg.

OGASAWARA, E.; DIAS, J.; OLIVEIRA, D.; et al. An algebraic approach for data-centric scientific workflows. **Proc. of VLDB Endowment**, v. 4, n. 12, p. 1328–1339, 2011.

OGASAWARA, E.; DIAS, J.; SILVA, V.; et al. Chiron: a parallel engine for algebraic scientific workflows. **Concurrency and Computation: Practice and Experience**, v. 25, n. 16, p. 2327–2341, 2013.

OLIVEIRA, D. **Uma Abordagem de Apoio à Execução Paralela de Workflows Científicos em Nuvens de Computadores**, 2012. Universidade Federal do Rio de Janeiro.

OLIVEIRA, D.; BAIÃO, F. A.; MATTOSO, M. Towards a taxonomy for cloud computing from an e-science perspective. **Cloud Computing**. p.47–62, 2010. Springer.

OLIVEIRA, D.; COSTA, F.; SILVA, V.; OCAÑA, K.; MATTOSO, M. Debugging Scientific Workflows with Provenance: Achievements and Lessons Learned. , 2014.

OLIVEIRA, D. DE; OCAÑA, K. A. C. S.; BAIÃO, F.; MATTOSO, M. A Provenance-based Adaptive Scheduling Heuristic for Parallel Scientific Workflows in Clouds. **Journal of Grid Computing**, v. 10, n. 3, p. 521–552, 2012.

OLIVEIRA, D.; OCAÑA, K. A. C. S.; OGASAWARA, E.; et al. Performance evaluation of parallel strategies in public clouds: A study with phylogenomic workflows. **Future Generation Computer Systems**, Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and

Scientific Applications — Big Data, Scalable Analytics, and Beyond., v. 29, n. 7, p. 1816–1825, 2013.

OLIVEIRA, D.; OCANA, K.; OGASAWARA, E.; et al. A Performance Evaluation of X-Ray Crystallography Scientific Workflow Using SciCumulus. 2011 IEEE International Conference on Cloud Computing (CLOUD). . p.708–715, 2011.

OLIVEIRA, D.; OGASAWARA, E.; BAIÃO, F.; MATTOSO, M. SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows. 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD). . p.378–385, 2010.

OLIVEIRA, D.; OGASAWARA, E.; OCAÑA, K.; BAIÃO, F.; MATTOSO, M. An adaptive parallel execution strategy for cloud-based scientific workflows. **Concurrency and Computation: Practice and Experience**, v. 24, n. 13, p. 1531–1550, 2012.

OLIVEIRA, W.; NEVES, V. C.; OCAÑA, K.; et al. Captura e Consulta a Dados de Proveniência Retrospectiva Implícita Intra-Atividade. .

OLSTON, C.; REED, B.; SRIVASTAVA, U.; KUMAR, R.; TOMKINS, A. Pig Latin: A Not-so-foreign Language for Data Processing. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. , SIGMOD '08. p.1099–1110, 2008. New York, NY, USA: ACM.

PARK, H.; IKEDA, R.; WIDOM, J. RAMP: A System for Capturing and Tracing Provenance in MapReduce Workflows. , 2011. Seattle, Washington: Stanford InfoLab.

PATAVINO, G. M.; DE ALMEIDA-NETO, C.; LIU, J.; et al. 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, 2012.

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

PINTAS, J. T. **Monitoramento em Tempo Real de Workflows Científicos Executados em Paralelo em Ambientes Distribuídos**, 2012. Universidade Federal do Rio de Janeiro.

PORTO, F.; MOURA, A. M.; SILVA, F. C.; et al. A metaphoric trajectory data warehouse for Olympic athlete follow-up. **Concurrency and Computation: Practice and Experience**, v. 24, n. 13, p. 1497–1512, 2012.

PRUITT, K. D.; TATUSOVA, T.; KLIMKE, W.; MAGLOTT, D. R. NCBI Reference Sequences: current status, policy and new initiatives. **Nucleic Acids Research**, v. 37, n. suppl 1, p. D32–D36, 2009.

SABINO, E. C.; GONÇALEZ, T. T.; CARNEIRO-PROIETTI, A. B.; et al. Human immunodeficiency virus prevalence, incidence, and residual risk of transmission by transfusions at Retrovirus Epidemiology Donor Study-II blood centers in Brazil. **Transfusion**, v. 52, n. 4, p. 870–879, 2012.

SANDHOLM, T.; LAI, K. MapReduce Optimization Using Regulated Dynamic Prioritization. Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems. , SIGMETRICS '09. p.299–310, 2009. New York, NY, USA: ACM.

SHI, L.; WANG, Z.; YU, W.; MENG, X. Performance Evaluation and Tuning of BioPig for Genomic Analysis. Proceedings of the 2015 International Workshop on Data-Intensive Scalable Computing Systems. , DISCS '15. p.9:1–9:7, 2015. New York, NY, USA: ACM.

SILVA, E.; OGASAWARA, E.; OLIVEIRA, D.; BENEVIDES, M.; MATTOSO, M. Especificação Formal e Verificação de Workflows Científicos. **IV e-Science**, 2010.

SILVA, V.; OLIVEIRA, D.; MATTOSO, M. SciCumulus 2.0: Um Sistema de Gerência de Workflows Científicos para Nuvens Orientado a Fluxo de Dados. **Sessão de Demos do XXIX Simpósio Brasileiro de Banco de Dados**, 2014.

SIMMHAN, Y.; VAN INGEN, C.; SUBRAMANIAN, G.; LI, J. Bridging the gap between desktop and the cloud for escience applications. Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. . p.474–481, 2010. IEEE.

SOANES, C.; STEVENSON, A. **Oxford Dictionary of English**. Oxford University Press, Incorporated, 2003.

STAMATAKIS, A. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. **Bioinformatics**, v. 22, n. 21, p. 2688–2690, 2006.

STONEBRAKER, M.; ABADI, D.; DEWITT, D. J.; et al. MapReduce and Parallel DBMSs: Friends or Foes? **Commun. ACM**, v. 53, n. 1, p. 64–71, 2010.

TAYLOR, I. J.; DEELMAN, E.; GANNON, D. B.; SHIELDS, M. **Workflows for e-Science: Scientific Workflows for Grids**. Springer Publishing Company, Incorporated, 2014.

TAYLOR, I.; SHIELDS, M.; WANG, I.; HARRISON, A. The Triana Workflow Environment: Architecture and Applications. In: I. J. Taylor; E. Deelman; D. B. Gannon; M. Shields (Orgs.); **Workflows for e-Science**. p.320–339, 2007. Springer London.

TAYLOR, R. C. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. **BMC Bioinformatics**, v. 11, n. 12, p. 1–6, 2010.

TRAVASSOS, G. H.; BARROS, M. O. Contributions of in virtuo and in silico experiments for the future of empirical studies in software engineering. 2nd Workshop on Empirical Software Engineering the Future of Empirical Studies in Software Engineering. . p.117–130, 2003.

VALDURIEZ, P.; PACITTI, E. Data Management in Large-Scale P2P Systems. In: M. Daydé; J. Dongarra; V. Hernández; J. M. L. M. Palma (Orgs.); **High Performance Computing for Computational Science - VECPAR 2004**, Lecture Notes in Computer Science. p.104–118, 2004. Springer Berlin Heidelberg.

VAQUERO, L. M.; RODERO-MERINO, L.; CACERES, J.; LINDNER, M. A Break in the Clouds: Towards a Cloud Definition. **SIGCOMM Comput. Commun. Rev.**, v. 39, n. 1, p. 50–55, 2008.

VARIA, J. Architecting for the cloud: Best practices. **Amazon Web Services**, 2011.

VARIA, J.; MATHEW, S. Overview of amazon web services. **Jan-2014**, 2013.

WANG, J.; CRAWL, D.; ALTINTAS, I. Kepler + Hadoop: A General Architecture Facilitating Data-intensive Applications in Scientific Workflow Systems. Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science. , WORKS '09. p.12:1–12:8, 2009. New York, NY, USA: ACM.

WANG, J.; CRAWL, D.; PURAWAT, S.; NGUYEN, M.; ALTINTAS, I. Big data provenance: Challenges, state of the art and opportunities. 2015 IEEE International Conference on Big Data (Big Data). . p.2509–2516, 2015.

WANG, L.; TAO, J.; KUNZE, M.; et al. Scientific Cloud Computing: Early Definition and Experience. HPCC. . v. 8, p.825–830, 2008.

WHITE, T. **Hadoop: The Definitive Guide**. O'Reilly Media, Inc., 2012.

WOZNIAK, J. M.; ARMSTRONG, T. G.; WILDE, M.; et al. Swift/T: large-scale application composition via distributed-memory dataflow processing. Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on. . p.95–102, 2013. IEEE.

XUAN, P.; ZHENG, Y.; SARUPRIA, S.; APON, A. SciFlow: A dataflow-driven model architecture for scientific computing using Hadoop. 2013 IEEE International Conference on Big Data. . p.36–44, 2013.

YOUSEFF, L.; BUTRICO, M.; DA SILVA, D. Toward a unified ontology of cloud computing. Grid Computing Environments Workshop, 2008. GCE'08. . p.1–10, 2008. IEEE.

ZHAO, Y.; HATEGAN, M.; CLIFFORD, B.; et al. Swift: Fast, Reliable, Loosely Coupled Parallel Computation. 2007 IEEE Congress on Services. . p.199–206, 2007.

ZHAO, Y.; LI, Y.; RAICU, I.; et al. Enabling scalable scientific workflow management in the Cloud. **Future Generation Computer Systems**, v. 46, p. 3–16, 2015.

ZHAO, Y.; LI, Y.; RAICU, I.; et al. A Service Framework for Scientific Workflow Management in the Cloud. **IEEE Transactions on Services Computing**, v. 8, n. 6, p. 930–944, 2015.

ZHU, J.; WANG, W. New-knowledge-view based ontology cloud model. Computer Science and Software Engineering, 2008 International Conference on. . v. 5, p.1140–1143, 2008. IEEE.

ZVELEBIL, M.; BAUM, J. **Understanding Bioinformatics**. Garland Science, 2007.