MONITORAMENTO EM TEMPO REAL DE *WORKFLOWS* CIENTÍFICOS EXECUTADOS EM PARALELO EM AMBIENTES DISTRIBUÍDOS

Julliano Trindade Pintas

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

Orientadora: Marta Lima de Queirós Mattoso

Rio de Janeiro

Outubro de 2012

MONITORAMENTO EM TEMPO REAL DE *WORKFLOWS* CIENTÍFICOS EXECUTADOS EM PARALELO EM AMBIENTES DISTRIBUÍDOS

Julliano Trindade Pintas

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

Examinada por:

_____

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

_____

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

_____

Prof[a]. Flávia Coimbra Delicato, D.Sc.

_____

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

RIO DE JANEIRO, RJ – BRASIL
OUTUBRO DE 2012

Pintas, Julliano Trindade

Monitoramento em Tempo Real de *Workflows* Científicos Executados em Paralelo em Ambientes Distribuídos/ Julliano Trindade Pintas – Rio de Janeiro: UFRJ/COPPE, 2012.

XI, 68 p.: il.; 29,7 cm.

Orientadora: Marta Lima de Queirós Mattoso

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

Referências Bibliográficas: p. 62-68.

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

Aos meus pais Joandyr e Claudia,

que são a base de tudo na minha vida.

À Deus por tornar tudo possível;

Aos meus irmãos Julliana e Joandyr, que são de extrema importância na minha vida;

À minha namorada Thaís, que sempre esteve ao meu lado cuidando de mim e me fazendo feliz;

Aos meus familiares e amigos, que foram totalmente compreensivos e me apoiaram mesmo nos momentos que eu necessitei me ausentar para realizar atividades do mestrado.

À Marta Mattoso, por ser uma Orientadora extraordinária e ter me dado a oportunidade de ser seu orientando;

Ao Daniel Oliveira que sempre me acompanhou de perto durante todo trabalho desta dissertação, se mostrando como um profissional altamente competente,

Ao Alexandre Assis, Flavia Delicato e Daniel de Oliveira por terem aceitado participar da banca do meu mestrado;

Ao Eduardo Ogasawara, Kary Ocaña, Jonas Dias, Vitor Gamboa e Anderson Marinho que contribuíram diretamente com o meu mestrado.

Aos professores da UNIRIO e da Coppe/UFRJ, que contribuíram diretamente com a minha formação;

À Ana Rabello, Cláudia Prata, Juliana Beltrami, Maria Mercedes, Natália Prata, Patrícia Leal, Solange Santos, Sônia Galliano, Gutierrez, que sempre me ajudaram com as questões administrativas;

Agradeço.

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

MONITORAMENTO EM TEMPO REAL DE *WORKFLOWS* CIENTÍFICOS EXECUTADOS EM PARALELO EM AMBIENTES DISTRIBUÍDOS

Julliano Trindade Pintas

Outubro/2012

Orientadores: Marta Lima de Queirós Mattoso

Programa: Engenharia de Sistemas e Computação

A maioria dos *workflows* científicos de larga escala apresenta execução de longa duração, tornando inviável para o cientista monitorar o estado da execução durante todo o tempo em um terminal. Nesta dissertação, apresentamos uma nova abordagem para monitoramento em tempo real de *workflows* científicos executados em paralelo, baseado em consultas aos dados de proveniência gerados em tempo real, que identifica eventos pré-configurados e notifica o cientista por meio de tecnologias de dispositivos móveis e redes sociais. A avaliação da solução proposta, intitulada SciLightning, foi realizada por meio do monitoramento da execução em paralelo do *workflow* de análise filogenética chamado SciPhy no ambiente de nuvem Amazon EC2 usando a máquina de execução de *workflows* em nuvem SciCumulus. A avaliação demonstrou que a abordagem proposta é eficaz no que tange ao monitoramento e notificação de eventos, e pode ser acoplada a soluções para a gerência de execução de *workflows* e que a notificação de eventos do *workflow* em tempo real é fundamental, uma vez que permite ajustes finos de parâmetros de execução de forma *online*.

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

REAL TIME MONITORING OF SCIENTIFIC WORKFLOWS PARALLEL EXECUTION ON DISTRIBUTED ENVIRONMENTS

Julliano Trindade Pintas

October/2012

Advisors: Marta Lima de Queirós Mattoso

Department: Systems and Computer Engineering

Most large-scale scientific workflows presents long duration execution, making it impractical for scientists to monitor the running status all the time through a terminal. In this dissertation, we present a new approach for real-time monitoring of scientific workflows executed in parallel, based on queries to data provenance generated in real time, identifying preconfigured events and notifying scientist through mobile devices and social networks technologies. The evaluation of the proposed solution, called SciLightning, was performed by monitoring the parallel execution of the phylogenetic analysis workflow called SciPhy in the Amazon EC2 cloud environment using the workflows cloud execution engine SciCumulus. The evaluation demonstrated that the proposed approach is effective with respect to monitoring and event notification, and can be easily coupled to management execution solutions of workflows that event notification workflow in real time is critical, since allows fine tuning of execution parameters online.

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1 - Introdução

Muitos experimentos científicos são baseados em simulações computacionais complexas que consomem e produzem um grande volume de dados e utilizam grandes quantidades de recursos computacionais. À medida que um experimento se torna cada vez mais complexo (seja em termos do número de vezes em que suas atividades são executadas ou do volume de dados a ser processado), gerenciar tais simulações torna-se um desafio. Como solução, os *workflows* científicos são aplicados com o intuito de apoiar a gerência de recursos envolvidos em simulações computacionais de larga escala. Um *workflow* pode ser definido como uma abstração que permite a composição de diversas atividades em um fluxo estruturado (I. J. Taylor, Deelman, Gannon, e Shields 2007), onde cada atividade corresponde à invocação de um programa ou serviço e as dependências representam o fluxo de dados entre elas.

Em alguns casos, esses *workflows* científicos precisam ser executados 100.000 vezes ou mais, variando-se os valores de seus parâmetros de entrada. Desse modo, é fundamental que estes *workflows* sejam executados utilizando técnicas de paralelismo em ambientes de Processamento de Alto Desempenho (PAD). 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; Beberg et al. 2009; Cirne et al. 2006), as redes ponto a ponto (do inglês *peer-to-peer* ou P2P) (Valduriez e Pacitti 2005) e mais recentemente as nuvens de computadores (D. Oliveira, Baião, e Mattoso 2010; Vaquero et al. 2009).

Existem soluções para execução distribuída de *workflows* em larga escala que visam prover o paralelismo necessário, tais como o Sistema de Gerência de *Workflows* Científicos (SGWfC) Swift (Zhao et al. 2007), implementações do modelo MapReduce (Dean e Ghemawat 2008) e o SciCumulus (Oliveira et al. 2011), sendo este último focado em paralelismo de *workflows* estritamente em nuvens de computadores (Vaquero et al. 2009).

Como os *workflows* em larga escala tendem a executar por semanas ou até mesmo meses dependendo do volume de dados a ser processado e de quão computacionalmente intensivo é cada programa do *workflow*, mecanismos de monitoramento são fundamentais para que os cientistas se mantenham informados a

respeito do estado da execução. Entretanto, não é trivial monitorar a execução de *workflows* científicos em ambientes distribuídos (Gil et al. 2007) permitindo o acompanhamento e a análise parcial dos resultados em pontos pré-determinados. Este acompanhamento e análise parcial são passos iniciais para prover mecanismos de condução do *workflow* (do inglês *workflow steering*) (Dias et al. 2011). Esses mecanismos são fundamentais para avaliar os dados produzidos e para oferecer maior controle na execução dos *workflows* possibilitando modificar seu curso durante a execução. Toda a avaliação realizada pelos cientistas é baseada nos dados de proveniência do *workflow* (Freire et al. 2008).

Entretanto, esse tipo de monitoramento ainda não é uma realidade nas abordagens existentes. Em sistemas como o Swift, ou implementações do MapReduce como o Hadoop(Abouzeid et al. 2009), não é possível consultar os dados de proveniência em tempo real (durante a execução), tornando os mecanismos de condução do *workflow* difíceis de serem implementados. Outras abordagens, como o SciCumulus, já proveem os dados de proveniência em tempo real. Entretanto não apresentam mecanismos de monitoramento. No SciCumulus, o cientista necessita realizar consultas periódicas ao repositório de proveniência por meio de sistemas de gerência de bancos de dados (SGBD) ou de *logs* para conseguir acompanhar a execução de *workflows* científicos, o que não é desejável pois é trabalhoso e tendencioso ao erro.

1.1 Caracterização do Problema

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

"Como permitir o monitoramento em tempo real de execuções de workflows em ambientes distribuídos durante longos períodos de tempo, baseado em definições explícitas do cientista sobre os eventos que lhe interessa, de uma maneira que seja confiável, aplicável a várias abordagens de execução e que não impacte o desempenho da execução do workflow?"

1.2 Hipótese: Automatização do processo de monitoramento

A hipótese geral desta dissertação é que **SE** for possível automatizar o monitoramento da execução do *workflow* de acordo com parâmetros pré-definidos pelo cientista e notificá-lo através de tecnologias de dispositivos móveis e redes sociais, **ENTÃO** será

possível realizar o acompanhamento em tempo real e consequentemente reduzir o tempo total do experimento.

Desse modo, o objetivo principal desta dissertação é propor uma nova abordagem no que tange ao monitoramento da execução de *workflows* em larga escala, baseando-se em duas premissas: a automatização do processo de monitoramento por meio da identificação de eventos que sejam importantes para o cientista e o envio imediato de notificações dos eventos definidos por um meio de comunicação que seja conveniente. A ideia principal é de que toda a informação que o cientista necessita saber sobre o estado de execução do *workflow* seja enviada automaticamente assim que for gerada, não sendo mais necessário que o cientista fique periodicamente pesquisando por tais informações em um terminal. Assim, além de oferecer um meio mais conveniente de efetuar o monitoramento, esta nova abordagem proporciona um acompanhamento em tempo real, tornando as ações de condução do *workflow* mais rápidas e menos custosas.

Para que seja possível automatizar o processo de identificação de eventos do *workflow* que sejam importantes para o cientista, é necessário realizar um levantamento e categorizar os principais eventos que poderiam ocorrer durante sua execução. Portanto, também é contribuição desta dissertação o levantamento junto aos cientistas dos principais tipos de eventos e respectivas modalidades de notificação. Para tanto, foram levantados os parâmetros necessários para identificar cada tipo de evento e respectivos formatos e conteúdos das notificações.

1.3 SciLightning

Para avaliar a abordagem proposta, foi concebido e implementado o SciLightning, um arcabouço de monitoramento em tempo real de execução de *workflows* científicos que pode ser acoplado às soluções para execução de *workflows* existentes. O SciLightning permite que o cientista defina uma série de parâmetros que são utilizados pelo próprio SciLightning para realizar análise automática dos dados de proveniência gerados em tempo real e notificar o cientista através de tecnologias de dispositivos móveis e redes sociais. Desta forma, o cientista se mantém informado a respeito do estado da execução de seus *workflows* e pode realizar análise parcial dos resultados para, em seguida, executar ações de condução do *workflow*.

Foram adotados alguns princípios básicos que guiaram a definição arquitetural do SciLightning de forma que a solução possa ser facilmente adaptada para ser utilizada

em outros domínios da ciência e ser acoplada a outras abordagens de execução de *workflows*. Os princípios básicos foram: permitir a adaptação a qualquer outra abordagem de execução, facilitar a inclusão dos novos tipos de notificação bem como alteração dos já definidos, minimizar o impacto no desempenho de execução do *workflow* em monitoramento e possibilitar a adoção de outras plataformas de dispositivos móveis e de redes sociais.

As redes sociais e aplicações de dispositivos móveis foram escolhidas como meios complementares de comunicação. Uma rede social se mostra mais prática do que uma aplicação de dispositivo móvel, pois pode ser acessada através de qualquer dispositivo que possua um browser e acesso a internet, como computadores pessoais convencionais, alguns modelos de televisão e até dos próprios dispositivos móveis. Também é possível aproveitar as funcionalidades nativas da rede social, como o envio de *e-mail* automático no recebimento de uma mensagem e compartilhamento de mensagens entre usuários. Por outro lado, a aplicação móvel é mais flexível, pois não sofre das restrições normalmente impostas pela utilização das redes sociais, como o limite máximo de tamanho da mensagem e taxa máxima de envio. Além disso, existe a possibilidade de organizar as notificações de uma maneira mais conveniente e oferecer uma interface personalizada.

O SciLightning foi projetado para ser acoplado a qualquer abordagem de execução que forneça dados de proveniência em tempo real. Nesta dissertação, para avaliar o funcionamento do SciLightning, foi realizado o acoplamento com o SciCumulus, que se trata de uma abordagem de execução de *workflows* projetada para distribuir e controlar a execução paralela de *workflows* científicos em um ambiente de nuvem. O projeto do SciLightning permite também a utilização de qualquer plataforma de aplicações móveis e de redes sociais para que sejam desenvolvidos os mecanismos de envio de mensagem. Nesta dissertação, esses mecanismos foram implementados através de uma aplicação da plataforma *Android*[1] e um componente integrado com a ferramenta de rede social e *microblogging Twitter*[2].

---

[1] http://www.android.com/

[2] http://www.twitter.com/

Para avaliar sua capacidade de monitoramento de execução de *workflows* em paralelo em ambientes distribuídos, o SciLightning foi submetido a uma série de testes e a um experimento final. Tanto os testes quanto o experimento final se basearam no monitoramento da execução em paralelo do *workflow* SciPhy (Ocaña, Oliveira, Ogasawara, et al. 2011), de análise filogenética, no ambiente de nuvem Amazon EC2 usando a abordagem de execução SciCumulus (D. Oliveira et al. 2011).

1.4 Organização da Dissertação

Além desta introdução, esta dissertação é organizada como segue. O Capítulo 2 apresenta conceitos relacionados a experimentos científicos, *workflows* científicos e proveniência de dados. Neste mesmo capítulo também são apresentados os trabalhos relacionados e o SciCumulus, a abordagem de execução utilizada como estudo de caso nesta dissertação. O Capítulo 3 apresenta o SciLightning, uma abordagem desenvolvida para monitoramento em tempo real da execução de *workflows* científicos executados em paralelo em ambientes distribuídos. O Capítulo 4 apresenta a série de avaliações que o SciLightning foi submetido, sendo a principal delas uma avaliação experimental realizada através do monitoramento de uma execução completa do *workflow* SciPhy que durou cerca de 6,3 dias sendo executado pelo SciCumulus paralelamente por 16 máquinas virtuais no ambiente de nuvem Amazon EC2. Por último, o Capítulo 5 conclui a dissertação apresentando os principais resultados alcançados e os desdobramentos possíveis para trabalhos futuros.

Capítulo 2 - *Workflow* Científico em Larga Escala

Um experimento científico pode ser definido como um teste que visa observar um fenômeno de interesse sob condições controladas para demonstrar a verdade, examinar a validade de uma hipótese ou determinar a eficácia de algo anteriormente não experimentado (Jarrard 2001). Antes do surgimento dos computadores, a maioria dos experimentos científicos era conduzida de duas maneiras principais, por meio de um organismo vivo ou por meio de componentes separados do organismo em um ambiente controlado. Esses tipos de experimentos são chamados de experimentos *in vivo* e *in vitro*, respectivamente.

Com a evolução da computação, muitos experimentos começaram a ser conduzidos por meio de simulações computacionais, originando duas novas classes de experimentos: (i) Experimento *in virtuo*: quando um objeto de pesquisa real é observado em um ambiente simulado (ii) Experimento *in silico*: tanto o ambiente quanto o objeto de estudo são simulados através da computação.

O tipo de experimento *in silico*, que é o foco desta dissertação, é aplicado em vários domínios científicos como, por exemplo, análises filogenéticas (Ocaña, Oliveira, Ogasawara, et al. 2011), genômica comparativa (Ocaña, Oliveira, Dias, et al. 2011), processamento de sequências biológicas (Lemos et al. 2004), estudos na área de saúde (de Almeida-Neto et al. 2011; Goncalez et al. 2011; Patavino et al. 2012; Sabino et al. 2011), prospecção de petróleo em águas profundas (Carvalho 2009; Martinho et al. 2009; Ogasawara et al. 2011; D. Oliveira et al. 2009), mapeamento dos corpos celestes (Hey, Tansley, e Tolle 2009), ecologia (Hartman et al. 2010), agricultura (Fileto et al. 2003), busca de genes ortólogos dos tripanosomas causadores de doenças tropicais negligenciadas (Coutinho et al. 2011; Dávila et al. 2008), dinâmica de fluidos computacional (G. M. Guerra e Rochinha 2009a; G. M. Guerra e Rochinha 2009b; G. M. Guerra e Rochinha 2010; G. Guerra et al. 2009; G. Guerra et al. 2012; Lins et al. 2009), estudos fisiológicos (Porto et al. 2011), previsão de precipitação (Evsukoff, Lima, e Ebecken 2011), monitoramento aquático (Pereira e Ebecken 2011) e pesquisa sobre energia escura (Governato et al. 2010).

Os resultados dos experimentos *in silico* são obtidos por meio da execução de uma sequência de programas sobre um conjunto inicial de dados, sendo que o conjunto

de dados de saída de cada programa é utilizado como entrada para os programas imediatamente seguintes na sequência. Muitos experimentos desse tipo são considerados de larga escala por serem compostos por simulações computacionalmente intensivas e que consomem e produzem um grande volume de dados e utilizam grandes quantidades de recursos computacionais.

Além de uma única execução desta sequência de programas demandar um grande volume de recursos computacionais, um único experimento pode executar a mesma sequência de programas centenas ou milhares de vezes, até que sua hipótese seja confirmada ou refutada. De uma execução para outra, podem ser alterados os dados de entrada ou os parâmetros de configuração. Neste contexto, são necessárias técnicas de paralelismo e uma infraestrutura de alto desempenho, como *clusters*, supercomputadores, grades computacionais, ambientes de computação voluntária e mais recentemente as nuvens de computadores.

2.1 *Workflow* Científico

À medida que um experimento se torna cada vez mais complexo (seja pelo número de vezes em que suas atividades são executadas ou volume de dados a ser processado), gerenciar tais simulações torna-se um desafio. Como solução, os *workflows* científicos são aplicados com o intuito de apoiar a gerência de recursos envolvidos em simulações computacionais de larga escala. Um *workflow* pode ser definido como uma abstração que permite a composição de diversas atividades em um fluxo estruturado (I. J. Taylor, Deelman, Gannon, e Shields 2007), onde cada atividade corresponde à invocação de um programa e as dependências representam o fluxo de dados entre elas. Uma atividade executada em ambiente distribuído pode ser dividida em várias tarefas a serem executadas em paralelo. Cada tarefa é a menor unidade de trabalho que pode ser processada em paralelo. Este conceito é apresentado aqui como tarefa para se obter um maior grau de generalização, mas dependendo da abordagem podem existir conceitos específicos e semelhantes, como *Cloud Activity*(Daniel Oliveira 2012) e ativação (Ogasawara et al. 2011).

Esses *workflows* são controlados e executados pelos Sistemas de Gerência de *Workflows* Científicos (SGWfC), que são mecanismos complexos que visam apoiar a configuração e execução dos *workflows*. Há muitos SGWfC disponíveis, como o Kepler (Altintas et al. 2004), o Taverna (Hull et al. 2006), o VisTrails (Callahan et al. 2006), o

Pegasus (Deelman et al. 2007), o Askalon (Fahringer et al. 2005), o P-Grade (Glatard et al. 2007; Kertész, Sipos, e Kacsuk 2007), o DagMan (Couvares et al. 2007), o Triana (I. Taylor, Shields, Wang, e Harrison 2007), o WOODS (Medeiros et al. 2005) e o Swift (Zhao et al. 2007), cada um com suas próprias características, vantagens e desvantagens.

2.2 Ciclo de Vida do Experimento Científico

Experimentos científicos são realizados através da execução de uma série de ensaios (do inglês *trials*). No caso dos experimentos executados por meio de *workflows* científicos, cada ensaio é realizado através de uma série de atividades relacionadas com uma execução distinta de um *workflow* científico. Essas atividades são agrupadas em fases que por sua vez compõem o ciclo de vida do experimento. É importante conhecer um modelo de ciclo de experimento para que seja possível identificar em qual fase se encaixa o trabalho proposto nesta dissertação. A Figura 1 apresenta um modelo de ciclo de vida proposto por Mattoso *et al.* (2010), que divide o experimento em três fases principais: composição, execução e análise.

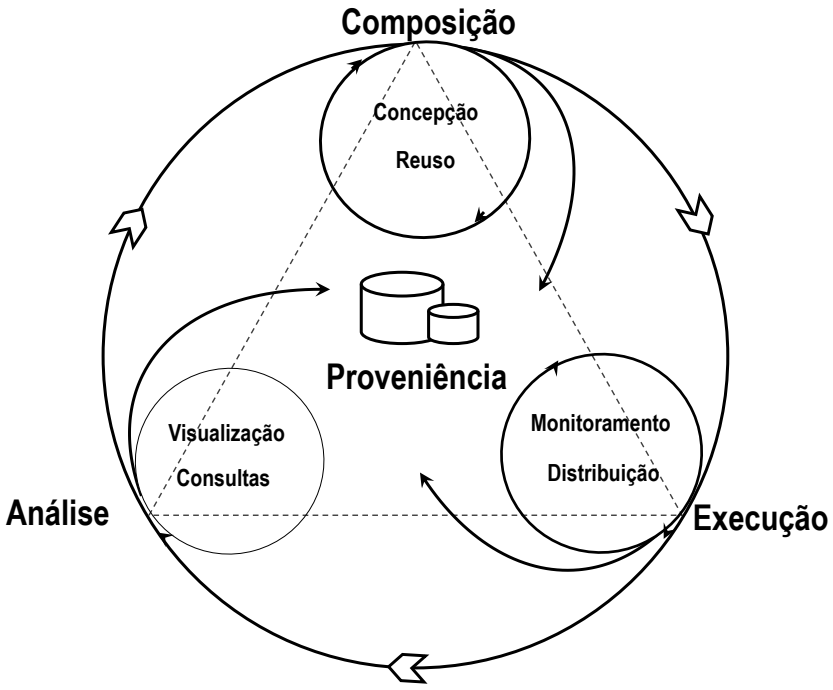

Figura 1 Ciclo de vida do experimento científico Oliveira (2012)

É na fase de composição que todo o experimento científico é modelado e estruturado por meio da definição do sequenciamento lógico entre suas atividades, que podem ser executadas tanto pelo próprio cientista quanto por um programa de

computador. Para cada atividade devem ser definidos os parâmetros e tipos de dados de entrada e de saída. Essa fase pode ser decomposta em duas subfases: concepção e reutilização. Na subfase de concepção, o *workflow* é modelado por meio de informações levantadas sobre o experimento, chamado de protocolo do experimento (Juristo e Moreno 2010). Já na subfase de reutilização, o *workflow* científico é modelado tendo como base outros *workflows* já existentes. A reutilização pode ocorrer tanto pela adaptação de um *workflow* quanto pela aplicação de partes (também chamados de componentes) de outros *workflows* já definidos.

Na fase de execução, dado o conjunto de dados de entrada e de parâmetros, as atividades são executadas de acordo com o sequenciamento definido na fase de composição, gerando assim um conjunto de dados de saída. Esse conjunto de dados de saída é o resultado final da fase de execução e serve como entrada principal para fase de análise. Para obter um conjunto de dados de saída confiável em um tempo razoável é necessário realizar o gerenciamento desta execução. Esta fase pode ser dividida em duas subfases: distribuição e monitoramento (Mattoso et al. 2010). A subfase de distribuição trata questões relacionadas com a necessidade da execução de atividades do *workflow* em ambientes distribuídos que muitas vezes se torna necessário para se obter o resultado em tempo viável. A subfase de monitoramento, que é o foco principal desta dissertação, trata de questões relacionadas à necessidade de conhecimento constante sobre o estado atual da execução do *workflow* pelo cientista, uma vez que este pode executar por um longo tempo. Todo trabalho desenvolvido nesta dissertação tem como objetivo principal apoiar a fase de execução, facilitando especificamente a subfase de monitoramento.

Por último, é na fase de análise que os dados gerados pelas fases de composição e execução são analisados para verificar se são satisfatórios e suficientes para confirmar ou refutar sua hipótese. Essa fase é altamente dependente dos dados de Proveniência (Freire et al. 2008), também chamada de histórico do experimento (i.e. sua definição, seus dados produzidos, etc.), que foram gerados nas fases anteriores. O processo de análise realiza um conjunto de consultas à base de proveniência e aos resultados com possíveis visualizações gráficas para facilitar o entendimento do conjunto de dados obtidos. Por tal motivo, a fase de análise pode ser divida em duas que recebem esses nomes. Mesmo que uma análise indique inicialmente que a hipótese foi confirmada, deve-se repetir a execução do *workflow* para garantir que não houve falhas em sua execução. Por este motivo, é importante que os dados de proveniência que foram

utilizados na análise sejam preservados para garantir a reprodutibilidade do experimento.

A proveniência tem um papel fundamental durante todo ciclo de vida do experimento. É importante que a proveniência seja registrada e tratada da maneira correta nas fases de composição e execução para que seja possível realizar a análise e garantir que o experimento seja reprodutível. Ou seja, todas as fases do ciclo de vida manipulam de alguma forma os dados de proveniência. Devido a esses fatores, os principais conceitos relacionados ao registro e tratamento da proveniência serão apresentados na próxima seção.

2.3 Proveniência do Experimento científico

Todas as informações provenientes das atividades envolvidas durante as fases do ciclo de vida do experimento científico, como sua definição e dados utilizados e gerados durante sua execução, são chamadas de proveniência do experimento. Além de ser essencial para realizar fase de análise do experimento, a proveniência é extremamente importante para garantir que ele possa ser reproduzido. Normalmente, para que o resultado de um experimento seja aceito pela comunidade científica, o mesmo deve ser passível de reprodução por outros grupos de pesquisadores utilizando o método originalmente utilizado.

Buneman *et al.* (2001) definiram a questão de proveniência de dados em experimentos científicos como a descrição da origem de um dado e o processo pelo qual este chegou a um banco de dados. Goble *et al.* (2003) resumem as diversas funcionalidades para as informações de proveniência da seguinte maneira: (i) garantia de qualidade dos dados: informações de proveniência podem ser utilizadas para estimar a qualidade e a confiabilidade dos dados baseando-se na origem dos dados e suas transformações; (ii) auditoria dos caminhos: os dados de proveniência podem traçar rotas dos dados, determinar a utilização de recursos e detectar erros na geração de dados; (iii) verificação de atribuição: mantém controle sobre as informações do dono do experimento e seus dados. Também permite a citação e atribuem responsabilidades em caso de dados errados e; (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.

2.4 SciCumulus

O SciCumulus é a abordagem de execução de *workflows* na qual acoplamos o SciLightning nesta dissertação. O SciCumulus é uma a máquina de execução de *workflows* projetada para distribuir e controlar a execução paralela de *workflows* científicos despachadas a partir de um SGWfC em um ambiente de nuvem, como o Amazon EC2 (Oliveira 2012). O SciCumulus orquestra a execução das atividades em um conjunto distribuído de máquinas virtuais (VM) (criando um *cluster* virtual), oferecendo dimensionamento dos recursos durante o curso de execução do *workflow*. O SciCumulus foi projetado para seguir uma arquitetura de quatro camadas (Figura 3):

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

(ii) Camada de Distribuição: esta camada cria e gerencia a distribuição de *cloud activities* em uma ou mais máquinas virtuais instanciadas em um ambiente de nuvem. Seus componentes podem ser instalados em qualquer ambiente, mas preferencialmente na nuvem para diminuir o impacto de comunicação com os componentes da camada de execução;

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

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

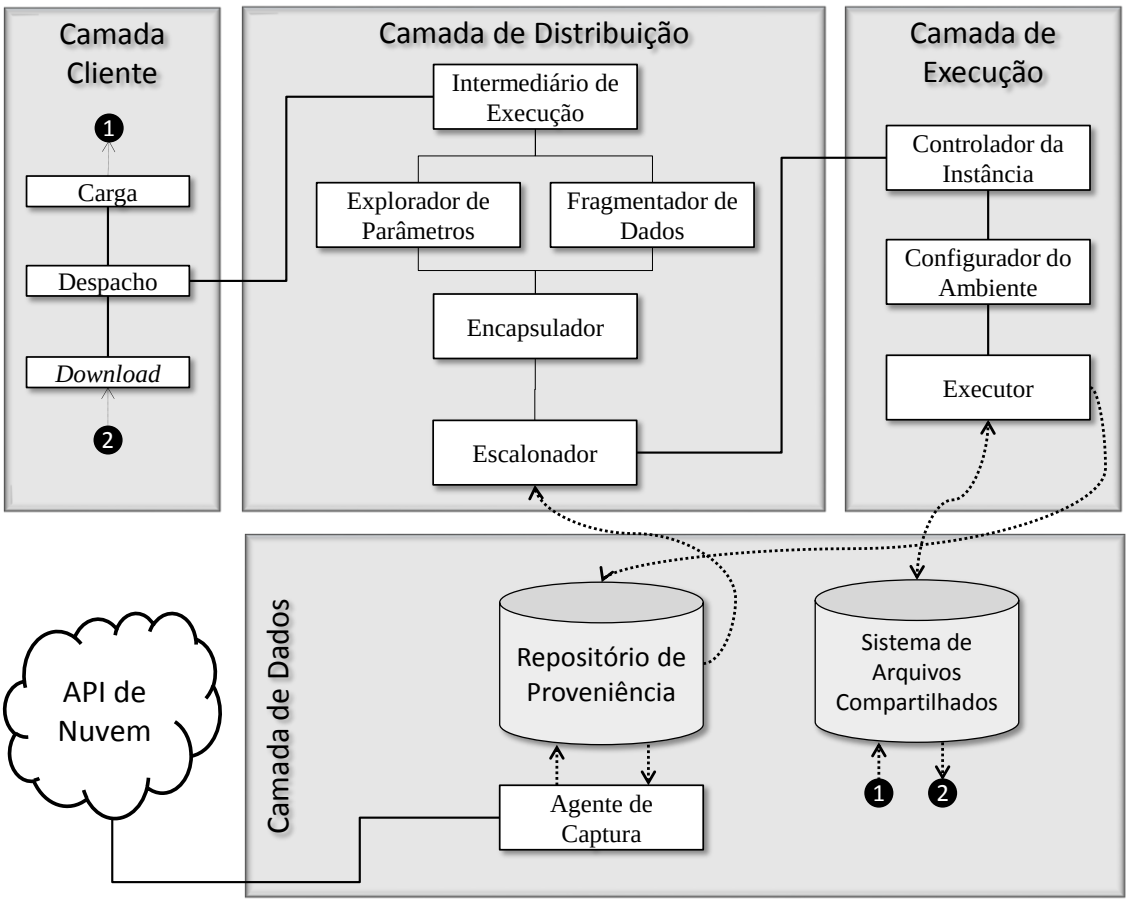

Figura 2 Arquitetura conceitual do SciCumulus

Desta forma, o SciCumulus fornece o apoio computacional para o paralelismo em *workflows* com coleta de proveniência distribuída, possibilitando consultas aos dados do experimento em tempo real. Entretanto, o SciCumulus, assim como as outras abordagens, não fornece mecanismos de monitoramento e notificação mais avançados que auxiliem o cientista a acompanhar o estado da execução sem que necessite estar fisicamente em algum terminal. Atualmente, o mecanismo de monitoramento no SciCumulus se dá por meio de consultas (realizadas pelo cientista) a base de proveniência. Mais informações sobre o SciCumulus são apresentadas por Oliveira *et al*. (2011).

2.5 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.

Balis *et al.* (2008) propõem uma taxonomia padrão de eventos de monitoramento de *workflows* em *Grids* de Computadores. Alguns tipos de eventos de monitoramento apresentados naquele artigo são equivalentes aos que foram definidos nesta dissertação, como evento de início de execução de atividade, falha de execução de uma unidade de execução e início de escrita em arquivo.

O Blackboard, proposto por Valerio *et al*(2008), é um sistema de monitoramento de execução de *workflows* científico baseado em ontologias para ser utilizado em conjunto com o SGWfC Trident. Neste sistema é aplicado um modelo de publicação/assinatura distribuído baseado em eventos. Diferentemente da abordagem apresenta nesta dissertação, não é objetivo do sistema Blackboard notificar o cientista diretamente, mas publicar as informações de monitoramento através de um serviço para que outros sistemas possam se utilizar daquelas informações. O grande foco deste trabalho relacionado é facilitar que informações de monitoramento provenientes de várias ferramentas sejam utilizadas por outras ferramentas, servindo assim como um barramento de serviços de monitoramento.

Em Valerio *et al*(2008), é realizada uma importante categorização dos tipos possíveis de monitoramento de *workflows* científicos. São definidas três categorias principais: (i) Monitoramento do estado da execução, no qual são rastreados os eventos relacionados à execução do *workflow* e de suas atividades; (ii) monitoramento da utilização de recursos, no qual é realizado o rastreamento do uso de recursos, como espaço em disco, utilização de *CPU* e utilização de memória ; (iii) monitoramento da evolução do *workflow*, onde alterações no desenho do *workflow* são rastreadas.

Outro trabalho relacionado é o MidMon (Cruz et al. 2008), que é uma ferramenta de monitoramento que gerencia o processo de execução de *workflows* científicos em clusters de computadores. Diferentemente do proposto nesta dissertação, o MidMon realiza o monitoramento da execução através do sistema operacional, o que faz com que ele não seja capaz de identificar eventos como tempos de execução atípicos. Já a solução apresentada nesta dissertação não tem como objetivo monitorar aspectos referentes aos processos e à infraestrutura onde estes são executados, mas

monitorar a execução do *workflow* científico através de seus resultados, notificando o cientista sobre eventos importantes através de dispositivos móveis e redes sociais.

Alguns trabalhos como RAVE (Grimstead, Avis, e Walker 2009) permitem que o cientista visualize o resultado das execuções de simulações científicas através de dispositivos móveis. Porém, ao contrário da solução desenvolvida nesta dissertação, este trabalho tem como objetivo monitorar a execução do experimento científico estruturados como *workflows* científicos, pois são aplicações para visualizar o resultado de simulações científicas.

O arcabouço proposto por Savarimuthu *et. al*(2004) se aproxima da nossa abordagem no momento em que utiliza critérios definidos pelo usuário para encontrar e notificar anomalias na execução do *workflow*. A grande diferença é que o arcabouço apresentado neste trabalho relacionado não é voltado para monitoramento de *workflows* científicos, mas para monitoramento de *workflows* de negócios. Outra diferença é que a notificação é realizada pela própria aplicação Desktop que realiza o monitoramento, não aplicando nenhuma tecnologia de dispositivos móveis, nem de redes sociais para realizar a notificação.

Através desta apresentação dos principais trabalhos relacionados é possível notar que existem diversas ferramentas que se aproximam da abordagem apresentada nesta dissertação em alguns aspectos, como o monitoramento de *workflows* em ambientes distribuídos (Cruz et al. 2008), a definição de eventos de monitoramento (Valerio et al. 2008), a utilização de dispositivos móveis (Grimstead, Avis, e Walker 2009) e utilização de critérios definidos pelos usuários para encontrar anomalias (Savarimuthu, Purvis, e Fleurke 2004). Entretanto não foi encontrado nenhum trabalho com arquitetura e funcionamento semelhantes às abordadas nesta dissertação.

Capítulo 3 - Acompanhamento de Execução de *Workflow* Científico em Ambientes Distribuídos

Workflows em larga escala tendem a executar por longos períodos de tempo em ambientes distribuídos, onde falhas de máquinas virtuais já não são uma exceção, mas sim uma característica. Por tais motivos, se torna ainda mais importante a existência de mecanismos de monitoramento. Mas, como dito anteriormente, as abordagens atuais oferecem apenas mecanismos básicos para realizar o acompanhamento da execução, como consultas diretamente na base de proveniência. Neste capítulo são detalhados todos os passos da concepção e implementação de uma nova abordagem de monitoramento de execução de *workflows* em larga escala. Esta nova abordagem é aplicada por meio da implementação do SciLightning, um arcabouço de monitoramento em tempo real de execução de *workflows* científicos que pode ser acoplado às soluções para execução de *workflows* existentes, como o SciCumulus.

A seção 3.1 tem por objetivo principal apresentar o levantamento que foi realizado para definir quais tipos de notificações seriam mais importantes para o cientista, detalhando quais parâmetros e formatos de mensagem foram escolhidos para cada tipo de notificação. Este levantamento foi utilizado como base para os próximos passos de construção da solução. A seção 3.2 apresenta a arquitetura conceitual do SciLightning, detalhando o funcionamento e a arquitetura interna de cada componente. Na seção 3.3, é realizado um detalhamento do modelo conceitual do repositório de proveniência do SciLightning e do esquema lógico desse mesmo repositório acoplado ao do SciCumulus. Por fim, a seção 3.4 apresenta os principais aspectos do processo de implementação de cada componente do SciLightning.

3.1 Tipos de Notificações Baseadas em Proveniência Gerada em Tempo Real

Esta dissertação apresenta um arcabouço que permite o monitoramento da execução de *Workflow* científicos por meio do envio de notificações para o cientista. Por este motivo, antes de iniciar o desenvolvimento da solução foi necessário realizar um levantamento de quais tipos de notificações seriam mais importantes para o cientista. Os tipos de notificação apresentados serviram de requisitos no desenvolvimento do

arcabouço, ou seja, o projeto e a construção foram executados para apoiar inicialmente esses tipos de notificações.

Com apoio de especialistas em bioinformática, foram identificados cinco tipos de notificações necessárias para os cientistas:

(i) **Erro na execução de uma atividade** - O cientista é notificado a cada erro de execução de atividades do *workflow* que ele deseja monitorar;

(ii) **Término de execução de *workflow*** - O cientista é notificado ao final da execução do *workflow* científico;

(iii) **Marco atingido** - Quando cada marco definido for atingido, é enviada uma notificação para o cientista especificando o marco e o em que horário que este foi atingido. Um marco pode ser de dois tipos: de início de atividade e de término de atividade;

(iv) **Arquivo gerado** - Podem ser definidas uma ou mais expressões regulares para monitoramento dos arquivos gerados. Para cada arquivo gerado pelo *workflow* científico que atenda a pelo menos uma das expressões regulares, é enviada uma notificação;

(v) **Execução de atividade com duração atípica** - Consulta realizada para identificar execuções de atividades com durações acima do tempo esperado.

O mesmo grupo de cientistas definiu um conjunto de informações, formato de mensagem e parâmetros de monitoramento diferentes para cada tipo de notificação. A única informação que é utilizada por todos os tipos de notificações é o nome/código do *workflow* que gerou a notificação. O restante das informações varia de um tipo de notificação para outro. Dependendo do tipo de notificação também é necessária a definição de parâmetros de monitoramento que vão identificar em quais situações serão geradas notificações. Esses parâmetros devem ser definidos para cada *workflow* que se deseja monitorar. A seguir são apresentados os formatos das mensagens e parâmetros de cada tipo de notificação.

O monitoramento para notificar erros na execução de uma atividade não possui parâmetros de notificação, ou seja, sempre que houver um erro na execução de uma atividade será enviada uma notificação possuindo o seguinte formato de mensagem: "*Error* - <descricão do erro>. *Activity* <nome da atividade>. *Task* <número da tarefa>. *Duration* <duração em segundos> *seconds*.".

O monitoramento de notificação de término de execução de *workflow* também não possui parâmetros de notificação, ou seja, sempre que o *workflow* que está sendo monitorado terminar a execução de todas as suas atividades será enviada uma notificação possuindo o seguinte formato de mensagem: "*The workflow has finished*. *Duration* <duração em segundos> *seconds*.".

O monitoramento para notificações de marco é obrigatoriamente parametrizado, pois é necessário definir quais atividades servirão como marcos. Além disso, cada marco pode ser definido como de início ou de término de atividade. No primeiro caso, a notificação será enviada assim que a atividade definida como marco começar a ser executada, já no segundo caso, a notificação só ocorrerá no final da execução da atividade. O cientista pode definir quanto marcos ele achar necessário e não é obrigatória a definição de nenhum marco. Quando cada marco definido for atingido, é enviada uma notificação para o cientista especificando o marco e o horário em que este foi atingido com o seguinte formato: "*The Activity* <nome da atividade> *has* [*started*|*finished*] *in* <data e horário que o marco foi atingido>".

O monitoramento de arquivos gerados é feito por meio da definição de um conjunto de uma ou mais expressões regulares que são comparadas com o nome de cada arquivo gerado. Uma expressão regular é uma cadeia de caracteres que é uma definição abreviada de um conjunto de cadeia de caracteres (um conjunto regular). Uma cadeia de caracteres é dita correspondente a uma expressão regular, se for um membro do conjunto regular descrito pela expressão regular (PostgreSQL 2012). Para cada arquivo gerado pelo *workflow* científico que o nome corresponda a pelo menos uma das expressões regulares, deve ser enviada uma notificação com o seguinte formato:" *The file* <nome do arquivo> *was produced by activity* <nome da atividade>". Foi adotada a sintaxe POSIX de expressões regulares básicas definida no documento ISO/IEC/IEEE 9945:2009 (ISO/IEC/IEEE 2009) por ser utilizado como padrão na maioria dos *softwares* que usam expressões regulares. Para facilitar a utilização de expressões regulares, foi definido que todas as comparações seriam realizadas de maneira insensível a maiúsculas e minúsculas (*Case Insensitive*). Por exemplo, expressão regular "aB" selecionará qualquer arquivo que possua pelo menos uma das seguintes cadeias de caracteres : "ab", "aB", "Ab" e "AB".

O padrão de expressões regulares escolhido possui uma série de caracteres especiais ou metacaracteres que são usados para indicar ações ou delimitar grupos, mas

é possível forçar esses caracteres especiais para serem interpretados como caracteres normais precedendo-os com um caractere de escape definido, a barra invertida "\" (ISO/IEC/IEEE 2009). Por exemplo, o ponto (“.”) é utilizado como um metacaractere coringa (*wild card*) para denotar qualquer valor, mas se precedido por uma barra invertida representa o caractere de ponto em si. A expressão “a.a” corresponde a várias cadeias, como, por exemplo, “aaa”, “aba”, “aza”, “a.a”, mas a cadeia “a\.a” corresponde apenas a "a.a". A barra invertida também escapa ela mesmo, ou seja, duas barras invertidas são interpretadas como um caractere de barra invertida literal (ISO/IEC/IEEE 2009).

As combinações de caracteres especiais são infinitas, mas existem algumas combinações que são mais interessantes para monitorar os nomes de arquivos gerados em uma execução de *workflow*. Por exemplo, se o cientista deseja ser notificado quando algum arquivo com extensão *.mafft* for gerado, ele dever incluir a seguinte expressão regular como parâmetro de monitoramento : “\.mafft$”. O símbolo “$” significa uma âncora de fim de cadeia de caracteres e foi utilizado para garantir que a cadeia de caracteres “.mafft” esteja exatamente no final do arquivo. Caso fosse definida a expressão regular “.mafft”, o arquivo com o nome “out.mafft.xml” também geraria notificação.

Muitas vezes o cientista está mais interessado em monitorar o início do nome do arquivo. De maneira semelhante, o símbolo “^”, que representa uma âncora de início de arquivo, poderá ser utilizado. Por exemplo, se o cientista deseja receber notificações sempre que um arquivo cujo nome se inicie com a cadeia de caracteres “experiment” for gerado, ele deve incluir a expressão regular “^experiment” como parâmetro por definição.

O monitoramento para identificar execuções de atividades com durações atípicas é o mais complexo, pois se baseia em dados de execuções anteriores para descobrir qual é o padrão de execução de cada tipo de atividade. As durações são consideradas populações estatísticas diferentes de acordo com o tipo de atividade, de modo que para a avaliação de cada atividade só seja considerado o histórico de execução das atividades do mesmo tipo. A medida de dispersão escolhida para esta dissertação foi o desvio padrão, pois se trata da medida de dispersão mais utilizada na inferência estatística (Braga 2010). O desvio padrão pode ser visto como o tamanho de um desvio típico ou representativo em relação à média da amostra e é calculada a partir da raiz quadrada da

média dos quadrados dos desvios das observações em relação a sua média amostral (Devore 2011). O desvio padrão será representado nesta dissertação pela letra grega sigma σ e quando se referir a uma população de um determinado tipo de atividade denominado X será representado por σ(X). De maneira semelhante, a média será representada por μ e μ(X).

Para verificar se estão ocorrendo atividades com durações anormais, é necessário realizar uma sequência de cálculos. Primeiramente, a média e desvio padrão das durações de cada tipo de atividade são calculados através do histórico de execuções (consultando o banco de proveniência). Então, para cada atividade em execução ou recém finalizada, é calculado o desvio de sua duração em relação a média de duração de execução das atividades do mesmo tipo. As atividades com o tempo de execução com desvio superior ou inferior à média com magnitude maior do que a amplitude limite são consideradas atividades com duração atípica (*outlier*). A amplitude limite pode ser ajustada pelo cientista para cada execução, sendo o valor padrão de 2,69 desvios-padrão em relação à média (Freedman, Pisani, e Purves 2007).

Em estatística, um *outlier*, ou valor atípico, é uma observação de que é numericamente distante do resto dos dados (Barnett e Lewis 1994). Segundo Grubbs (1969), uma observação periférica, ou *outlier*, é aquela que parece desviar-se acentuadamente a partir de outros membros da amostra em que ocorre. Para melhor entendimento, é apresentado um cenário fictício com atividades em três situações diferentes: (i) uma com a duração que se desvia inferiormente da média estando fora da amplitude limite; (ii) uma com a duração que se desvia de maneira superior à média, e também fora da amplitude limite e (iii) uma dentro da amplitude limite. As duas primeiras são consideradas durações com tempo atípico (*outliers*) por estarem abaixo ou acima da amplitude limite, já a terceira é considerada duração com tempo típico ou normal. No cenário apresentado na Tabela 1 abaixo, essas três atividades possuem Id 19, 20 e 21. Assume-se um cenário com uma amplitude limite de 3 desvios-padrão e o seguinte histórico de execuções:

Tabela 1 Histórico de execuções fictício

Id Atividade	*Tipo*	*Estado*	*Duração em segundos*
1	A	Finalizada	122
2	B	Finalizada	85

3	C	Finalizada	27
4	A	Finalizada	103
5	B	Finalizada	73
6	C	Finalizada	38
7	A	Finalizada	102
8	B	Finalizada	96
9	C	Finalizada	23
10	A	Finalizada	163
11	B	Finalizada	101
12	C	Finalizada	43
13	A	Finalizada	123
14	B	Finalizada	88
15	C	Finalizada	24
16	A	Finalizada	146
17	B	Finalizada	89
18	C	Finalizada	23
19	A	Finalizada	45
20	B	Em execução	143
21	C	Em execução	48

O primeiro passo é calcular a média e o desvio padrão das durações para cada tipo de atividades através do histórico de execuções, representadas por $\mu(A)$, $\sigma(A)$, $\mu(B)$, $\sigma(B)$, $\mu(C)$ e $\sigma(C)$.

Tabela 2 Médias e desvios-padrão do cenário

População	*Média(μ)*	*Desvio padrão(μ)*
Tipo A	126,5	24,07
Tipo B	88,67	9,65
Tipo C	29,67	8,66

Possuindo a média e o desvio padrão de cada tipo de atividade é possível calcular o desvio em relação à média medido em desvios-padrão das atividades, sendo possível verificar se cada atividade está dentro ou fora da amplitude limite. Sabendo que a amplitude limite foi definida como 3 desvios-padrão neste cenário, apenas as atividades de Id 19 e 20 são consideradas *outliers*, ou seja, apresentam duração anormal.

Tabela 3Análise de tarefas em execução ou recém finalizadas

Id Atividade	*Tipo*	*Desvio*		*Situação*
		Em segundos	*Em desvios-padrão*	
19	A	-81,5	-3,38	*Outlier* Inferior
20	B	54,33	5,63	*Outlier* Superior
21	C	18,33	2,11	Normal

É importante salientar que o conjunto de tipos de notificações que foi apresentado nesta seção é aquele que consideramos mais importante até o momento do desenvolvimento. Porém, de acordo com o domínio do experimento e conjunto de informações disponíveis na base de proveniência, outros tipos de notificações podem se tornar interessantes. Por tal motivo, tomamos como premissa de desenvolvimento do SciLightning tornar o mais simples possível a inclusão de novos tipos de notificações. Este esforço para facilitar esta extensão/adaptação pode ser notado em alguns componentes da arquitetura do SciLightning que é apresentada na próxima seção.

3.2 Arquitetura do SciLightning

A arquitetura do SciLightning (Figura 3) é dividida em seis componentes a serem acoplados às soluções de execução de *workflows*:

(i) **Cartucho**: componente que implementa um cartucho (Birsan 2005) para realizar a interface entre o SciLightning e a abordagem de execução de *workflows*;

(ii) **Base de Proveniência**: armazena informações sobre usuário, parâmetros de configuração do monitoramento e histórico de todas as notificações enviadas;

(iii) **Portal**: componente que permite que o cientista configure os parâmetros de monitoramento e consulte o histórico de notificações geradas.

(iv) **SciLightning Monitor**: componente de monitoramento e análise de proveniência;

(v) **SciLightning *Mobile***: componente para envio das notificações através de tecnologias móveis;

(vi) **SciLightning Social**: componente de publicação das notificações nas redes sociais.

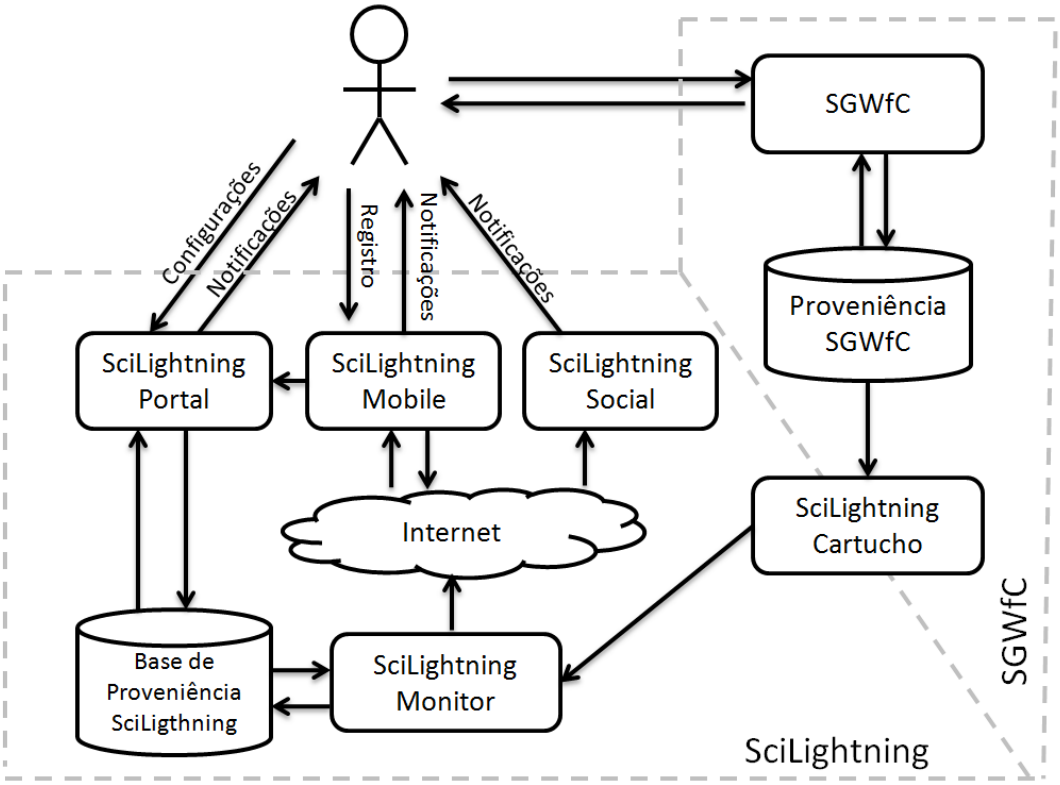

Figura 3 Arquitetura conceitual do SciLightning

Cada componente possui uma arquitetura interna e um modo de funcionamento próprio e é justamente através da interação entres esses componentes que o funcionamento do arcabouço SciLightning ocorre. As seções seguintes apresentam os aspectos arquiteturais e funcionais de cada um destes componentes.

3.2.1 Arquitetura do Cartucho

O SciLightning é projetado para se conectar a qualquer abordagem de execução de *workflows* que possua uma base de proveniência que possa ser consultada em tempo real. Esta base de proveniência pode estar em qualquer formato, por exemplo, tabelas de um banco de dados relacional ou arquivos de *logs*. Por este motivo, o SciLightning necessita conhecer o esquema/formato de cada base de proveniência para extrair os dados necessários, como arquivos produzidos, tempos de execução, *etc*. Desta forma, para cada base de proveniência o SciLightning implementa um cartucho diferente.

O modo de implementação do cartucho é livre, a única restrição é que deve ser implementada uma visão (*database view*) com uma assinatura predefinida que retorne as notificações que devem ser enviadas. O cartucho é o único ponto de contato entre o SciLightning e a abordagem de execução e foi planejado para oferecer máxima independência dos outros componentes do SciLightning em relação à abordagem de execução.

O funcionamento do cartucho é semelhante ao do padrão de projeto *Adapter* (Gamma et al. 1994), que converte a interface de uma classe em outra interface esperada pelos clientes, permitindo que classes com interfaces incompatíveis possam trabalhar juntas. O cartucho foi inicialmente planejado para servir apenas como um adaptador, convertendo a interface das abordagens de execução para uma interface padrão esperada por todos outros componentes do SciLightning. Porém, durante a especificação do SciLightning notou-se que os tipos de notificações e as regras em suas consultas poderiam variar bastante de acordo com as informações que sejam disponibilizadas por cada abordagem de execução. Por exemplo, pode existir uma base de proveniência onde não são armazenadas informações de geração de arquivos, porém são armazenadas informações sobre o uso da infraestrutura. Nesse caso, não seria possível monitorar a geração de arquivos diretamente através da base de proveniência, mas talvez se tornasse interessante criar um tipo de notificação relacionado ao uso da infraestrutura. Então, com o objetivo de proporcionar maior flexibilidade na adaptação do SciLightning, foi decidido manter todas as regras de notificações diretamente no cartucho. Fica sob responsabilidade de quem está implementando o cartucho a definição do conjunto de tipos de notificações que se deseja monitorar.

Como todas as regras de notificação se encontram implementadas no cartucho, qualquer alteração nestas regras pode ser realizada sem necessidade de adaptação dos demais componentes do SciLightning. Tal alteração pode ser necessária para, por exemplo, absorver mudanças no esquema da base de proveniência, alterar a lógica ou o conteúdo das notificações já definidas e até criar novos tipos de notificações.

Apesar de que a ideia inicial do SciLightning fosse efetuar o monitoramento através de consultas à base de proveniência da solução de execução, a arquitetura foi definida de forma que seja possível adotar outros métodos de monitoramento, como o monitoramento diretamente à infraestrutura ou aos processos em execução. Qualquer elemento de tecnologia da informação pode ser monitorado, desde que o mecanismo

seja devidamente implementado no cartucho. Vale ressaltar que mesmo que outros elementos de tecnologia da informação comecem a ser monitorados, a princípio não haverá necessidade de alterar os demais módulos do SciLightning.

3.2.2 Base de Proveniência do SciLightning

O SciLightning possui uma base de proveniência própria que armazena todas as informações relacionadas ao monitoramento, como informações sobre usuários do SciLightning, parâmetros de configuração de monitoramento e um histórico de todas as notificações enviadas. Foi decidido manter esta base de proveniência separada para que não houvesse a necessidade de adaptar/estender a base de proveniência monitorada para suportar o SciLightning e para que todos módulos do SciLightning possam acessar informações de proveniência diretamente, sem a necessidade da intermediação do cartucho. Se as informações de monitoramento fossem armazenadas na base de proveniência monitorada, não poderia haver acesso direto, pois, como definido anteriormente, o SciLightning acessa a base de proveniência monitorada exclusivamente através do cartucho.

A base de proveniência SciLightning armazena informações sobre cada usuário utilizadas nos processos de *login* (nome de usuário e senha) e de envio de mensagens (nomes de usuários das redes sociais e códigos de identificação dos seus dispositivos móveis). Além disso, são armazenadas informações que permitem a definição de grupos de usuários, que são utilizados para enviar todas as notificações de um monitoramento para um conjunto definido de cientistas.

Os parâmetros de configuração do monitoramento, que também são armazenados na base de proveniência SciLightning, incluem a lista de *workflows* que devem ser monitorados e parâmetros de monitoramento para cada *workflow*, de acordo com os tipos de notificação.

Nesta seção foram apresentados os requisitos básicos desta base de proveniência. O modelo de proveniência desenvolvido neste trabalho de dissertação é apresentado na seção 3.3 de Modelo de Proveniência.

3.2.3 Arquitetura do SciLightning Portal

A relação de quais *workflows* serão monitorados, assim como a definição de parâmetros de monitoramento e destinatários das notificações para cada *workflow* são definidos

através de um portal *Web* e armazenados na base de proveniência do SciLightning. Este portal também permite que o cientista consulte o histórico de todas as notificações que foram enviadas para ele e permite o registro de aparelhos móveis para receber notificações.

Para configurar um monitoramento, o cientista deve seguir os seguintes passos:

(i) Definir qual o *workflow* será monitorado identificando o nome/*tag* do *workflow*;

(ii) Definir os parâmetros de monitoramento de acordo com os tipos de notificações, conforme foi explicado na seção anterior; e

(iii) Definir para quais usuários e grupos serão enviadas as notificações deste monitoramento.

O SciLightning só monitora e, consequentemente, envia notificações para os *workflows* com o monitoramento previamente configurado no Portal. Esta configuração é estritamente necessária para cada monitoramento, pois define os destinatários das notificações. Como a identificação ocorre pelo nome/*tag* do *workflow*, todas as execuções do mesmo *workflow* podem ser acompanhadas com uma única configuração. Ou seja, todos os parâmetros de monitoramento e destinatários das mensagens serão reutilizados automaticamente para todas as execuções daquele mesmo *workflow*.

É possível atualizar a configuração de um monitoramento por meio do Portal. Quando isso ocorrer, todas as execuções daquele *workflow* passarão a utilizar essa nova configuração. Também é possível configurar mais de um monitoramento para um mesmo *workflow*, possuindo parâmetros de monitoramento distinto e, possivelmente, destinatários diferentes.

O componente SciLightning Portal possui a responsabilidade de manter um serviço que possa ser utilizado pelo SciLightning *Mobile* para registro dos dispositivos móveis para recebimento das notificações. Tal serviço é necessário, pois o SciLightning *Mobile*, que é implementado através de aplicações de dispositivos móveis, não consegue acessar diretamente a base de proveniência do SciLightning para efetuar a autenticação do cientista e armazenar o código de identificação do aparelho. O processo de registro de dispositivos móveis será detalhado na seção 3.4 de Detalhes de Implementação.

A manutenção do cadastro dos usuários SciLightning também é realizada através do Portal. Informações de nome, *login*, senha, *e-mail*, contas das redes sociais e

instituição do usuário podem ser definidas/atualizadas tanto na realização de um cadastro novo quanto na atualização de um já existente.

3.2.4 Arquitetura do SciLightning Monitor

O SciLightning Monitor é o componente que tem como principal funcionalidade realizar o monitoramento da execução de *workflows* por meio de consultas periódicas à base de proveniência da abordagem de execução (utilizando o cartucho). O monitoramento realizado é totalmente baseado nas configurações definidas por meio do Portal e armazenadas na base de proveniência SciLightning. A frequência de monitoramento, ou seja, o tempo entre as consultas, é configurável, possuindo valor padrão de 30 segundos. Uma vez redefinida, a nova frequência valerá para todos os *workflows* monitorados.

Além de realizar o monitoramento da base de proveniência, o SciLightning Monitor é responsável por enviar as notificações geradas para serem tratadas pelos componentes SciLightning *Mobile* e SciLightning Social. Para isso, o SciLightning Monitor utiliza os códigos de identificação dos dispositivo móveis e nomes de usuários de redes sociais associados a cada cientista para direcionar as notificações para seus devidos destinatários. O código de identificação é relacionado a apenas um dispositivo móvel e cada dispositivo móvel é, por sua vez, relacionado a apenas um único cientista. Como um único cientista pode possuir mais de um dispositivo móvel, o SciLightning permite o registro de um número ilimitado de dispositivos para cada cientista. Para cada notificação destinada a um cientista, o SciLightning Monitor envia uma cópia da notificação para cada dispositivo móvel associado a esse cientista. O SciLightning Monitor apenas envia a mensagem de notificação. O recebimento e tratamento da notificação pelo dispositivo móvel é realizado através do componente SciLightning *Mobile*. De maneira semelhante, o SciLightning Monitor envia uma cópia da notificação para cada conta de rede social associada ao cientista.

Devido à possível ocorrência de falhas que são inerentes aos ambientes distribuídos, principalmente quando se considera a utilização de nuvens públicas, é necessário que sejam utilizados mecanismos de tolerância a falhas. Caso haja algum problema que impeça o funcionamento correto do SciLightning durante um período, todas as notificações retroativas devem ser automaticamente identificadas e enviadas assim que o funcionamento for restabelecido.

3.2.5 Arquitetura do SciLightning *Mobile*

O SciLightning *Mobile* é um componente de tecnologia móvel que tem como principal objetivo receber, armazenar e exibir as notificações geradas pelo SciLightning Monitor. Existem vários sistemas operacionais e, consequentemente, várias plataformas de desenvolvimento para dispositivos móveis. A arquitetura do SciLightning *Mobile* foi planejada para não ser dependente ou limitada a nenhuma dessas plataformas. Ao invés disso, os conceitos apresentados nesta seção poderão ser utilizados na implementação de aplicativos de qualquer plataforma. A ideia é que aplicações que implementem o SciLightning *Mobile* em plataformas diferentes possuam arquitetura e funcionamento semelhantes.

O SciLightning *Mobile* foi desenhado para possuir 3 funcionalidades principais:

(i) Registro – utilizada para o registro da aplicação junto ao SciLightning Portal.

(ii) Resumo – apresenta um resumo de notificações recebidas agrupadas por *workflow.*

(iii) Mensagens de *Workflow* – apresenta uma lista detalhada de todas as mensagens recebidas de um determinado *workflow*.

O diagrama a seguir (Figura 4), representado por meio da notação do diagrama de maquina de estados da UML (Bezerra 2007), apresenta o esquema de navegação do SciLightning *Mobile*. A aplicação deverá ser registrada em nome de um cientista (junto ao SciLightning Portal) para que esteja habilitada a receber notificações. Sempre que a aplicação for iniciada, deve ser feita uma verificação se o aparelho está devidamente registrado. Caso não esteja, a aplicação deve ser automaticamente redirecionada para a tela de registro, onde serão solicitadas as credenciais do cientista (*login* e senha). As credenciais do cientista devem ser enviadas para o SciLightning Portal juntamente com o código de identificação do dispositivo móvel. No caso da utilização de serviços de entrega de mensagens para dispositivos móveis, o código de identificação do dispositivo móvel é obtido junto ao provedor do serviço.

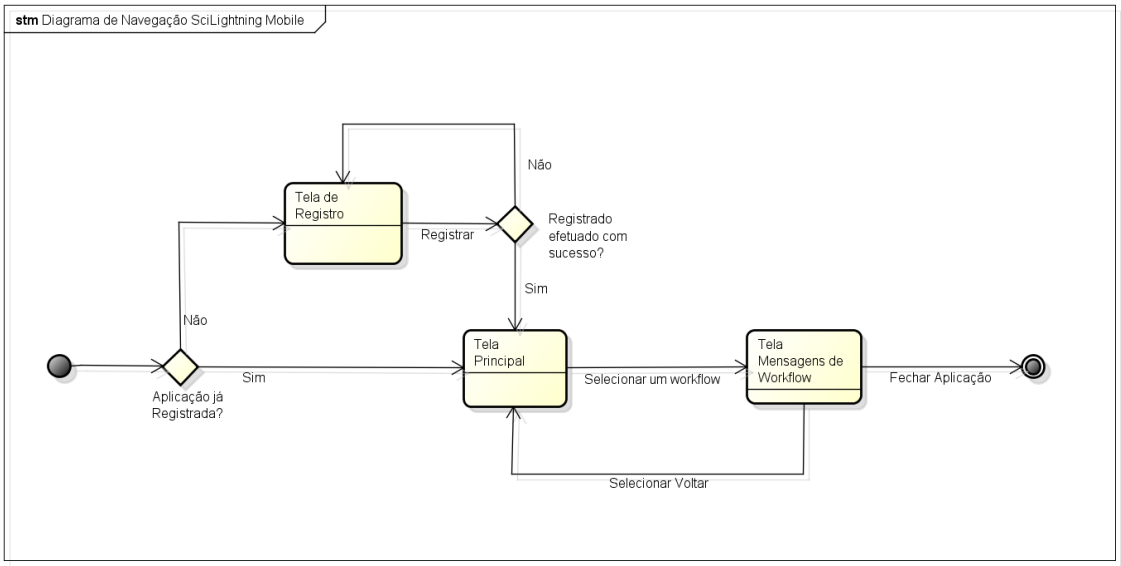

Figura 4 Mapa de navegação SciLightning *Mobile*

3.2.6 Arquitetura do SciLightning Social

O SciLightning Social é um componente que faz a interface entre o SciLightning e as redes sociais. O SciLightning Social utiliza a API das redes sociais para publicar as notificações do monitoramento. De maneira semelhante ao SciLightning *Mobile*, o conceito do componente SciLightning Social não é dependente especificamente de nenhuma rede social. Ao invés disso, a ideia é que esse componente possa ser implementado em qualquer ferramenta de rede social que disponibilize uma API de desenvolvimento.

As implementações do SciLightning Social devem permitir o envio de notificações tanto por meio de um mecanismo de mensagens privadas quanto através de um mecanismo de mensagens públicas. Durante a configuração de um monitoramento de um *workflow* no SciLightning Portal, o cientista define qual é o modo desejado de recebimento das mensagens através de redes sociais. O modo privado é interessante quando o cientista deseja ter sigilo das informações contidas nas notificações. Já o modo público é interessante quando se deseja utilizar as funcionalidades das redes sociais para promover e compartilhar as notificações com outros usuários.

O SciLightning Social se divide em dois subcomponentes: (i) um componente que encapsula a API de desenvolvimento que se acopla ao SciLightning Monitor (ii) uma conta na rede social que possua o devido perfil de acesso para envio de mensagens através da API de desenvolvimento. O principal objetivo do primeiro subcomponente é tornar transparente para o Monitor os detalhes de implementação de cada API de

desenvolvimento, fornecendo uma interface padrão de envio de mensagens para qualquer que seja a ferramenta de rede social. Já a conta tem o objetivo de fornecer uma entidade dentro da rede social que vai ser utilizada para enviar as notificações de fato para o usuário do cientista. O componente que encapsula a API de desenvolvimento deve possuir o devido acesso à conta que envia as mensagens para o cientista.

3.3 Modelo de Proveniência do SciLightning

Além de realizar o monitoramento e as notificações, o SciLightning também armazena os dados de proveniência relativos a esse monitoramento. Desse modo, o SciLightning possui um modelo de proveniência próprio que deve ser acoplado ao modelo de dados da abordagem de execução de *workflows*. O modelo do SciLightning foi projetado para se acoplar idealmente a modelos de proveniência de SGWfCs que sejam baseados no *Open Provenance Model* (OPM) (Moreau, Freire, et al. 2008; Moreau, Ludäscher, et al. 2008). O OPM é uma recomendação aberta que tem como objetivo a interoperabilidade entre os dados de proveniência dos diversos SGWfC. O OPM representa as relações causais entre processos, agentes, artefatos e papéis normalmente envolvidos em uma execução de *workflow* científico.

A Figura 5 apresenta o modelo conceitual do repositório de proveniência do SciLightning, enquanto que a Figura 6 demonstra o esquema lógico da base de proveniência criado a partir do modelo conceitual e acoplado ao esquema lógico do SciCumulus. O modelo conceitual da base de proveniência, além de apresentar as entidades que são do próprio SciLightning, apresenta também as entidades da base de proveniência do SGWfC que também são utilizadas (diferenciadas no modelo por possuírem o nome sublinhado). Essas entidades são as principais do modelo de proveniência do SGWfC para efeito de monitoramento e têm relação direta com algumas entidades do modelo do SciLightning. A entidade *Workflow* representa cada execução de *workflow* no SGWfC. A entidade *Atividade* representa uma atividade de uma determinada execução de *workflow* e a entidade Tarefa representa uma tarefa de uma determinada atividade. Já a entidade *Arquivo* representa um arquivo gerado por uma atividade durante a execução do *workflow*.

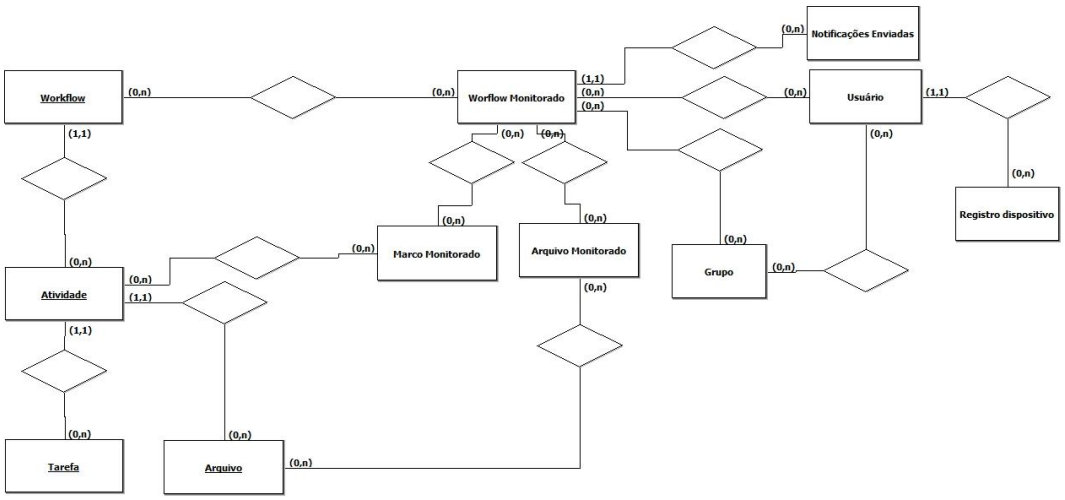

Figura 5 Modelo conceitual do repositório de proveniência do SciLightning

A entidade *Workflow Monitorado* representa um monitoramento de execução de um *workflow* científico. Essa é a entidade central do modelo de proveniência do SciLightning. Todas as outras entidades se relacionam direta ou indiretamente com a entidade *Workflow Monitorado* e servem para representar as configurações, destinatários e históricos de um determinado monitoramento. A entidade *Marco Monitorado* representa uma configuração de monitoramento de marco atingido e a entidade *Arquivo Monitorado* representa uma configuração de um monitoramento de geração de arquivo. Tanto o monitoramento de marco atingido quanto o monitoramento de geração de arquivo são configurados através do SciLightning Portal e estão relacionados a tipos específicos de notificação, conforme explicado na Seção 3.1.

As entidades *Usuário*, *Grupo de Usuários* e *Dispositivo Registrado* são utilizadas para realizar o controle de acesso ao SciLightning Portal e fazer o envio de notificações tanto através do SciLightning Social e *Mobile*. O monitoramento de um *workflow*, representado pela entidade *Workflow Monitorado,* pode possuir tanto usuários quanto grupos de usuários como destinatários. Todas as notificações geradas a partir de um determinado monitoramento serão enviadas para todos os usuários e grupos associados a esse monitoramento. Quando uma notificação tem como destinatário um grupo, representado pela entidade *Grupo de Usuários*, todos os usuários que sejam componentes deste grupo a recebem.

A entidade *Dispositivo Registrado* representa um dispositivo móvel utilizando o SciLightning *Mobile* registrado para receber notificações em nome de um determinado cientista. Uma instância da entidade *Usuário* pode estar relacionada com mais de uma instância da entidade *Dispositivo Registrado*, pois um único usuário pode ter vários dispositivos registrados em seu nome. Nesse caso, todas as mensagens que sejam destinadas ao usuário são enviadas para todos seus dispositivos móveis. Informações de conta de redes sociais para envio de notificações através do SciLightning Social e as informações para efetuar o controle de acesso ao Portal são armazenadas diretamente na entidade *Usuário*.

Por último, a entidade *Notificação Enviada* é utilizada para armazenar todo o histórico de envio de notificações pelo SciLightning. Este histórico pode ser visualizado através do SciLightning Portal e permite uma análise da eficácia do SciLightning. Durante a avaliação experimental, que será apresentada no Capítulo 4, o histórico de notificações foi de fundamental importância para realizar todas as análises do SciLightning de eficácia da ferramenta.

O modelo apresentado na Figura 6 traz o modelo de dados do SciLightning acoplado ao modelo de dados do SciCumulus. O modelo de dados do SciCumulus foi estendido para permitir a definição de parâmetros de monitoramento, o registro da aplicação SciLightning *Mobile* e o controle de usuários e grupos. Nenhuma das entidades utilizadas no SciCumulus sofreram alterações em sua estrutura ou restrições. Portanto, as alterações efetuadas para utilizar o SciLightning não impactam a base do SciCumulus. As entidades criadas se integram ao modelo do SciCumulus através de chaves estrangeiras e são apresentadas em destaque na cor cinza no diagrama abaixo (Figura 6). As classes do SciCumulus são apresentadas sem os atributos para facilitar a visualização das classes do SciLightning.

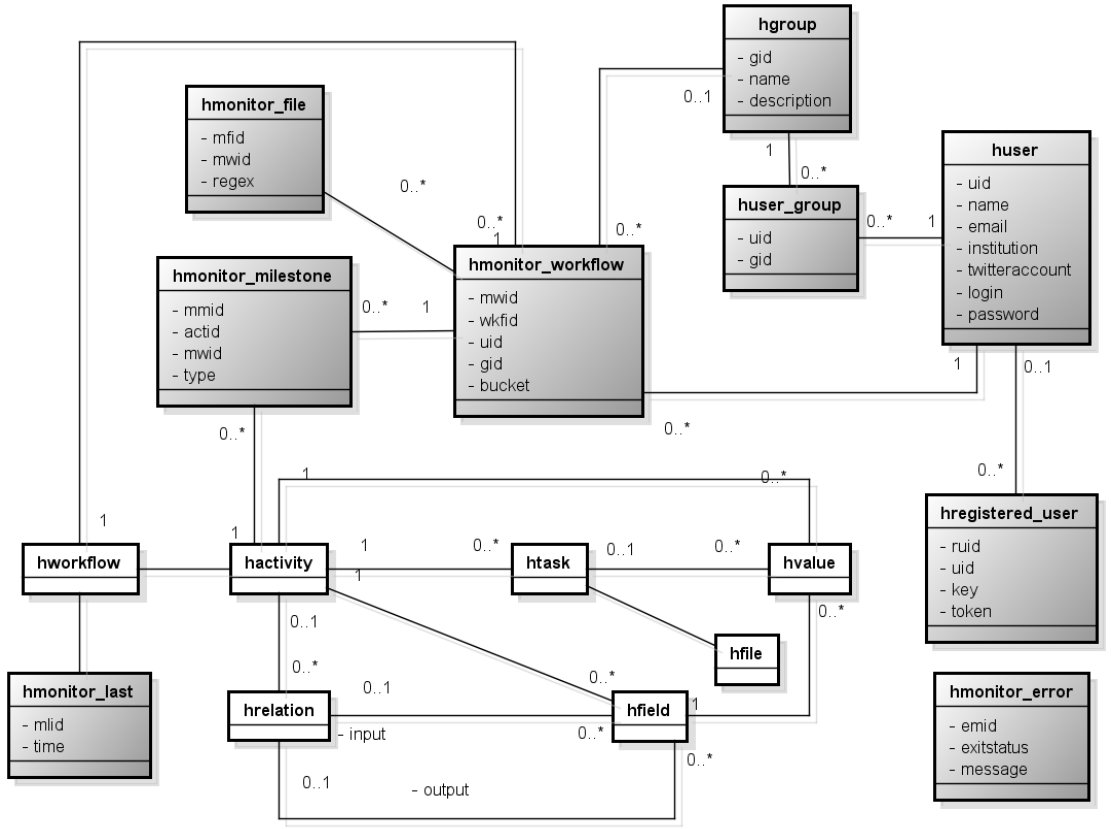

Figura 6 Modelo de Proveniência do SciLightning acoplado ao SciCumulus

A entidade *hmonitor_workflow* armazena os monitoramentos, ou seja, quais *workflows* são monitorados e quais usuários ou grupos serão notificados. Para cada monitoramento podem ser definidos parâmetros que configuram o envio de notificações quando o *workflow* gerar um determinado arquivo ou quando atingir um determinado marco conforme explicado na seção 3.1. Esses dois parâmetros de monitoramento são representados pelas classes *hmonitor_file* e *hmonitor_milestone*, respectivamente. Tanto a relação de *workflows* que devem ser monitorados, quanto os parâmetros de monitoramento são preenchidos através do portal *Web* antes do início da execução do *workflow*. Informações sobre usuário, como *login* e senha utilizados para o registro da aplicação SciLightning *Mobile* e a conta do *Twitter* utilizada para envio de notificações pelo SciLightning Social são armazenadas na tabela *huser*. Esses usuários podem ser organizados em grupos através das tabelas *hgroup* e *huser_group*, o que permite que as notificações sejam destinadas a todos os integrantes do grupo. Os *tokens* de registro da aplicação SciLightning *Mobile* não puderam ser armazenados na classe *huser*, uma vez que podem existir diversos dispositivos registrados para um mesmo usuário. Por este motivo, foi criada a tabela *hregistered_user*. A *hmonitor_last* é uma tabela auxiliar que armazena a data do último monitoramento realizado e tem como objetivo otimizar as

consultas de proveniência. Por último, a *hmonitor_error* é uma classe que armazena os códigos de erros e suas respectivas mensagens, que são utilizadas para tornar os textos de notificações de erros mais claros para o cientista.

3.4 Detalhes de Implementação

3.4.1 Implementação do Cartucho

Nesta dissertação foi implementado um cartucho para conectar o SciLightning à abordagem de execução de *workflows* SciCumulus. O cartucho foi implementado por meio de uma pilha de visões de banco de dados. Tal estrutura foi escolhida para implementação do cartucho devido à base de proveniência do SciCumulus já ser armazenada em um banco de dados relacional e o cartucho ter que implementar uma visão de banco de dados, conforme definido em 3.2.1Arquitetura do Cartucho.

A pilha implementada é composta por 11 visões, sendo a maioria delas agrupadas por tipo notificação. Para o monitoramento de erros na execução de uma atividade foi criada a visão *v_monitor_error*. Para monitorar o término de execução de *workflows* foi criada a visão *v_finished_workflows* e *v_monitor_finished*. Para monitorar o atingimento de marcos foi criada a visão *v_monitor_milestone*. Para monitorar a geração de arquivos foi criada a visão *v_monitor_file*. A tabela a seguir mostra as visões que compõem a pilha, identificando a dependência entre elas.

Tabela 4 Visões de implementação do Cartucho

Nº	*Visão*	*Tipo de Notificação*	*dependências*
1	v_monitor_last	Todos os tipos	
2	v_monitor_error	Erro em Tarefa	1
3	v_finished_*workflow*s	Término de Execução	
4	v_monitor_finished	Término de Execução	1
5	v_monitor_milestone	Atingimento de Marco	1
6	v_monitor_file	Geração de Arquivos	1
7	v_monitor_starttime	Tarefas Atípicas	
8	v_monitor_stddev	Tarefas Atípicas	7
9	v_monitor_unusual	Tarefas Atípicas	1,8
10	v_monitor_notification_pre	Todos os tipos	2,4,5,6,9
12	v_monitor_notification	Todos os tipos	10

A *v_monitor_last* é uma visão auxiliar que retorna a data/hora do último envio de notificação e serve para auxiliar outras visões em identificar quais notificações são novas. A visão *v_monitor_notification_pre* faz a união das visões de todos os tipos de notificação. A visão *v_monitor_notification*, que está no topo da pilha de visão, implementa a interface padrão do SciLightning Cartucho, ou seja, esta visão será o único ponto de contato entre o SciLightning Monitor e o SciLightning cartucho.

3.4.1 Implementação do SciLightning Portal

O SciLightning Portal foi implementado através de um sistema Web utilizando a linguagem de programação C# e utilizando o Framework .NET MVC 4[3]. Foi adotado o framework NHibernate[4] para realizar o mapeamento objeto relacional. Toda a interface foi construída com HTML, CSS e Javascript, JQuery[5] e Razor[6]. JQuery é uma biblioteca JavaScript compatível com vários navegadores (*cross-browser)* desenvolvida para simplificar os scripts do lado do cliente que interage com o HTML e o Razor é um motor de visualização da Microsoft que permite inserir códigos diretamente na camada de visualização da aplicação, facilitando a codificação do projeto.

O sistema divide as funcionalidades em três grupos. O primeiro grupo exibe o histórico de notificações que foram recebidas pelo usuário que esteja ativo na sessão. O segundo grupo contém as funcionalidades que permitem a criação, alteração e desativação de configurações de monitoramento. O último grupo está relacionado com a manutenção do cadastro dos cientistas que poderão receber notificações. A Figura 7 apresenta a tela da funcionalidade de inclusão de uma configuração de monitoramento. Esta funcionalidade foi dividida em 3 passos: Selecionar *Workflow*, Definir Parâmetros de Monitoramento e Definir os Destinatários das Notificações.

---

[3] http://www.asp.net/mvc/mvc4

[4] http://nhforge.org/

[5] http://jquery.com/

[6] http://msdn.microsoft.com/pt-br/library/gg675215.aspx

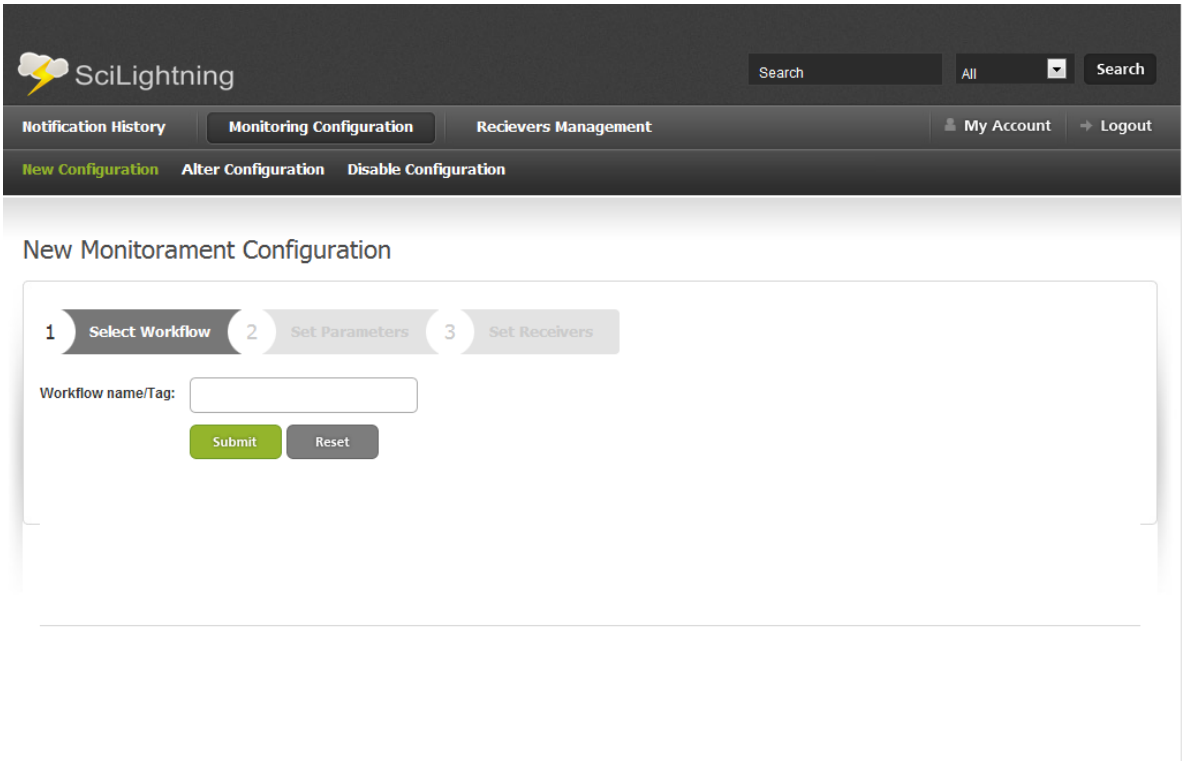

Figura 7 Tela de Inclusão de nova configuração de monitoramento – SciLightning Portal

3.4.2 Implementação do SciLightning Monitor

O SciLightning Monitor é um componente desenvolvido em Java versão 6.27 que tem como principal funcionalidade realizar o monitoramento da execução de *workflows* por meio de consultas periódicas à base de proveniência (utilizando o cartucho). O diagrama de sequência desta atividade de monitoramento é representado na Figura 8. A frequência de monitoramento, ou seja, o tempo entre as consultas é configurável, possuindo valor padrão de 30 segundos. Além desta funcionalidade principal, o SciLightning Monitor envia notificações para serem tratadas pelos componentes SciLightning Social e SciLightning *Mobile*.

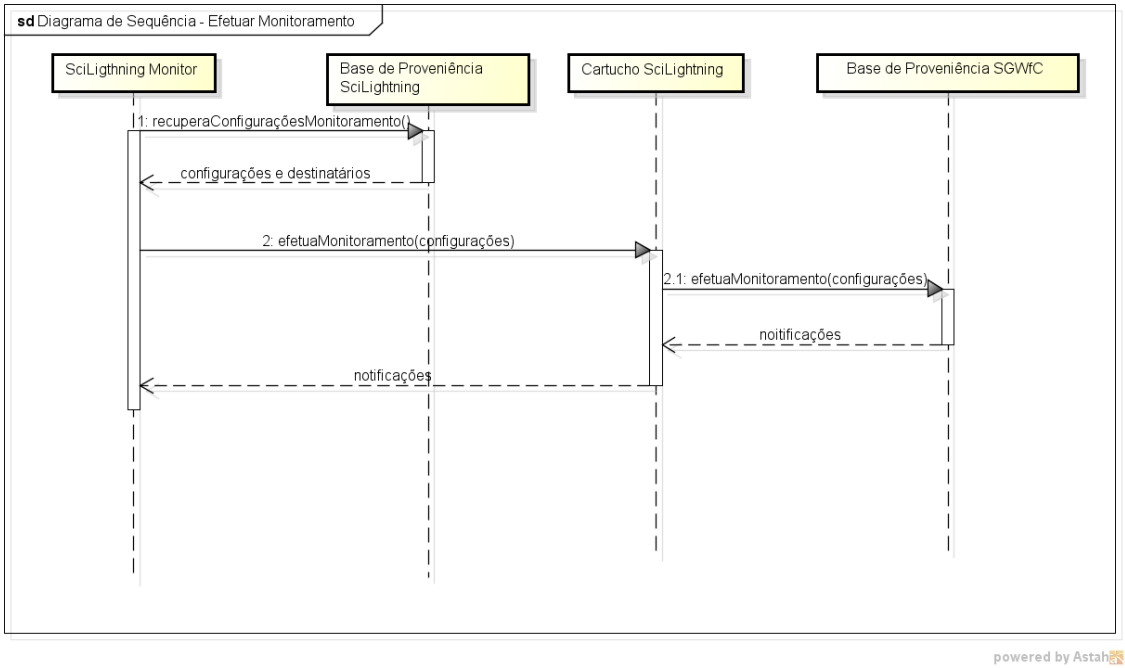

Figura 8 Diagrama de Sequência - Efetuar Monitoramento

O SciLightning Monitor possui o *token* de identificação de cada dispositivo móvel registrado. Cada *token* de identificação é relacionado a apenas um cientista e cada cientista pode possuir vários dispositivos registrados. Quando uma notificação tem um cientista como destinatário, esta notificação é enviada para todos dispositivos móveis registrados para ele. Ao receber uma notificação, é criada uma mensagem na barra de *status* do dispositivo. Caso o cientista necessite obter detalhes da notificação, ele pode selecionar a mensagem na barra de *status* para abrir a tela da aplicação SciLightning *Mobile* e listar todas as notificações recebidas.

3.4.3 Implementação do SciLightning *Mobile*

O SciLightning *Mobile* é um componente de tecnologia móvel que recebe, armazena e exibe as notificações geradas pelo SciLightning Monitor. Como dito anteriormente, o componente SciLightning *Mobile* pode ser implementado em diversas plataformas de dispositivos móveis, desde que atenda os aspectos arquiteturais e funcionais apresentados na seção 3.2.5 Arquitetura do SciLightning .

Nesta dissertação, o SciLightning *Mobile* foi implementado por meio de uma aplicação *Android*[7]. A plataforma *Android* foi escolhida, dentre as outras opções de tecnologias para dispositivos móveis, por ser uma solução de código aberto com ferramentas de desenvolvimento gratuitas e com grande quantidade de ferramentas desenvolvidas pela comunidade. Além disso, é utilizada em uma ampla gama de aparelhos (celulares, *tablets* e *netbooks*) de diversos fabricantes.

O serviço C2DM (*Cloud To Device Message),* que se trata de mecanismo para envio de mensagens em aplicações de dispositivos móveis, foi escolhido para enviar as notificações de monitoramento para a aplicação *Android* desenvolvida. O serviço C2DM trata todos os aspectos de enfileiramento de mensagens e entrega para o aplicativo em execução no dispositivo de destino. Este serviço fornece um mecanismo simples e leve que os servidores podem usar para avisar aplicativos móveis para contactar o servidor diretamente e buscar atualização da aplicação ou dados do usuário (Google Developers 2012).

Ao utilizar o serviço de C2DM, não é necessário que a aplicação *Android* fique sendo executada em background fazendo requisições periodicamente ao servidor para verificar a existência de novas mensagens resultando em economia considerável de banda de internet e energia do dispositivo móvel. As mensagens são recebidas pela aplicação mesmo que ela não esteja sendo executada no momento. Quando isso ocorre, o próprio Sistema Operacional *Android* se responsabiliza por iniciar a execução da aplicação e repassar a mensagem para ela.

Apesar das grandes vantagens apresentadas na utilização deste serviço, algumas limitações são impostas e foram levadas em consideração no momento da implementação do SciLightning *Mobile* . As principais limitações são o fato de que este serviço não garante a ordem da entrega de mensagens, o tamanho limite de uma mensagem é 1024 bytes e só funciona em dispositivos móveis baseados no SO *Android* 2.2 ou versão superior. O serviço C2DM também não foi projeto para enviar uma grande quantidade de conteúdo por meio das mensagens. No caso de haver a necessidade de enviar uma grande quantidade de dados, é melhor utilizar o serviço

[7] http://www.android.com/

apenas para dizer à aplicação móvel que existem novos dados no servidor e, então, a aplicação deve ir buscar os dados junto ao servidor.

Perto da data da conclusão da escrita desta dissertação, em 26 de julho de 2012, o Google anunciou que o serviço C2DM passou a não ser mais evoluído pelo mesmo, sendo substituido por uma nova versão chamada GCM (Google Cloud Message). Apesar de ser recomendada a migração para esta nova versão, o serviço do C2DM será mantido, não sendo permitido apenas a inclusão de novas aplicações. Como o SciLightning *Mobile* já se encontrava cadastrado, seu funcionamento não foi afetado por esta mudança.

O cientista necessita apenas de seu nome de usuário e senha para realizar o registro da aplicação. Porém, internamente, o registro ocorre em dois passos: (i) A aplicação *Android* solicita um *token* de identificação aos servidores do serviço C2DM (*Cloud To Device Message*, um mecanismo para envio de mensagens em aplicações de dispositivos móveis) e, (ii) se o *token* for recebido com sucesso, as informações de nome de usuário, senha e *token* de identificação são submetidas para o SciLightning *Portal*, que só então finalizará o registro após a validação das credenciais do cientista. O diagrama de sequência apresentado na Figura 9 demonstra como o registro ocorre internamente.

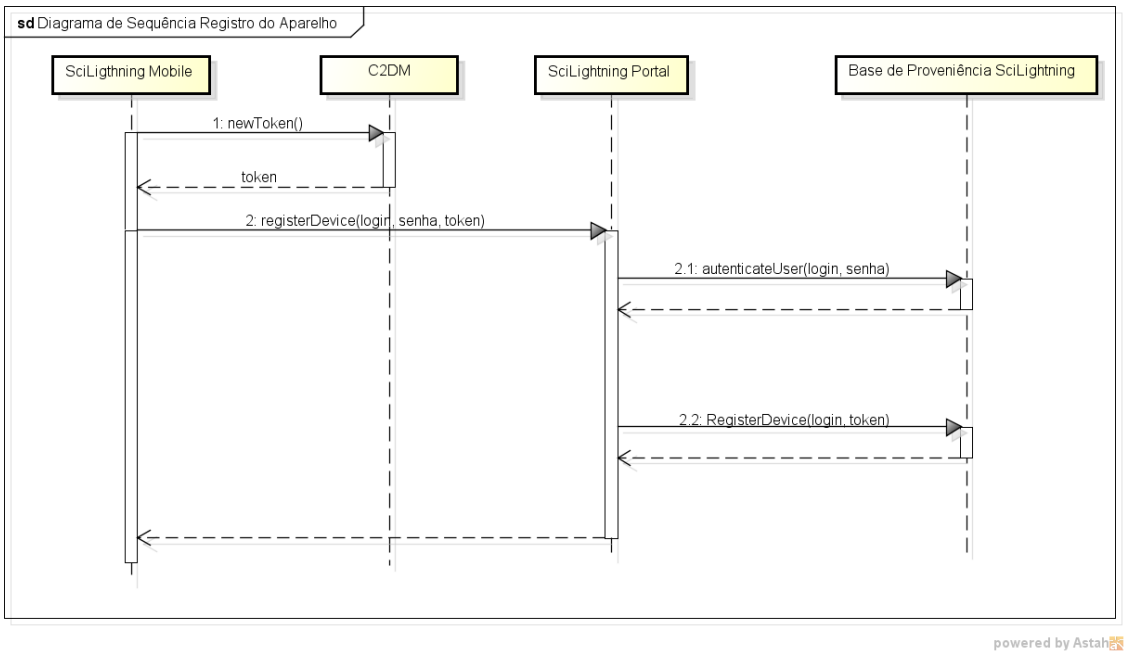

Figura 9 Diagrama de Sequência do Registro do dispositivo móvel

Após a fase de registro, o *token* referente àquela aplicação, naquele aparelho, estará registrado na base de proveniência do SciLightning associado ao registro do cientista. Este *token* será utilizado sempre que o SciLightning Monitor estiver enviando notificações para aquele cientista. O cientista pode registrar quantos aparelhos desejar(Figura 10). Neste caso, as notificações que têm este cientista como destinatário serão enviadas para todos os *tokens*.

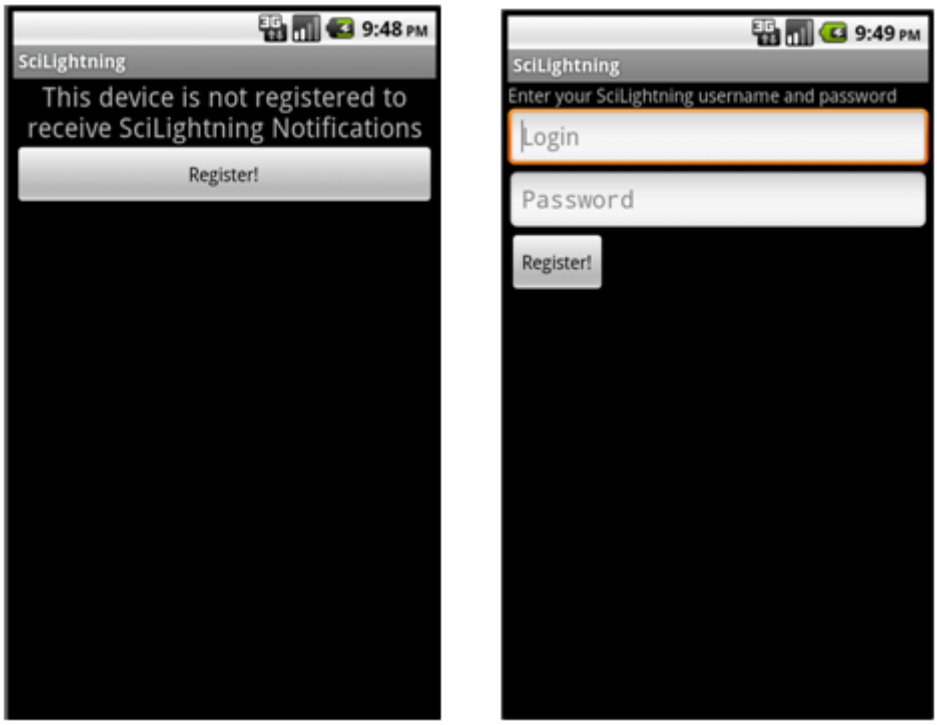

Figura 10 Telas de registro do dispositivo móvel do SciLightning *Mobile*

As mensagens recebidas pela aplicação móvel do SciLightning são armazenadas localmente através do SQLite[8], uma biblioteca em linguagem C que implementa um banco de dados SQL embutido. Esta ferramenta possui total suporte da plataforma *Android*, o que permite que aplicações possam ter acesso a um banco de dados relacional sem ter executar um processo SGBD separado.

Como o SciLightning pode gerar uma grande quantidade de notificações para um mesmo cientista, as mensagens não são enviadas diretamente através do serviço C2DM. Ao invés disso, quando há notificações novas para um dispositivo, é enviada uma mensagem apenas com o texto "*sync*", que foi escolhida para sinalizar a existência

[8] http://www.sqlite.org

de novas notificações. Assim que recebe esta mensagem, o SciLightning *Mobile* faz uma solicitação ao SciLightning Portal solicitando o sincronismo de dados. A data da última notificação recebida pelo dispositivo é passada como parâmetro da solicitação para que o Portal só "responda" as notificações que ainda não foram recebidas por aquele dispositivo. A interação entre o SciLightning Monitor, Móbile e Portal são apresentados no diagrama de sequência a seguir:

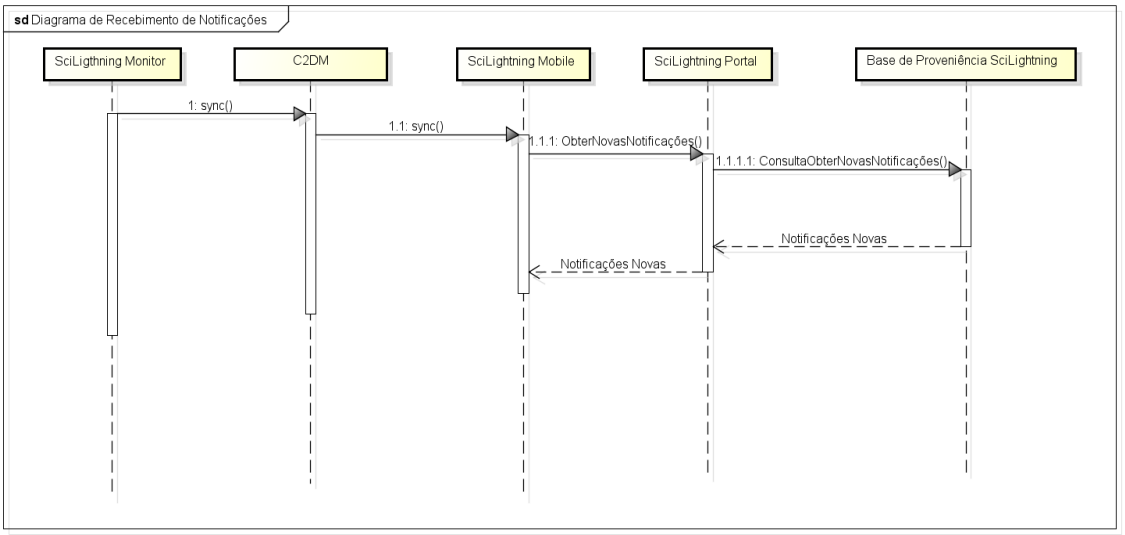

Figura 11 Diagrama de Sequência do Registro do dispositivo móvel

Ao receber uma notificação, é criada uma mensagem na barra de *status* do aparelho contendo o nome do *workflow* um resumo da mensagem (Figura 12). Mesmo que seja recebida mais de uma notificação em sequência, é mantida apenas uma mensagem na barra de *status* referente à última notificação recebida a fim de evitar sobrecarga de informação para o cientista.

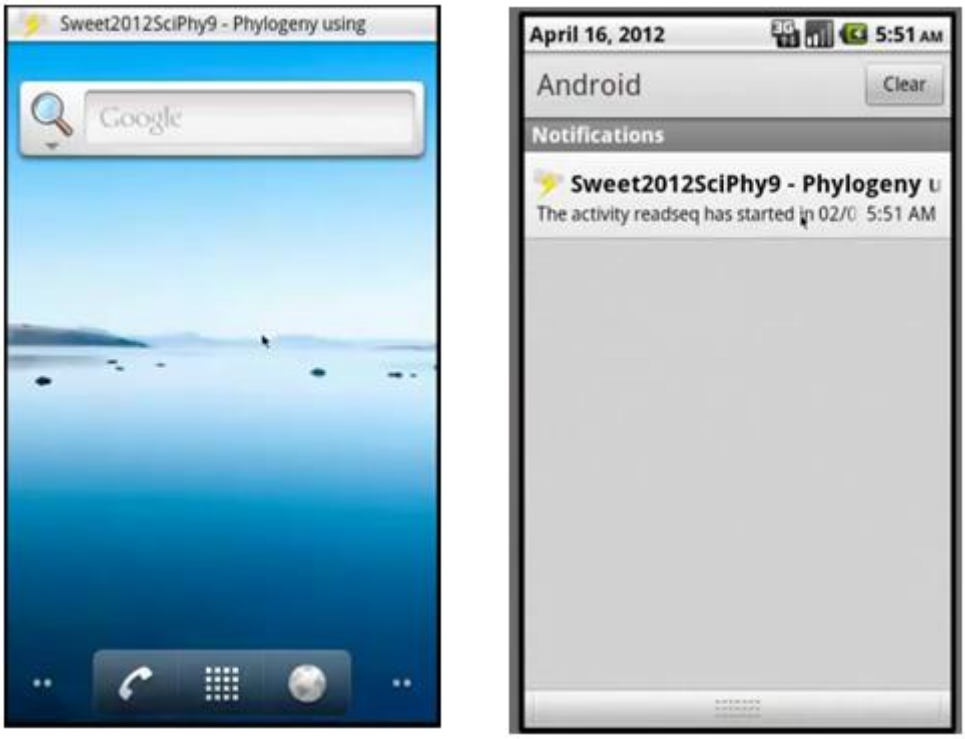

Figura 12 Recebimento de Notificações SciLightning *Mobile*

Ao selecionar a mensagem na barra de *status* ou abrir a aplicação através do menu de aplicações *Android*, é aberta a tela principal da aplicação SciLightning *Mobile*. Esta tela possui um resumo das notificações recebidas agrupadas por *workflow*. Para cada *workflow*, são exibidos o nome do *workflow*, número total de notificações recebidas, número de notificações não lidas e o tempo desde o recebimento da última notificação (Figura 13). Os *workflows* que possuírem notificações não lidas, além de exibir a quantidade das notificações nesta situação, são destacados por uma cor diferente das demais. A ordenação dos *workflows* é realizada de maneira decrescente em relação à data da última notificação recebida, ou seja, os *workflows* que receberam mensagens por último são mantidos no topo da lista.

Figura 13 Tela principal do SciLightning *Mobile*

Quando algum *workflow* é selecionado, é aberta uma tela com o detalhe de todas as notificações recebidas referentes àquele *workflow*. Para cada notificação, são exibidas a mensagem, data e hora de recebimento e um ícone identificando o tipo de notificação (Figura 14). Foi escolhido um ícone diferente para identificar cada tipo de notificação com o objetivo de facilitar a identificação do tipo de notificação e a procura por uma notificação em específico. De maneira semelhante ao que acontece na tela principal, as notificações de um determinado *workflow* são ordenadas de maneira decrescente, a data/horário de recebimento e as notificações novas são destacadas por uma coloração diferente de forma que o cientista encontre facilmente as notificações de um determinado tipo e identifique de maneira mais simples e rápida o tipo de notificação.

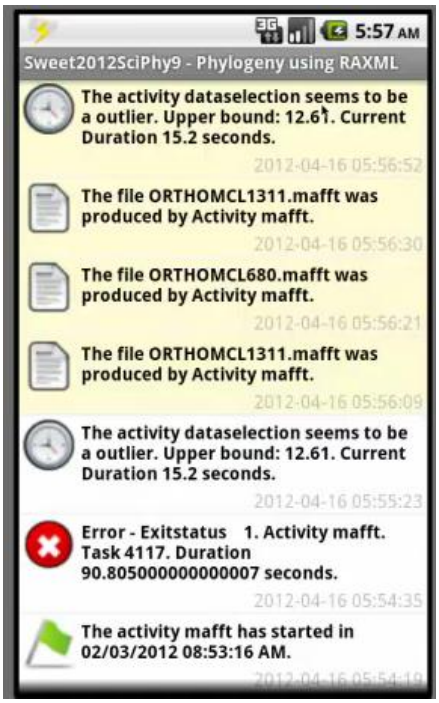

Figura 14 Tela de notificações de *Workflow* do SciLightning *Mobile*

3.4.4 Implementação do SciLightning Social

O SciLightning Social é um componente que faz a interface entre o SciLightning e as redes sociais, conforme explicado na Seção 3.2.6. Nesta dissertação, o componente SciLightning Social foi implementado através da rede social e servidor para *microblogging Twitter*[9]. O *Twitter* possui uma API de desenvolvimento que permite o acesso a várias funcionalidades da ferramenta. A implementação *Twitter* do SciLightning Social está associada a uma conta específica (*@SciLightning*) configurada para enviar notificações sobre a execução de *workflows* aos cientistas por meio dos serviços de mensagens.

Foram escolhidos dois mecanismos para enviar as notificações para o cientista, um que envia mensagens privadas e outro que envia mensagens públicas, estando de acordo com a arquitetura proposta na Seção 3.2.6. A mensagem direta, que é uma mensagem pessoal entre dois usuários (*i.e.* cientistas), foi escolhida como o modo privado de envio de mensagens. Para receber notificações por meio de mensagens

[9] https://twitter.com/

diretas, o cientista deve obrigatoriamente seguir usuário do SciLightning. A Figura 15 apresenta o recebimento de notificações de monitoramento por meio de mensagens diretas ao usuário.

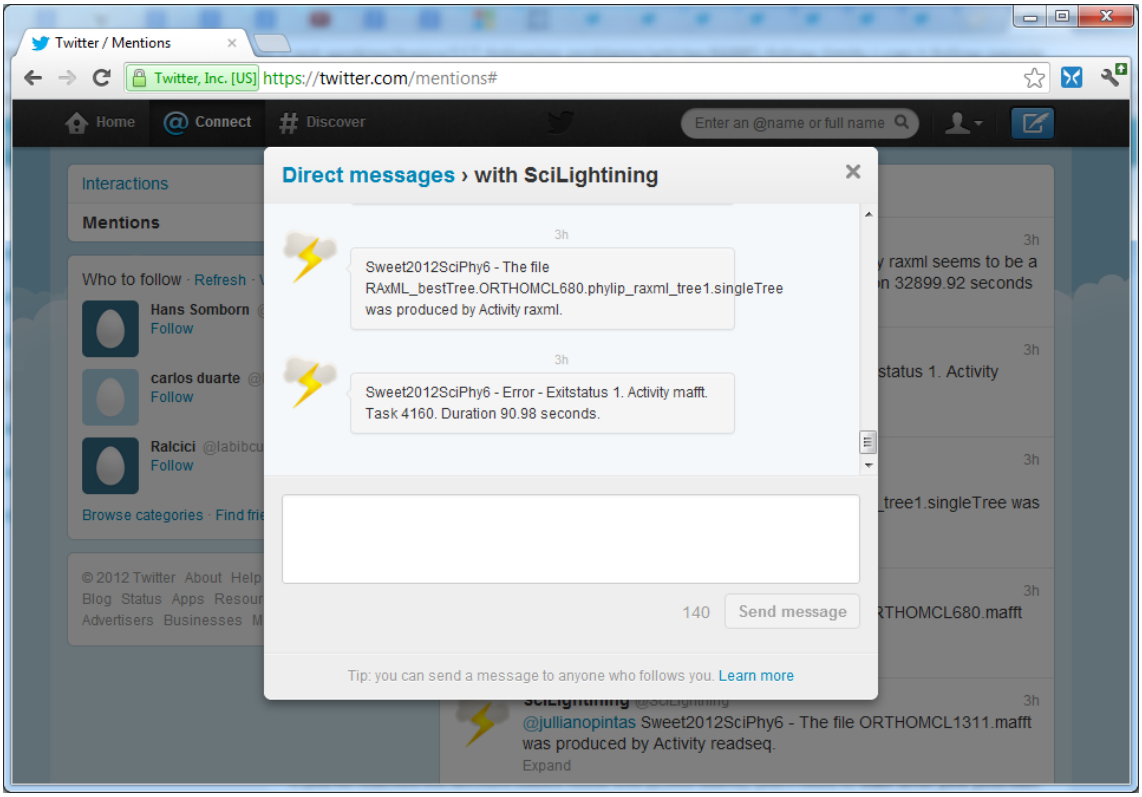

Figura 15 Recebimento de notificações através de mensagens diretas - SciLightning Social

Já o *tweet* normal, que é uma mensagem pública (pode ser visualizada por qualquer usuário), foi escolhido como modo público de envio de mensagens. Os *tweets* podem ser visualizados pelo cientista mesmo que ele não esteja seguindo o usuário do SciLightning, através da página de perfil do usuário SciLightning. Porém, é conveniente que o cientista esteja seguindo o usuário que envia as notificações, pois desta forma os *tweets* aparecerão diretamente no histórico da sua página inicial. Como os *tweets* não são mensagens pessoais como as mensagens diretas, é necessário realizar a menção de qual é o usuário destinatário da notificação. No *Twitter*, uma menção é realizada adicionando o nome de usuário precedido pelo símbolo "@" ao texto da mensagem. Desta forma, aquele *tweet* também será exibido na aba *Menções* deste usuário. A imagem abaixo (Figura 16) mostra as notificações recebidas por meio de *tweets* normais e visualizadas na aba *Menções*.

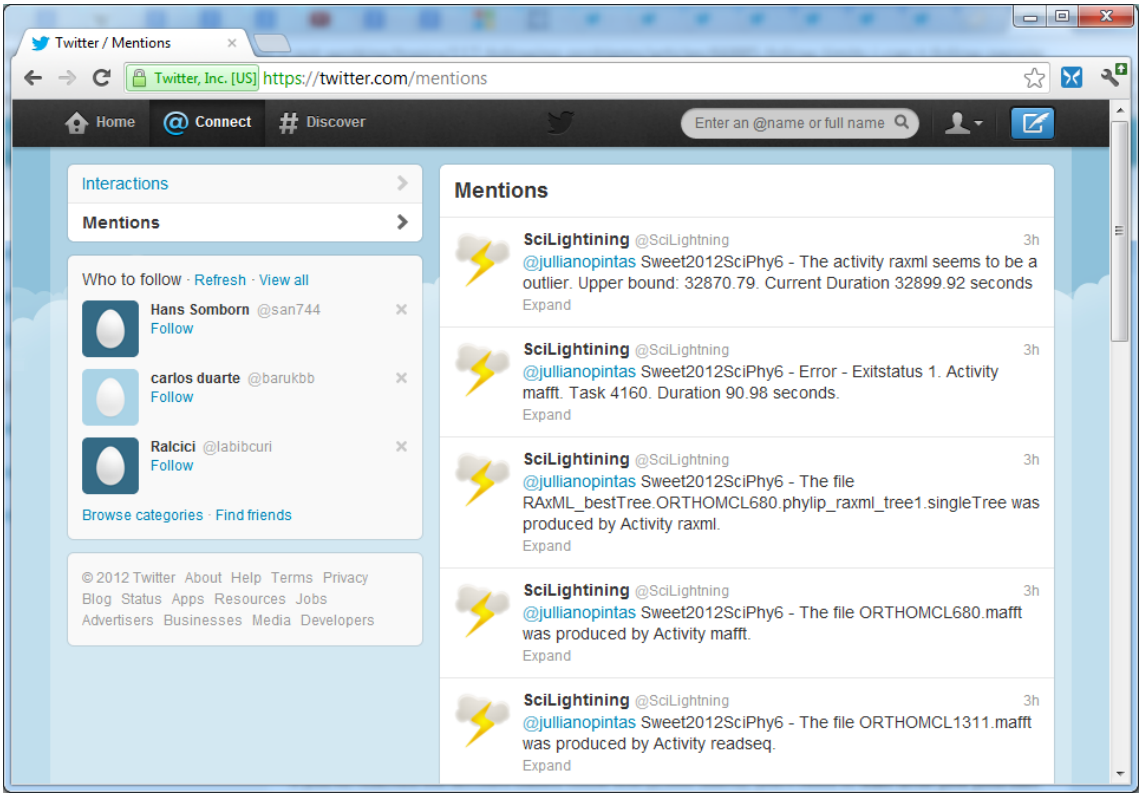

Figura 16 Recebimento de notificações através de *tweets* públicos - SciLightning Social

O cientista pode configurar por meio do portal *Web* o recebimento de notificações através do SciLightning *Mobile* e/ou SciLightning Social. No mesmo portal, durante a configuração de um monitoramento de um *workflow* no SciLightning Portal, o cientista define qual é o modo desejado de recebimento das mensagens através de redes sociais (privado ou público).

A biblioteca Twitter4J foi utilizada com o objetivo de facilitar a comunicação do ScliLightning Social com o *Twitter*, já que esta biblioteca encapsula todas as requisições HTTP e tratamentos de resposta necessários para a comunicação de uma aplicação *Java* com o *Twitter* (Twitter4j 2011). Para autenticar a aplicação junto aos servidores do *Twitter*, foi utilizado o mecanismo de autenticação *OAuth*, que é um padrão aberto no qual não é necessário o compartilhamento das credenciais (tipicamente, um nome de usuário e senha), mas de *tokens* fornecidos pela aplicação autenticadora (OAuth 2010).

O *Twitter* impõe limites severos na taxa de requisições feitas por uma aplicação de acordo com o método de autorização utilizado. São permitidas até 150 chamadas por hora para chamadas não autenticadas. Já para chamadas autenticadas através do OAuth,

que é o caso do SciLightning Social, são permitidas até 350 chamadas/hora (Twitter 2012a). Além desse limite de taxa de chamadas à API do *Twitter*, também são aplicados limites mais rigorosos de acordo com os tipos de serviços utilizados. Cada conta, incluindo contas de aplicativos que é o caso desta implementação do SciLightning Social, pode enviar no máximo 250 mensagens diretas e 1000 *tweets* normais por dia (Twitter 2012b).

O SciLightning Social é mais prático do que o SciLightning *Mobile*, pois pode ser utilizado por meio de qualquer dispositivo que possua um browser e acesso à internet, como dispositivos móveis que não utilizam *Android* e computadores pessoais convencionais. Também é possível aproveitar as funcionalidades nativas do *Twitter*, como o envio de *e-mail* automático no recebimento e compartilhamento de mensagens com outros usuários. Por outro lado, o SciLightning *Mobile* não sofre das restrições impostas pela utilização desta ferramenta de *microblogging*, como o limite máximo de tamanho da mensagem. Além de ter a oportunidade de organizar as notificações de uma maneira mais conveniente e oferecer funcionalidades que não seriam possíveis através do *Twitter*.

Capítulo 4 - Avaliação Experimental

O SciLightning foi projetado para permitir o monitoramento em tempo real da execução de *workflows* científicos em larga escala executados em ambiente distribuído. Como esses tipos de *workflows* tendem a executar por longos períodos de tempo (semanas ou até meses) e as falhas são inerentes aos ambientes distribuídos, mecanismos de monitoramento são fundamentais para que os cientistas acompanhem sua execução. Entretanto, esses mesmos motivos que tornam os mecanismos de monitoramento fundamentais, tornam o monitoramento algo complexo de ser realizado e propenso a falhas. Ou seja, efetuar o monitoramento em um ambiente distribuído durante um longo período de tempo é algo realmente complexo de ser realizado e propenso a falhas.

Por todos esses motivos, o SciLightning foi submetido a uma série de testes e a um experimento final para avaliar sua capacidade de monitoramento de execução dos *workflows* executados em paralelo em ambientes distribuídos durante longos períodos de tempo. Tanto os testes iniciais quanto o experimento final se basearam na execução do *workflow* SciPhy (Ocaña, Oliveira, Ogasawara, et al. 2011) pela abordagem de execução SciCumulus (Daniel Oliveira et al. 2010). Os principais conceitos do SciCumulus foram apresentados na seção 2.4, enquanto que o *workflow* SciPhy será detalhado neste capítulo, na seção 4.1.

Devido à grande quantidade de componentes do SciLightning e à complexidade de se executar um experimento de larga escala, a avaliação de eficácia do SciLightning foi dividida em três etapas. A primeira etapa, apresentada na seção 4.2, avaliou isoladamente as consultas que compõem o cartucho, já que nelas se encontram todas as regras que geram as notificações. Já a segunda etapa, apresentada na seção 4.3, avaliou a integração de todos os componentes do SciLightning por meio de uma série execuções do *workflow SciPhy* em um ambiente reduzido. Por último, foi executado um experimento final que é apresentado na seção 4.4, no qual foi realizada a análise do monitoramento de uma execução completa do *workflow* SciPhy que durou cerca de 6,3 dias sendo executado paralelamente por 16 máquinas virtuais.

Figura 17 Pirâmide da avaliação do SciLightning

As três etapas de avaliação podem ser vistas como uma pirâmide(Figura 17), onde cada uma delas serve como base para a seguinte. A avaliação das consultas serviu principalmente para melhorar o SciLightning para que a avaliação de integração fosse a mais bem sucedida possível. A avaliação da integração dos componentes, por sua vez, teve como principal objetivo preparar o SciLightning para a avaliação experimental. Quanto mais alto na pirâmide, o ambiente no qual é feito a análise é mais próximo do mundo real, ao mesmo tempo cada ciclo de avaliação é mais complexo, demorado e custoso. Tanto a avaliação das consultas quanto a de integração dos componentes foram realizadas de maneira iterativa com o objetivo de descobrir falhas e oportunidades de melhoria, permitindo assim o aprimoramento dos componentes do SciLightning. Por tal motivo, as seções 4.2 e 4.3, além de apresentar o processo de análise, indicam também quais as correções e melhorias mais significativas que foram realizadas como resultado do processo. Já a avaliação experimental final teve por objetivo fornecer um parecer final sobre o funcionamento do SciLightning em um caso real de monitoramento de uma execução de um *workflow* de larga escala.

Na avaliação das consultas não foi executado nenhum *workflow* de fato, pois ficou decidido em se utilizar os dados de rodadas de *workflows* anteriores para realizar esta análise. Já a análise de integração e a análise experimental foram baseadas na

execução do SciPhy pelo SciCumulus no ambiente de nuvem da Amazon EC2. O Amazon EC2 é um dos ambientes de computação em nuvem mais populares e muitas aplicações científicas e comerciais estão sendo implantadas sobre ele (Armbrust et al. 2010; Hey, Tansley, e Tolle 2009; Hoffa et al. 2008; Matsunaga, Tsugawa, e Fortes 2008; Ocaña, Oliveira, Ogasawara, et al. 2011; Ocaña, Oliveira, Dias, et al. 2011). Na análise de integração foi utilizado um ambiente simplificado (apenas 2 máquinas) e os dados de entrada do *workflow* e seus parâmetros foram manipulados para que o SciLightning fosse analisado em várias situações de execução diferentes de maneira mais simples.

4.1 SciPhy

Experimentos filogenéticos são exemplos de *workflows* que demandam PAD e que são compostos por milhares de execuções de atividades. Os *workflows* filogenéticos visam produzir árvores filogenéticas para apoiar as relações evolutivas existentes entre organismos que são utilizadas por biólogos para inferir uma filogenia (*e.g.* as relações ancestrais entre espécies de organismos conhecidos). O SciPhy (Ocaña, Oliveira, Ogasawara, et al. 2011) é um exemplo de *workflow* de análise filogenética que tem como objetivo o processamento de uma grande coleção de arquivos multi-fasta (cada um contendo várias sequências biológicas de organismos diferentes) para obter árvores filogenéticas. Ele auxilia biólogos a explorar filogenia e determinar a vida evolutiva de genes ou genomas destes organismos. O *workflow* SciPhy é composto por quatro atividades principais: (i) A construção do alinhamento múltiplo de sequências (AMS); (ii) A conversão de formato do alinhamento; (iii) A pesquisa e eleição do melhor modelo evolutivo a ser usado; e (iv) A construção da árvore filogenética. Estas atividades, respectivamente, executam os seguintes programas (*i.e.* aplicações) de bioinformática: programas de AMS (permitindo ao cientista escolher entre o MAFFT, o Kalign, o ClustalW, o Muscle, ou o ProbCons), o ReadSeq, o ModelGenerator e o RAxML. O SciPhy foi projetado para ser executado em paralelo onde cada arquivo multi-fasta é processado de forma independente nas diversas VM e, devido às flutuações de desempenho no ambiente de nuvem, uma execução típica do SciPhy apresenta entre 2% e 9% de falhas de execução de atividades, o que exige que o biólogo fique acompanhando a execução para ter ciência do seu estado. Mais informações sobre o SciPhy podem ser encontradas em Ocaña *et al.* (2011).

4.2 Avaliação das regras das consultas de monitoramento

Nesta dissertação, o cartucho que permitiu o acoplamento com o SciCumulus foi implementado por meio de uma pilha de visões de banco de dados. Como todas as regras das notificações são implementadas diretamente no cartucho (conforme explicado na seção 3.2.1), a exatidão na definição dessas visões é essencial para um funcionamento correto do SciLightning. Por tal motivo, antes mesmo de começar a avaliar o funcionamento do SciLightning como um todo, as consultas que compõem essas visões passaram por um processo iterativo de avaliação e ajuste até que estivessem totalmente estáveis.

Como o objetivo desta avaliação em específico era analisar as consultas de maneira isolada aos outros componentes do SciLightning com a finalidade de se obter maior controle na avaliação, foi decidido utilizar o histórico de execuções passadas que já estavam armazenadas na base de proveniência do SciCumulus ao invés de monitorar uma execução em tempo real. Além de permitir avaliar as consultas por meio de dados de várias execuções ao mesmo tempo, a decisão de usar dados já armazenados permitiu a realização de uma curadoria nestes dados.

A cada iteração do processo de avaliação foi verificado se as consultas gerariam o conjunto correto de notificações retroativas para todas as execuções passadas. Em outras palavras, o objetivo foi simular o monitoramento de execuções anteriores e verificar se as notificações geradas por cada consulta estavam de acordo com aquilo que estava registrado na base de proveniência monitorada. Para isso, foram utilizados vários cenários, cada qual com um conjunto de configurações de monitoramento diferente. Esses cenários foram criados levando em consideração a estrutura interna das consultas para que fosse possível abranger o máximo de regras. Por exemplo, foram utilizados cenários com marcos tanto de início de atividade quanto de fim de atividade, uma vez que já era conhecido que a lógica para cada tipo de marco estava em uma parte diferente da consulta.

Todo o processo de avaliação e melhoria das consultas aconteceu de maneira iterativa, sendo que, ao final de cada iteração, composta por uma sequência de atividades, fosse obtida melhorias nas consultas. O ciclo se repetiu até que as consultas estivessem estáveis e robustas, ou seja, quando a iteração não apresentava mais

nenhuma oportunidade para melhoria. O diagrama abaixo (Figura 18) apresenta esse processo de melhoria das consultas. Esse mesmo processo pode ser aplicado sempre que for necessária a implementação ou modificação de um Cartucho SciLightning.

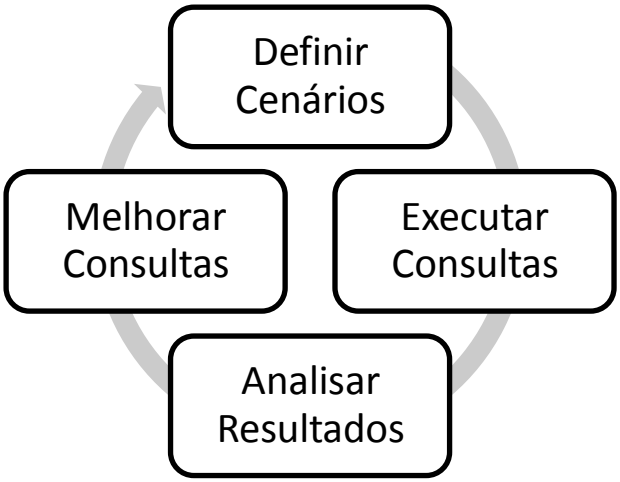

Figura 18 Processo de melhoria das consultas de notificações

Antes da primeira iteração, foi necessário ajustar as consultas para que elas, ao invés de considerar apenas a execução atual, também considerem a proveniência de execuções anteriores. A primeira atividade de cada iteração é a de definir um conjunto de configurações de monitoramento que vão compor o cenário de avaliação daquele ciclo. Aspectos internos das consultas devem ser levados em consideração para escolha das melhores combinações de configurações para identificar falhas e oportunidades de melhorias nas consultas. Também é importante se atentar para as melhorias realizadas no ciclo anterior, a fim de se certificar que as alterações realmente surtiram efeito e não afetaram o restante da lógica da consulta. Todas as configurações são registradas normalmente através do SciLightning Portal.

Já com os cenários definidos e registrados, as consultas são executadas de fato e seus resultados armazenados em uma tabela auxiliar do banco. Todas as consultas são executadas utilizando a ferramenta de banco de dados. Na próxima atividade, chamada de *Analisar Resultados*, os resultados das consultas são comparados com as configurações registradas e com o que está registrado na base de proveniência. O objetivo principal dessa atividade é tentar encontrar alguma inconsistência ou oportunidade de melhoria. As inconsistências encontradas podem ser agrupadas em 3 tipos principais: (i) Falso positivo, quando alguma notificação é gerada para um evento que não deveria gerar notificações; (ii) Falso negativo, quando nenhuma notificação é

gerada para o evento que deveria gerar notificações; (iii) Um ou mais problemas nas informações das notificações com problemas, quando a notificação é gerada no evento correto, porém existe algum problema em suas informações. Por exemplo, um problema na informação seria o formato da mensagem da notificação não estar de acordo com o que foi definido na seção 3.1 ou os destinatários não estarem de acordo com o que foi definido na atividade *Definir Cenário*. As oportunidades de melhorias encontradas foram de dois tipos principais: baixo desempenho da consulta ou grande complexidade no código das consultas. A figura abaixo mostra a evolução do número de inconsistências e de oportunidades de melhorias encontradas ao longo das iterações do processo de melhoria.

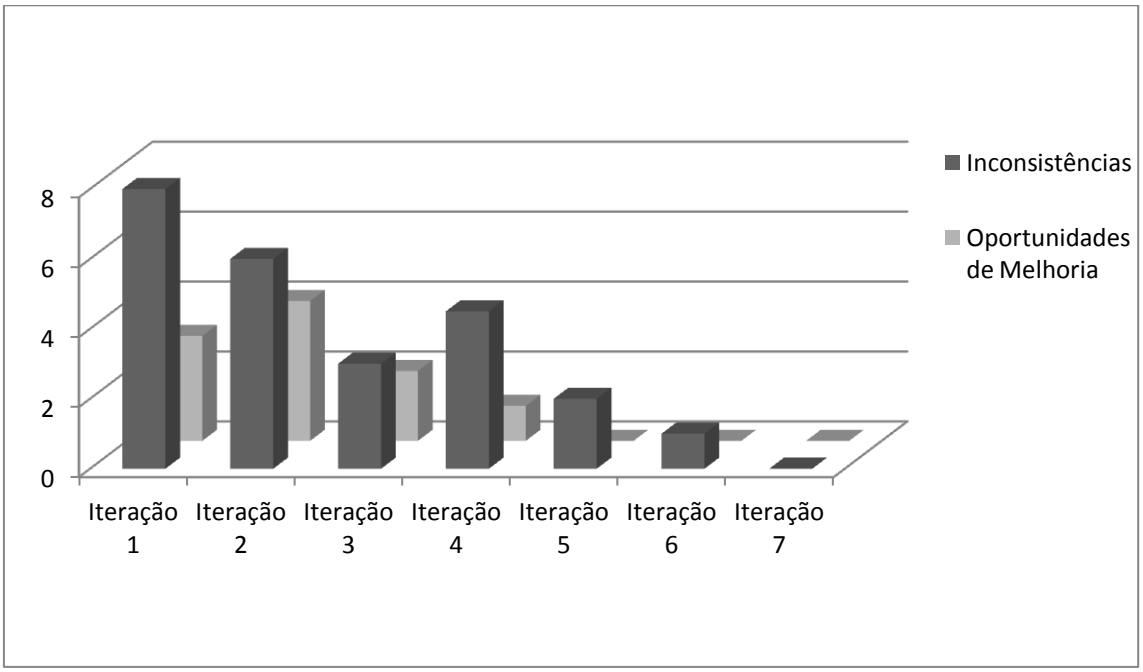

Figura 19 Gráfico das iterações da avaliação das consultas de monitoramento

Cada inconsistência encontrada foi analisada detalhadamente para saber se existia realmente alguma falha na consulta ou se aquela inconsistência era derivada de alguma inconsistência nos dados armazenados. Nos casos em que foi detectado que a inconsistência era realmente derivada de uma falha na consulta, as devidas alterações foram realizadas através da atividade *Analisar Resultados.* Cada oportunidade de melhoria também foi analisada e, sempre que possível, os códigos das consultas foram adaptados para conseguir maior desempenho e simplicidade. A identificação das oportunidades de melhoria permitiu a criação de consultas auxiliares com o objetivo de

diminuir a complexidade ou melhorar o desempenho da consulta principal. Por exemplo, a consulta da visão principal de monitoramento de notificações de tarefas atípicas, chamada *v_monitor_unusual*, foi identificada como oportunidade de melhoria, pois possuía o código bem complexo. Por este motivo, essa consulta foi divida em mais uma subconsulta que ficou responsável especificamente por calcular a média e o desvio-padrão das durações de tarefas agrupadas por tipo de atividade, originando então a visão *v_monitor_stddev*.

Como dito anteriormente, a avaliação das consultas serviu também para realizar a curadoria dos dados armazenados na base de proveniência. À medida que as consultas iam sendo executadas para cada cenário, foram identificados padrões nas informações armazenadas no banco. Com os padrões bem definidos, foi possível identificar quais informações estavam fora dos padrões de normalidade. Um caso curioso foi a identificação de um problema em um registro de execução de uma tarefa devido a ajustes de horário de verão. A tarefa, que é do tipo *mafft*, apresentou duração de mais de uma hora, enquanto que a média para seu tipo de atividade era de 67,97 segundos e o desvio padrão era de 247,81 segundos. Com uma duração de 3.687,41 segundos, a tarefa foi identificada como possuindo tempo de execução anormal, estando 14,87 desvios-padrão distante da média. Quando foi realizada uma análise mais detalhada, notou-se que a tarefa possuía horário de início às 23:59:31.86 do dia 15 de outubro de 2011 e horário de término ás 01:00:57.27 do dia 16 de outubro de 2011. Foi confirmado que a máquina responsável por executar aquele *workflow* realmente estava configurada para ajustar automaticamente o horário do sistema de acordo com o horário de verão brasileiro. Conclui-se, portanto, que a duração da atividade foi distorcida devido a esta alteração do ambiente computacional.

4.3 Avaliação da Integração dos componentes

A seção anterior apresentou o processo no qual todas as consultas do cartucho foram validadas e otimizadas. Apesar da grande importância em se avaliar isoladamente a exatidão das consultas que compõem o cartucho, já que elas possuem toda regra de notificação, também é fundamental avaliar o funcionamento do SciLightning como um todo. Ou seja, validar o funcionamento de todos os componentes do SciLightning quando estão trabalhando em conjunto. A avaliação apresentada nesta seção é baseada em um processo onde cada iteração executa o que seria equivalente a um teste de

integração e as devidas correções são efetuadas antes de iniciar a próxima iteração até que os componentes estejam perfeitamente integrados. Na engenharia de software, um teste de integração é o processo de verificar se os componentes do sistema, juntos, trabalham conforme descrito nas especificações do sistema e do projeto do programa (Pfleeger 2001).

A avaliação da integração dos componentes já foi realizada no ambiente de nuvem da Amazon EC2 como na avaliação experimental final, porém com uma quantidade reduzida de instâncias. Foram instanciadas apenas 2 máquinas virtuais tipo *large* (7,5 GB RAM, 850 GB de armazenamento, 2 núcleos virtuais). Cada máquina virtual instanciada é baseada no sistema operacional Cent OS 5 (64 bits).

Ao contrário do que ocorreu na avaliação experimental final, nesta avaliação de integração, os dados de entrada do *workflow* e seus parâmetros foram manipulados para que o SciLightning fosse analisado em várias situações de execução diferentes. Em algumas iterações, os dados de entrada foram corrompidos propositalmente para que fossem geradas várias notificações de erro na execução das tarefas do *workflow*. Em outras iterações, os dados de entrada foram reduzidos ou aumentados para que notificações de anormalidade no tempo de execução fossem geradas.

O processo de avaliação da integração ocorreu de maneira semelhante ao processo de avaliação das consultas que foi apresentado na seção anterior. Todo o processo de avaliação da integração dos componentes também aconteceu de maneira iterativa, sendo que, ao final de cada iteração, composta por uma sequência de atividades, fossem realizadas correções e melhorias nos componentes. O ciclo se repetiu até não houvesse mais falhas ou oportunidades de melhorias no funcionamento de todos os componentes, ou seja, quando a iteração não apresentou mais nenhuma oportunidade para melhoria. O diagrama da Figura 20 apresenta o ciclo de atividades do processo de avaliação de integração dos componentes.

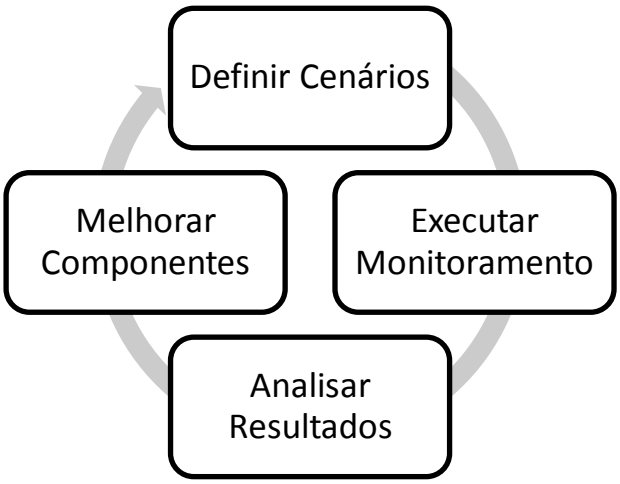

Figura 20 Processo de avaliação e melhoria da integração dos componentes

Logo na primeira iteração da avaliação foi verificado um problema para identificar quais tarefas possuíam duração atípica antes de seu término. Ao contrário do que foi especificado na seção 3.1, as notificações de duração atípica só estavam sendo geradas e enviadas após o término da tarefa. Ao fazer uma análise mais detalhada se descobriu que o SciCumulus, por questão de performance, só preenchia o tempo de início da tarefa no final de execução da mesma. Este problema não pôde ser identificado na avaliação das consultas, pois os dados utilizados já estavam preenchidos anteriormente.

Como foi adotado como premissa de desenvolvimento do SciLightning, minimizar ao máximo o impacto na performance da abordagem de execução que estivesse sendo monitorada, foi descartada a opção mais direta, que era a de alterar o processamento do SciCumulus para que a data de início fosse inserida assim que a execução da tarefa fosse iniciada. Alternativamente, decidiu-se criar uma consulta auxiliar para estimar o início da atividade, originando assim a visão *v_monitor_starttime*. Esta visão estima a data/horário de início de execução de uma tarefa baseando-se no horário de término da última tarefa que foi executada em cada máquina.

Na terceira iteração da avaliação o SciLightning foi submetido a uma situação onde foram gerados inúmeros arquivos que atendessem as expressões regulares definidas na configuração do monitoramento em um curto período de tempo. Notou-se então que as notificações foram registradas no histórico de notificações, porém se perderam antes de chegar a aplicação do SciLightning *Mobile*. Descobriu-se que as

perdas ocorreram pelo serviço de entrega utilizado, o C2DM, que não é preparado para enviar uma grande quantidade de conteúdo através das mensagens. Portanto o componente *Mobile* foi preparado para, ao invés de enviar as mensagens diretamente do C2DM, utilizar o C2DM apenas para avisar a aplicação *Android* que existem notificações novas. Ao receber esta mensagem, a própria aplicação *Android* se reponsabiliza por requisitar as notificações novas para o SciLightning Portal.

4.4 Avaliação Experimental Final

Após serem avaliadas as consultas e a integração dos componentes, foi realizada uma avaliação experimental final através de uma execução completa de um *workflow* real em um ambiente real de alto desempenho. O objetivo deste experimento foi avaliar o comportamento do SciLightning numa situação real de execução de experimento científico de larga escala, onde são utilizados ambientes altamente distribuídos e a execução tende a durar um longo período de tempo.

O *workflow* SciPhy (Ocaña, Oliveira, Ogasawara, et al. 2011) foi escolhido como estudo de caso. O *workflow* SciPhy foi executado pelo SciCumulus no ambiente de nuvem da Amazon EC2, sendo utilizadas 16 máquinas virtuais tipo *large* (7,5 GB RAM, 850 GB de armazenamento, 2 núcleos virtuais). Cada máquina virtual instanciada é baseada no sistema operacional Cent OS 5 (64 bits).

Como a base de proveniência do SciCumulus que foi utilizada pelas outras duas avaliações é comumente usada para testes, decidiu-se utilizar uma base de proveniência inicialmente vazia para este experimento. Isto garante que não haja distorções no monitoramento, principalmente no que tange ao monitoramento de tarefas com duração atípica, devido a dados inconsistentes na base de proveniência.

O experimento usou como entrada um conjunto de 250 arquivos multi-fasta contendo sequências de proteínas extraídas do banco de dados biológico RefSeq *release* 48. Cada entrada de arquivo multi-fasta foi processada usando as seguintes versões dos programas: ClustalW versão 2.1, Kalign versão 1.04, MAFFT versão 6,857, Muscle versão 3.8.31, PROBCONS versão 1.12, ModelGenerator versão 0.85, ReadSeq versão 2.1.26 e RAxML-7.2.8-ALPHA, exatamente conforme detalhado por Ocaña *et al.* (2011). Foi executado um total de 5.250 tarefas paralelas e a execução durou aproximadamente 6,3 dias.

Para este experimento foram adotadas as seguintes configurações de monitoramento. O envio de notificações nos seguintes casos: (i) Na ocorrência de qualquer tipo de erro; (ii) No término da execução do *workflow*; (iii) No início da atividade ModelGenerator e no fim da atividade ModelGenerator; (iv) Na geração de arquivos cujos nomes respeitem a expressão regular *^[A-Z]*.mafft*; e (v) Nas atividades com duração atípica que esteja 2,69 desvios-padrão distante da média. Para a avaliação dos resultados, foram utilizadas métricas de precisão e cobertura (Baeza-Yates e Ribeiro-Neto 2011) para verificar o grau de acerto de cada um dos cinco tipos de notificações. A métrica *Precisão* é a porcentagem do conjunto de eventos notificados que representa os eventos relevantes. A métrica *Cobertura* é a porcentagem dos eventos relevantes que foram recuperados. A Tabela 5 apresenta os resultados obtidos.

Tabela 5 Resultados Experimentais

Tipo de Notificação	Total de Itens	Itens Notificados	Itens Relavantes Recuperados	Precisão	Cobertura
Erro em Atividade	73	73	73	100%	100%
Término de Execução do *Workflow*	1	1	1	100%	100%
Início e Fim de Atividade	2	2	2	100%	100%
Geração de Arquivos	1.701	1.701	1.701	100%	100%
Atividades com Duração Atípica	38	49	38	77%	100%

Para cada um dos cinco tipos de notificação configurados, o SciLightning apresentou 100% de cobertura, o que significa que é capaz de identificar os eventos e enviar as notificações com base em dados de proveniência gerados em tempo de execução. O único caso em que o SciLightning não conseguiu 100% de precisão foi na notificação de duração atípica de atividades. Como explicado anteriormente, uma atividade tem sua execução classificada como atípica se seu tempo de execução supera 2,69 desvios-padrão em relação à média dos tempos de execução já existentes no repositório de proveniência. Neste caso, não existiam dados de proveniência previamente registrados no repositório, o que fez com que as primeiras execuções de atividades sempre fossem consideradas *outliers*. À medida que novos tempos foram

computados, esse comportamento atípico cessou. Um vídeo de demonstração do SciLightning pode ser acessado em http://sites.google.com/site/scilightning/.

O SciLightning é executado em uma máquina separada do ambiente de execução do *workflow*, não acessando as máquinas que estão executando os processos. Portanto, o único impacto no desempenho da execução do *workflow* é a realização de consultas na mesma base de proveniência que está sendo alimentada pelo motor de execução. Como nesta avaliação experimental foi identificado que as consultas periódicas realizadas pelo SciLightning demoraram em média apenas 0,5 segundos, conclui-se que o SciLightning não impacta de maneira significante a execução do *workflow*.

Capítulo 5 - Conclusões e trabalhos futuros

Experimentos científicos em larga-escala envolvem diversas execuções de *workflows* científicos computacionalmente intensivos. Estes *workflows* precisam ser executados em paralelo em ambientes de PAD, como nuvens de computadores. No entanto, uma vez que a execução de um *workflow* pode envolver 100.000 atividades ou mais, demorando semanas ou meses para terminar, é inviável que o cientista acompanhe esta execução do computador. Monitorar uma execução de *workflow* é fundamental para garantir a confiabilidade na execução e para prover mecanismos de condução do *workflow*. Para que um cientista possa interferir na execução de um *workflow*, o mesmo deve ter ciência do que está acontecendo na execução. No entanto, nenhuma das abordagens existentes para *workflows* que demandam PAD focam em prover mecanismos de monitoramento e notificação em tempo real. Nos melhores cenários, o cientista tem que realizar o acompanhamento através de consultas diretas à base de proveniência ou de *logs*.

O objetivo principal desta dissertação foi apresentar e avaliar uma nova abordagem no que tange ao monitoramento da execução de *workflows* em larga escala. Esta abordagem utiliza a proveniência gerada em tempo de execução para notificar eventos durante a execução do *workflow* como erros em atividades, produção de arquivos e término do *workflow*. O processo de monitoramento da proveniência em tempo real é automatizado, permitindo que o cientista defina previamente, através de parâmetros de monitoramento, quais eventos ele tem interesse em monitorar em cada execução de *workflow*. A cada evento ocorrido na execução do *workflow* que atenda os parâmetros definidos, o sistema de monitoramento notifica automaticamente o cientista através de tecnologias de dispositivo móveis e redes sociais. Portanto, a abordagem proposta se baseia principalmente na existência de um mecanismo que participe ativamente do monitoramento, notificando automaticamente o cientista de forma ubíqua sobre eventos que lhe sejam importantes. Esta dissertação se baseou no seguinte artigo completo:

> Pintas, J., Oliveira, D.,Ocaña, K., Dias, J., Mattoso, M., 2012. "Monitoramento em Tempo Real de *Workflows* Científicos Executados em Paralelo em Ambientes Distribuídos". In *VI e-Science workshop - XXXII Congresso da Sociedade Brasileira da Computação,* Curitiba, Paraná, Brazil.

O primeiro passo para aplicar esta nova abordagem foi realizar um levantamento para conhecer quais eventos os cientistas mais necessitam monitorar e quais parâmetros seriam necessários para efetuar isso automaticamente. Após isso, este levantamento foi usado como base para o desenvolvimento de um arcabouço de monitoramento de execução de *workflows* científicos chamado SciLightning. O arcabouço SciLightning foi projetado para ser acoplado a qualquer abordagem de execução que forneça dados de proveniência em tempo real e se basear em quaisquer plataformas de dispositivos móveis e redes sociais para desenvolvimento dos mecanismos de envio de notificação.

Nesta mesma dissertação, o arcabouço SciLightning foi instanciado em um sistema completo, onde foi acoplado à abordagem de execução de *workflows* na nuvem chamada SciCumulus, baseando-se na utilização da rede social *Twitter* e numa aplicação *Android* desenvolvida para envio das notificações. Além de oferecer uma maneira mais conveniente de efetuar o monitoramento, o SciLightning proporciona um acompanhamento em tempo real, tornando as ações de condução de execução do *workflow* mais rápidas e menos custosas.

Foi realizada uma série de testes e uma avaliação experimental que comprovaram que SciLightning é capaz de identificar e notificar 100% dos eventos monitorados. Com a exceção da notificação de tempo de execução atípico, todas as notificações tiveram precisão de 100%, o que demonstrou que o SciLightning é confiável e eficaz na identificação e notificação de eventos. O SciLightning possui uma arquitetura de serviço que é genérica e pode ser facilmente acoplada a abordagens para a gestão de execução do *workflows*, como o SciCumulus e outras, desde que os dados de proveniência sejam fornecidos em tempo de execução. Notificar eventos dos *workflows* em tempo de execução é requisito para possibilitar interferências na execução de experimentos como ajustes finos de parâmetros e sua adaptação durante sua execução (Dias et al. 2011). De outra forma, tal adaptabilidade pode ser difícil ou mesmo impossível de ser pré-programada.

5.1 Trabalhos Futuros

A abordagem desta dissertação por si só já apresenta grande utilidade para a comunidade científica. Além disso, ela abre caminhos para trabalhos futuros que não eram vislumbrados nas abordagens anteriores.

Um importante trabalho futuro será possibilitar a condução do *workflow* através do SciLightning *Mobile*, permitindo que o *workflow* seja interrompido ou parâmetros sejam ajustados em tempo de execução.

O SciLightning Social foi implementado nesta dissertação por meio da ferramenta de *microblogging Twitter*. Um trabalho futuro é estender a solução, utilizando outras ferramentas de rede social, como o *Facebook*. Do mesmo modo, o SciLightning *Mobile* foi implementado nesta dissertação através da plataforma *Android*. Um trabalho futuro é estender a solução, utilizando outras plataformas de dispositivos móveis, como o Apple iOS.

Na versão atual da ferramenta, para criar novos tipos de notificação que utilizem parâmetros é necessário realizar uma adaptação do modelo de proveniência SciLightning. Um trabalho futuro será remodelar a base de proveniência do SciLightning criando uma estrutura genérica para parâmetros.

Referências Bibliográficas

Abouzeid, Azza, Kamil Bajda-Pawlikowski, Daniel Abadi, Avi Silberschatz, e Alexander Rasin. 2009. "HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads". *Proceedings of the VLDB Endowment (PVLDB)* 2 (1): 922–933.

de Almeida-Neto, Cesar, Jing Liu, David J. Wright, Alfredo Mendrone-Junior, Pedro L. Takecian, Yu Sun, Joao Eduardo Ferreira, et al. 2011. "Demographic characteristics and prevalence of serologic markers among blood donors who use confidential unit exclusion (CUE) in São Paulo, Brazil: implications for modification of CUE polices in Brazil". *Transfusion* 51 (1): 191–197. doi:10.1111/j.1537-2995.2010.02799.x.

Altintas, I., C. Berkley, E. Jaeger, M. Jones, B. Ludascher, e S. Mock. 2004. "Kepler: an extensible system for design and execution of scientific workflows". In *Scientific and Statistical Database Management*, 423-424. Greece. doi:10.1109/SSDM.2004.1311241.

Anderson, David, Jeff Cobb, Eric Korpela, Matt Lebofsky, e Dan Werthimer. 2002. "SETI@home: an experiment in public-resource computing". *Commun. ACM* 45 (11) (novembro): 61, 56.

Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, et al. 2010. "A view of cloud computing". *Commun. ACM* 53 (4): 50-58. doi:10.1145/1721654.1721672.

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

Barnett, V., e T. Lewis. 1994. *Outliers in Statistical Data*. 3º ed. John Wiley & Sons.

Beberg, Adam L., Daniel L. Ensign, Guha Jayachandran, Siraj Khaliq, e Vijay S. Pande. 2009. "Folding@home: Lessons from eight years of volunteer distributed computing". In *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, 1–8. IPDPS '09. Washington, DC, USA: IEEE Computer Society. doi:10.1109/IPDPS.2009.5160922. http://dx.doi.org/10.1109/IPDPS.2009.5160922.

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

Birsan, Dorian. 2005. "On plug-ins and extensible architectures". *Queue* 3 (2): 40-46. doi:10.1145/1053331.1053345.

Braga, Luis Paulo Vieira. 2010. *Compreendendo Probabilidade e Estatística*. Rio de Janeiro: E-papers Serviços Editorias Ltda. http://books.google.com.br/books?id=jtUOfHuhme8C&lpg=PP1&ots=gSmuQf3JS0&dq=estatistica&pg=PA99#v=onepage&q=desvio%20padr%C3%A3o&f=false.

Buneman, Peter, Sanjeev Khanna, e Wang-chiew Tan. 2001. "Why and Where: A Characterization of Data Provenance". *International Conference on Database Theory*: 316-330. doi:10.1.1.6.1848.

Callahan, Steven P., Juliana Freire, Emanuele Santos, Carlos E. Scheidegger, Cláudio T. Silva, e Huy T. Vo. 2006. "VisTrails: visualization meets data management". In *SIGMOD International Conference on Management of Data*, 745-747. Chicago, Illinois, USA: ACM. doi:10.1145/1142473.1142574. http://portal.acm.org/citation.cfm?id=1142473.1142574.

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

Cirne, Walfredo, Francisco Brasileiro, Nazareno Andrade, Lauro Costa, Alisson Andrade, Reynaldo Novaes, e Miranda Mowbray. 2006. "Labs of the World, Unite!!!" *Journal of Grid Computing* 4 (3): 225-246.

Coutinho, Fábio, Eduardo Ogasawara, Daniel Oliveira, Vanessa Braganholo, Alexandre A. B Lima, Alberto M. R Dávila, e Marta Mattoso. 2011. "Many Task Computing for Orthologous Genes Identification in Protozoan Genomes Using Hydra". *Concurrency and Computation: Practice and Experience* 23 (17) (dezembro 10): 2326-2337. doi:10.1002/cpe.1786.

Couvares, Peter, Tevfik Kosar, Alain Roy, Jeff Weber, e Kent Wenger. 2007. "Workflow Management in Condor". In *Workflows for e-Science*, 357-375. Springer. http://dx.doi.org/10.1007/978-1-84628-757-2_22.

Cruz, Sergio Manuel Serra da, Fabricio Nogueira da Silva, Luiz M. R. Gadelha Jr., Maria Claudia Reis Cavalcanti, Maria Luiza M. Campos, e Marta Mattoso. 2008. "A Lightweight Middleware Monitor for Distributed Scientific Workflows". In *CCGRID '08*, 693-698.

Dantas, M. 2005a. "Clusters Computacionais". In *Computação Distribuída de Alto Desempenho: Redes, Clusters e Grids Computacionais*, 145-180. 1º ed. Rio de Janeiro: Axcel Books.

———. 2005b. "Grids Computacionais". In *Computação Distribuída de Alto Desempenho: Redes, Clusters e Grids Computacionais*, 201-238. 1º ed. Rio de Janeiro: Axcel Books.

Dávila, Alberto M. R., Pablo N. Mendes, Glauber Wagner, Diogo A. Tschoeke, Rafael R. C. Cuadrat, Felipe Liberman, Luciana Matos, et al. 2008. "ProtozoaDB: dynamic visualization and exploration of protozoan genomes". *Nucleic Acids Research* 36 (Database issue): D547-D552. doi:10.1093/nar/gkm820.

Dean, Jeffrey, e Sanjay Ghemawat. 2008. "MapReduce: simplified data processing on large clusters". *Communications of the ACM* 51 (1): 107–113. doi:10.1145/1327452.1327492.

Deelman, Ewa, Gaurang Mehta, Gurmeet Singh, Mei-Hui Su, e Karan Vahi. 2007. "Pegasus: Mapping Large-Scale Workflows to Distributed Resources". In *Workflows for e-Science*, 376-394. Springer. http://dx.doi.org/10.1007/978-1-84628-757-2_23.

Devore, Jay L. 2011. *Probability and Statistics for Engineering and the Sciences*. USA: Brooks/Cole, Cengage Learning.

Dias, Jonas, Eduardo Ogasawara, Daniel Oliveira, Fabio Porto, Alvaro Coutinho, e Marta Mattoso. 2011. "Supporting Dynamic Parameter Sweep in Adaptive and

User-Steered Workflow". In *6th Workshop on Workflows in Support of Large-Scale Science*, 31-36. WORKS '11. Seattle, WA, USA: ACM.

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

Fahringer, T., R. Prodan, Rubing Duan, F. Nerieri, S. Podlipnig, Jun Qin, M. Siddiqui, Hong-Linh Truong, A. Villazon, e M. Wieczorek. 2005. "ASKALON: a Grid application development and computing environment". In *6th IEEE/ACM International Workshop on Grid Computing*, 122-131. Seattle, Washington, USA: IEEE. doi:10.1109/GRID.2005.1542733.

Fileto, Renato, Ling Liu, Calton Pu, Eduardo Delgado Assad, e Claudia Bauzer Medeiros. 2003. "POESIA: An ontological workflow approach for composing Web services in agriculture". *The VLDB Journal* 12 (4) (novembro): 352–367. doi:10.1007/s00778-003-0103-3.

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

Freedman, David, Robert Pisani, e Roger Purves. 2007. *Statistics, 4th Edition*. 4th ed. W. W. Norton.

Freire, Juliana, David Koop, Emanuele Santos, e Cláudio T. Silva. 2008. "Provenance for Computational Tasks: A Survey". *Computing in Science and Engineering, v.10* (3): 11-21.

Gamma, Erich, Richard Helm, Ralph Johnson, e John M. Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

Gil, Yolanda, Ewa Deelman, Mark Ellisman, Thomas Fahringer, Geoffrey Fox, Dennis Gannon, Carole Goble, Miron Livny, Luc Moreau, e Jim Myers. 2007. "Examining the Challenges of Scientific Workflows". *Computer* 40 (12): 24-32. doi:10.1109/MC.2007.421.

Glatard, Tristan, Gergely Sipos, Johan Montagnat, Zoltan Farkas, e Peter Kacsuk. 2007. "Workflow-Level Parametric Study Support by MOTEUR and the P-GRADE Portal". In *Workflows for e-Science*, 279-299. Springer. http://dx.doi.org/10.1007/978-1-84628-757-2_18.

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

Goncalez, Thelma T., Ester C. Sabino, Ligia Capuani, Jing Liu, David J. Wright, Judy H. Walsh, Joao E. Ferreira, et al. 2011. "Blood transfusion utilization and recipient survival at Hospital das Clinicas in São Paulo, Brazil". *Transfusion*: no–no. doi:10.1111/j.1537-2995.2011.03387.x.

Google Developers. 2012. "Android Cloud to Device Messaging Framework - Android — Google Developers". https://developers.google.com/android/c2dm/.

Governato, F., C. Brook, L. Mayer, A. Brooks, G. Rhee, J. Wadsley, P. Jonsson, et al. 2010. "Bulgeless dwarf galaxies and dark matter cores from supernova-driven outflows". *Nature* 463 (7278) (janeiro 14): 203-206. doi:10.1038/nature08640.

Grimstead, I. J., N. J. Avis, e D. W. Walker. 2009. “RAVE: the resource-aware visualization environment”. *Concurrency and Computation: Practice and Experience* 21 (4) (março 25): 415-448. doi:10.1002/cpe.1327.

Grubbs, Frank E. 1969. “Procedures for detecting outlying observations in samples”.

Guerra, Gabriel, Fernando Rochinha, Renato Elias, Alvaro Coutinho, Vanessa Braganholo, Daniel de Oliveira, Eduardo Ogasawara, Fernando Chirigati, e Marta Mattoso. 2009. “Scientific Workflow Management System Applied to Uncertainty Quantification in Large Eddy Simulation”. In *Congresso Ibero Americano de Métodos Computacionais em Engenharia*, 1-13. Búzios, Rio de Janeiro, Brazil: CILAMCE.

Guerra, Gabriel, Fernando Rochinha, Renato Elias, Daniel Oliveira, Eduardo Ogasawara, Jonas Dias, Marta Mattoso, e Alvaro L.G.A. Coutinho. 2012. “Uncertainty Quantification in Computational Predictive Models for Fluid Dynamics Using Workflow Management Engine”. *International Journal for Uncertainty Quantification* 2 (1): 53-71.

Guerra, Gabriel. M, e Fernando A Rochinha. 2009a. “Uncertainty quantification in fluid-structure interaction via sparse grid stochastic collocation method”. In *30th Iberian-Latin-American Congress on Computational Methods in Engineering, 2009, Buzios*.

———. 2009b. “A Sparse Grid Method Applied to Stochastic Fluid-Structure Interaction”. In *COBEM, 2009, Gramado, Brazil*.

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

Hartman, Amber L, Sean Riddle, Timothy McPhillips, Bertram Ludäscher, e Jonathan A Eisen. 2010. “Introducing W.A.T.E.R.S.: a Workflow for the Alignment, Taxonomy, and Ecology of Ribosomal Sequences”. *BMC Bioinformatics* 11 (1): 317. doi:10.1186/1471-2105-11-317.

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

Hoffa, C., G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, e J. Good. 2008. “On the use of cloud computing for scientific workflows”. In *IEEE Fourth International Conference on eScience (eScience 2008), Indianapolis, USA*, 7–12.

Hull, Duncan, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R Pocock, Peter Li, e Tom Oinn. 2006. “Taverna: a tool for building and running workflows of services”. *Nucleic Acids Research* 34 (2): 729-732. doi:10.1093/nar/gkl320.

ISO/IEC/IEEE. 2009. “ISO/IEC/IEEE 9945:2009 Information technology - Portable Operating System Interface (POSIX®) Base Specifications, Issue 7”. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50516.

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

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

Kertész, Attila, Gergely Sipos, e Péter Kacsuk. 2007. "Brokering Multi-grid Workflows in the P-GRADE Portal". In *Euro-Par 2006: Parallel Processing*, org. Wolfgang Lehner, Norbert Meyer, Achim Streit, e Craig Stewart, 4375:138-149. Berlin, Heidelberg: Springer Berlin Heidelberg. http://www.springerlink.com/content/4717408063306472/.

Lemos, Melissa, Marco Antonio Casanova, Luiz Fernando Bessa Seibel, José Antonio Fernandes Macedo, e Antonio Basílio Miranda. 2004. "Ontology-Driven Workflow Management for Biosequence Processing Systems". In *Database and Expert Systems Applications*, org. Fernando Galindo, Makoto Takizawa, e Roland Traunmüller, 781-790. Berlin, Heidelberg: Springer Berlin Heidelberg. http://www.springerlink.com/content/rx24ttq8239ymcjr/.

Lins, Erb F, Renato N Elias, Gabriel M Guerra, Fernando A Rochinha, e Alvaro L. G. A Coutinho. 2009. "Edge-based finite element implementation of the residual-based variational multiscale method". *International Journal for Numerical Methods in Fluids* 61 (1): 1–22.

Martinho, W., E. Ogasawara, D. Oliveira, F. Chirigati, I. Santos, G.H.T. Travassos, e M. Mattoso. 2009. "A Conception Process for Abstract Workflows: An Example on Deep Water Oil Exploitation Domain". In *5th IEEE International Conference on e-Science*. Oxford, UK.

Matsunaga, Andréa, Maurício Tsugawa, e José Fortes. 2008. "CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications". *IEEE eScience 2008*: 229, 222.

Mattoso, Marta, Cláudia Werner, Guilherme Horta Travassos, Vanessa Braganholo, Leonardo Murta, Eduardo Ogasawara, Daniel Oliveira, Sergio Manuel Serra da Cruz, e Wallace Martinho. 2010. "Towards Supporting the Life Cycle of Large-scale Scientific Experiments". *International Journal of Business Process Integration and Management* 5 (1): 79–92.

Medeiros, C. B., J. Perez-Alcazar, L. Digiampietri, Jr G. Z. Pastorello, A. Santanche, R. S. Torres, E. Madeira, e E. Bacarin. 2005. "WOODSS and the Web: annotating and reusing scientific workflows". *SIGMOD Record* 34 (3): 18-23. doi:10.1145/1084805.1084810.

Moreau, Luc, Juliana Freire, Joe Futrelle, Robert McGrath, Jim Myers, e Patrick Paulson. 2008. "The Open Provenance Model: An Overview". In *Provenance and Annotation of Data and Processes*, 323-326. http://dx.doi.org/10.1007/978-3-540-89965-5_31.

Moreau, Luc, Bertram Ludäscher, Ilkay Altintas, Roger S. Barga, Shawn Bowers, Steven Callahan, Jr George Chin, et al. 2008. "Special Issue: The First Provenance Challenge". *Concurrency and Computation: Practice and Experience* 20 (5): 409-418.

OAuth. 2010. *OAuth Community Site*. http://oauth.net/.

Ocaña, Kary A. C. S., Daniel Oliveira, Jonas Dias, Eduardo Ogasawara, e Marta Mattoso. 2011. "Optimizing Phylogenetic Analysis Using SciHmm Cloud-based

Scientific Workflow". In *2011 IEEE Seventh International Conference on e-Science (e-Science)*, 190-197. Stockholm, Sweden: IEEE.

Ocaña, Kary A. C. S., Daniel Oliveira, Eduardo Ogasawara, Alberto M. R. Dávila, Alexandre A. B. Lima, e Marta Mattoso. 2011. "SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes". In *Advances in Bioinformatics and Computational Biology*, org. Osmar Norberto de Souza, Guilherme P. Telles, e Mathew Palakal, 6832:66-70. Berlin, Heidelberg: Springer Berlin Heidelberg. http://www.springerlink.com/index/10.1007/978-3-642-22825-4_9.

Ogasawara, Eduardo, Jonas Dias, Daniel Oliveira, Fabio Porto, Patrick Valduriez, e Marta Mattoso. 2011. "An Algebraic Approach for Data-Centric Scientific Workflows". *Proc. of VLDB Endowment* 4 (12): 1328-1339.

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

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

Oliveira, D., E. Ogasawara, K. Ocana, F. Baiao, e M. Mattoso. 2011. "An Adaptive Parallel Execution Strategy for Cloud-based Scientific Workflows". *Concurrency and Computation: Practice and Experience* (online).

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

Oliveira, Daniel, Eduardo Ogasawara, Fernanda Baião, e Marta Mattoso. 2010. "SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows". In *3rd International Conference on Cloud Computing*, 378–385. CLOUD '10. Washington, DC, USA: IEEE Computer Society. doi:10.1109/CLOUD.2010.64. http://dx.doi.org/10.1109/CLOUD.2010.64.

Patavino, Giuseppina Maria, Cesar de Almeida-Neto, Jing Liu, David J. Wright, Alfredo Mendrone-Junior, Maria Inês Lopes Ferreira, Anna Bárbara de Freitas Carneiro, et al. 2012. "Number of recent sexual partners among blood donors in Brazil: associations with donor demographics, donation characteristics, and infectious disease markers". *Transfusion* 52 (1): 151–159. doi:10.1111/j.1537-2995.2011.03248.x.

Pereira, G.C., e N.F.F. Ebecken. 2011. "Combining in situ flow cytometry and artificial neural networks for aquatic systems monitoring". *Expert Systems with Applications* 38 (8): 9626 - 9632. doi:10.1016/j.eswa.2011.01.140.

Pfleeger, Shari Lawrence. 2001. *Software Engineering: Theory and Practice*. 2º ed. Prentice Hall.

Porto, Fabio, Ana Maria Moura, Frederico C Silva, Adriana Bassini, Daniele C Palazzi, Maira Poltosi, Luis Eduardo Viveiros Castro, e L. C Cameron. 2011. "A

Metaphoric Trajectory Data Warehouse for Olympic Athlete Follow-up". *Concurrency and Computation: Practice and Experience*. doi:10.1002/cpe.1869. http://onlinelibrary.wiley.com/doi/10.1002/cpe.1869/abstract.

PostgreSQL. 2012. "PostgreSQL: Documentation: 8.3: Pattern Matching". http://www.postgresql.org/docs/8.3/static/functions-matching.html.

Sabino, Ester C., Thelma T. Gonçalez, Anna Bárbara Carneiro-Proietti, Moussa Sarr, João Eduardo Ferreira, Divaldo A. Sampaio, Nanci A. Salles, et al. 2011. "Human immunodeficiency virus prevalence, incidence, and residual risk of transmission by transfusions at Retrovirus Epidemiology Donor Study-II blood centers in Brazil". *Transfusion*: no–no. doi:10.1111/j.1537-2995.2011.03344.x.

Savarimuthu, Bastin Tony Roy, Maryam Purvis, e Martin Fleurke. 2004. "Monitoring and controlling of a multi-agent based workflow system". In *Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation - Volume 32*, 127–132. ACSW Frontiers '04. Darlinghurst, Australia, Australia: Australian Computer Society, Inc. http://dl.acm.org/citation.cfm?id=976440.976459.

Taylor, Ian J., Ewa Deelman, Dennis B. Gannon, e Matthew Shields. 2007. *Workflows for e-Science: Scientific Workflows for Grids*. 1º ed. Springer.

Taylor, Ian, Matthew Shields, Ian Wang, e Andrew Harrison. 2007. "The Triana Workflow Environment: Architecture and Applications". In *Workflows for e-Science*, 320-339. Springer. http://dx.doi.org/10.1007/978-1-84628-757-2_20.

Twitter. 2012a. "Rate Limiting | Twitter Developers". https://dev.twitter.com/docs/rate-limiting.

———. 2012b. "Central de Ajuda do Twitter | Os Limites do Twitter (Atualização, API, MD e Seguir)". https://support.twitter.com/articles/289867-os-limites-do-twitter-atualizacao-api-md-e-seguir#.

Twitter4j. 2011. "Twitter4J - A Java library for the Twitter API". http://twitter4j.org/en/index.html.

Valduriez, Patrick, e Esther Pacitti. 2005. "Data Management in Large-Scale P2P Systems". In *High Performance Computing for Computational Science - VECPAR 2004*, 104-118. http://dx.doi.org/10.1007/11403937_9.

Valerio, Matthew D., Satya S. Sahoo, Roger S. Barga, e Jared J. Jackson. 2008. "Capturing Workflow Event Data for Monitoring, Performance Analysis, and Management of Scientific Workflows". In , 626-633. IEEE. doi:10.1109/eScience.2008.164. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4736876.

Vaquero, Luis M., Luis Rodero-Merino, Juan Caceres, e Maik Lindner. 2009. "A break in the clouds: towards a cloud definition". *SIGCOMM Comput. Commun. Rev.* 39 (1): 50-55. doi:10.1145/1496091.1496100.

Zhao, Yong, M Hategan, B Clifford, I Foster, G von Laszewski, V Nefedova, I Raicu, T Stef-Praun, e M Wilde. 2007. "Swift: Fast, Reliable, Loosely Coupled Parallel Computation". In *3rd IEEE World Congress on Services*, 206, 199. Salt Lake City, USA. http://dx.doi.org/10.1109/SERVICES.2007.63.