Monitoramento em Tempo Real de *Workflows* Científicos Executados em Paralelo em Ambientes Distribuídos*

Julliano Pintas, Daniel de Oliveira, Kary Ocaña, Jonas Dias, Marta Mattoso

Programa de Engenharia de Sistemas e Computação – COPPE / UFRJ

{julliano, danielc, kary, jonasdias, marta}@cos.ufrj.br

Resumo. *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. Neste artigo, 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 através de tecnologias de dispositivos móveis e redes sociais. A avaliação da solução proposta, chamada SciLightning, foi realizada através 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 chamada SciCumulus. A avaliação mostrou que esta nova abordagem é eficaz no que tange o monitoramento e notificação de eventos, e pode ser facilmente acoplada a abordagens 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 de execução de forma online.*

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, 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 (Taylor *et al.* 2007), onde cada atividade corresponde à invocação de um programa 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 dados de entrada. Assim, é fundamental que estes *workflows* sejam executados utilizando técnicas de paralelismo em ambientes de Processamento de Alto Desempenho (PAD) para se obter os resultados em tempo hábil. 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 2010) e o SciCumulus (Oliveira *et al.* 2011), sendo este último focado em paralelismo de *workflows* em nuvens de computadores (Vaquero *et al.* 2009).

Como os *workflows* em larga escala tendem a executar por semanas ou até meses, mecanismos de monitoramento são fundamentais para que os cientistas mantenham-se informados 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 predeterminados. Este acompanhamento e análise parcial são passos iniciais para se

* Esse artigo foi parcialmente financiado pela CAPES, FAPERJ e CNPq.

prover mecanismos de condução do *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 o curso do *workflow* durante sua execução. Toda a avaliação realizada pelos cientistas é baseada nos dados de proveniência do *workflow* (Freire *et al.* 2008). Entretanto, este tipo de monitoramento ainda não é uma realidade nas abordagens existentes. Em sistemas como o Swift, ou implementações do MapReduce, não é possível consultar os dados de proveniência em tempo real (durante a execução) tornando os mecanismos de condução do *workflow* difícil de serem implementados. Outras abordagens como o SciCumulus já proveem os dados de proveniência em tempo real, entretanto não apresentam mecanismos de monitoramento mais avançados. No SciCumulus, o cientista necessita realizar consultas periódicas no repositório de proveniência através 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 e nem viável.

O objetivo deste artigo é apresentar o SciLightning, um arcabouço de monitoramento de execução de *workflows* científicos que pode ser acoplado às soluções para execução de *workflows* existentes. O SciLightning analisa os dados de proveniência gerados em tempo real e notifica o cientista sobre determinados eventos que lhe sejam importantes através de tecnologias de dispositivos móveis e redes sociais. Desta forma, o cientista mantem-se informado a respeito do estado da execução de seus *workflows*, e pode realizar a análise parcial dos resultados para, em seguida, executar ações de condução do *workflow*. Para analisar a solução proposta, o *workflow* SciPhy (Ocaña et al. 2011), de análise filogenética, foi executado em paralelo no ambiente de nuvem Amazon EC2 usando a máquina de execução SciCumulus (Oliveira *et al.* 2011). Além desta introdução, este artigo é organizado como segue. A Seção 2 apresenta a máquina de execução de *workflows* em nuvem chamada SciCumulus. A Seção 3 descreve a arquitetura conceitual da solução proposta, o funcionamento de seus componentes e apresenta o modelo de proveniência utilizado. A Seção 4 apresenta os resultados experimentais enquanto que a Seção 5 discute os trabalhos relacionados. Finalmente a Seção 6 conclui este artigo.

2. SciCumulus

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. 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 1): (i) **Camada Cliente**: os componentes desta camada despacham as atividades do *workflow* para execução em paralelo na nuvem; (ii) **Camada de Distribuição**: cria e gerencia a execução paralela de atividades em uma ou mais VM instanciadas em uma nuvem; (iii) **Camada de Execução**: seus componentes são instalados nas várias VM envolvidas na execução paralela das atividades. É responsável pela execução de programas e por coletar e armazenar dados de proveniência; e (iv) **Camada de Dados**: essa camada é responsável por armazenar dados de entrada e dados de proveniência consumidos e gerados pela execução paralela das atividades. 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 através de consultas SQL (realizadas pelo cientista) a base de proveniência. Mais informações sobre o SciCumulus são apresentadas por Oliveira *et al*. (2011).

3. SciLightning

3.1 Arquitetura do SciLightning

A arquitetura do SciLightning (Figura 1) é dividida em quatro componentes que são acoplados às soluções de execução de *workflows*: (i) **SciLightning *Cartridge***: componente que implementa um cartucho (Birsan 2005) para realizar a interface entre o SciLightning e a abordagem de execução; (ii) **SciLightning Monitor**: componente de monitoramento e análise de proveniência; (iii) **SciLightning Social**: componente de publicação das notificações nas redes sociais; e o (iv) **SciLightning *Mobile***: o componente para envio das notificações através de tecnologias móveis. O SciLightning é projetado para se conectar a qualquer abordagem de execução de *workflows* em qualquer tipo de ambiente computacional, como a nuvem, um *cluster* e até um *desktop*, desde que sejam fornecidos dados de proveniência que possam ser consultados em tempo real.

O SciLightning se conecta a cada abordagem de execução diferente implementando um cartucho que permite a tradução de um esquema de origem em uma interface padrão, uma visão (*i.e. database view*) com uma assinatura predefinida que retorna as notificações que devem ser enviadas. Esta visão é a única restrição na implementação do cartucho, que poderá envolver qualquer tipo de artefato de *software*, desde visões e procedimentos de banco de dados até programas externos. Este ponto de interface entre o SciLightning e a abordagem de execução foi planejado para oferecer máxima independência na implementação do cartucho. Como todas as regras de notificação se encontram implementadas no cartucho, qualquer alteração nestas regras pode ser realizada sem necessidade de adaptação dos outros 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. A relação de quais *workflows* serão monitorados, assim como a definição de parâmetros de monitoramento para cada *workflow* são definidos através de um portal *Web* e enviados ao SciLightning no início da execução do *workflow*. O cientista pode configurar uma série de regras de notificação no SciLightning através deste portal.

Com o apoio de cientistas, foram identificados cinco tipos de notificações: (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 *workflow* científico; (iii) **Marco atingido** - Um marco pode ser de dois tipos: de início de atividade e de término de atividade. Quando cada marco definido for atingido, é enviada uma notificação para o cientista especificando o marco e o horário que este foi atingido; (iv) **Arquivo gerado** - Podem ser definidas uma ou mais expressões regulares para monitoramento dos arquivos gerados. Para cada arquivo gerado pelo *workflow* científico monitorado que atenda a, pelo menos, uma das expressões regulares, é enviada uma notificação; (v) **Tarefa com duração atípica** - Consulta realizada para identificar execuções de atividades com durações atípicas. Primeiramente, a média e desvio padrão das durações de execução de cada tipo de atividade são calculados através do histórico de execuções

(consultando o banco de proveniência). Após isto, a distância entre a média e a duração de cada atividade em execução é calculada. As atividades com o tempo de execução fora da amplitude limite são consideradas *outliers*. 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 *et al.* 2007).

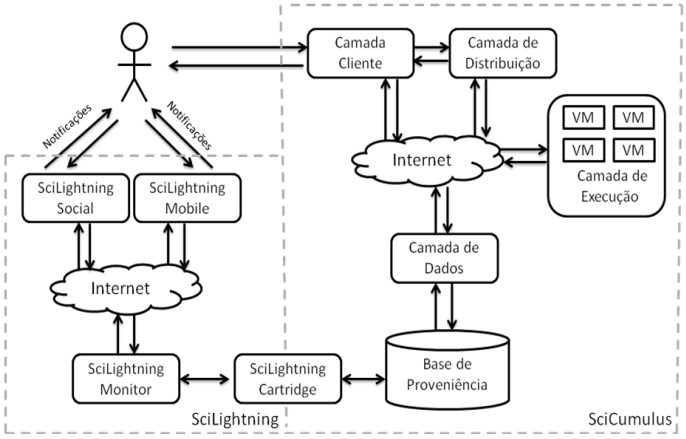

Figura 1 Arquitetura do SciLightning Acoplada ao SciCumulus

O SciLightning Monitor é uma componente desenvolvido em Java para *Web* versão 6.27 que tem como principal funcionalidade realizar o monitoramento da execução de *workflows* através de consultas periódicas à base de proveniência (utilizando o cartucho). A frequência de monitoramento, ou seja, o tempo entre as consultas é configurável, com valor padrão de 30 segundos. Além desta funcionalidade principal, o SciLightning Monitor permite o registro de dispositivos móveis e envia notificações para serem tratadas pelos componentes SciLightning Social e SciLightning *Mobile*. 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.

O SciLightning *Mobile* é uma aplicação *Android* que recebe, armazena e organiza as notificações derivadas do SciLightning Monitor. A plataforma *Android* foi a escolhida, dentre as outras opções de tecnologias para dispositivos móveis, pois é 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. Para registrar a aplicação SciLightning *Mobile*, o cientista necessita apenas de seu nome de usuário e senha. Porém, internamente, o registro ocorre em dois passos: (i) A aplicação 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 *Mobile*, que só então finalizará o registro após a validação das credenciais do cientista.

O SciLightning Social é um componente que faz a interface entre o SciLightning e as redes sociais. O SciLightning Social acessa a API das redes sociais para publicar resultados de monitoramento. Por exemplo, no caso do Twitter, o SciLightning Social

está associado a uma conta específica (@SciLightning) configurada para enviar notificações sobre a execução de *workflows* aos cientistas através do serviço de mensagens diretas (mantendo assim a privacidade). Para receber notificações, basta que o cientista siga este usuário. O cientista pode configurar tanto para receber notificações através do SciLightning *Mobile* e/ou SciLightning Social através do portal *Web*.

3.2 Modelo de Proveniência

Além de realizar o monitoramento e as notificações, o SciLightning também armazena os dados de proveniência relativos a esse monitoramento, como a definição de parâmetros de monitoramento, o registro da aplicação SciLightning Mobile e o controle de usuários e grupos. Assim, o SciLightning possui um modelo de proveniência próprio que deve ser acoplado ao modelo de dados da abordagem de execução de *workflows*. O modelo apresentado na Figura 2 traz o modelo de dados do SciLightning acoplado ao modelo de dados do SciCumulus (máquina de execução escolhida). Todas as entidades utilizadas no SciCumulus não sofreram alterações em sua estrutura. 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 2). As classes do SciCumulus são apresentadas sem os atributos para facilitar a visualização das classes do SciLightning.

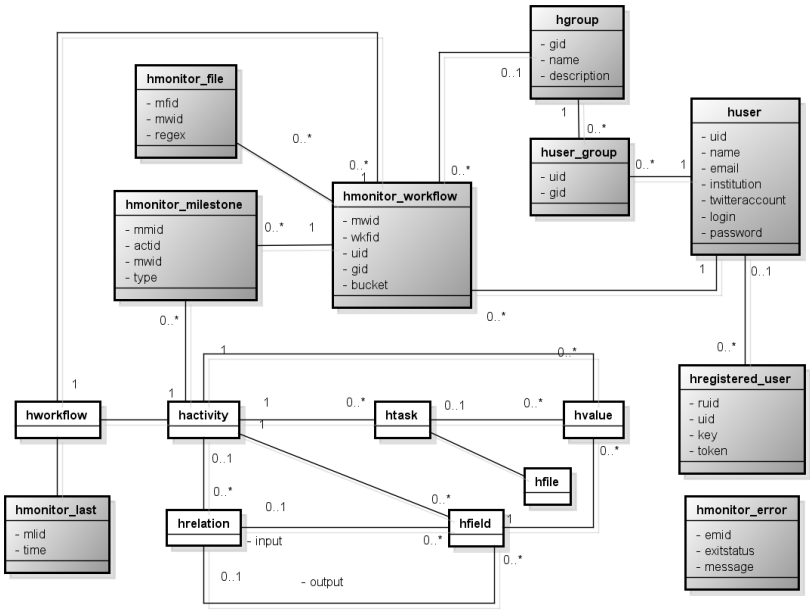

Figura 2 Modelo de Proveniência do SciLightning acoplado ao SciCumulus

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 de notificação, que configuram o envio de notificações quando o *workflow* gerar um determinado arquivo ou quando o *workflow* 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 serão 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 conta do Twitter utilizada para envio de notificações pelo SciLightning Social são armazenadas na tabela *huser*. Estes 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, que 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.

4. Avaliação Experimental

4.1 O *Workflow* SciPhy

Experimentos filogenéticos são exemplos de *workflows* que demandam PAD e que são compostos por milhares de execuções de atividades. Os *workflows* de análise filogenética 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 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 (Zvelebil e Baum 2007). 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 de 2 a 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 Resultados Experimentais

O SciLightning acoplado ao SciCumulus foi avaliado no ambiente Amazon EC2. No experimento apresentado neste artigo foram instanciadas 16 VM do tipo *large* (7,5 GB RAM, 850 GB de armazenamento, 2 núcleos virtuais). Cada VM instanciada é baseada no sistema operacional Cent OS 5 (64 bits). Neste experimento, o *workflow* SciPhy foi escolhido como estudo de caso. O experimento usa 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 é 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, conforme detalhado por Ocaña *et al.* (2011). Em cada execução é gerado um total de 5.000 tarefas paralelas. A execução do SciPhy 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 cujo nome respeite a expressão regular *^[A-Z]*.mafft*; e (v) Nas atividades com duração atípica. 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 1 apresenta os resultados obtidos.

Tabela 1 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	1	1	1	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 https://sites.google.com/site/scilightning/.

5. Trabalhos Relacionados

Existem alguns trabalhos na literatura que tratam da questão do monitoramento de *workflows* científicos executados em paralelo. Dentre os trabalhos relacionados está 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 nesse artigo, o MidMon monitora a execução através do sistema operacional, o que faz com que ele não seja capaz de identificar eventos como tempos de execução atípicos. O SciLightning não tem como objetivo monitorar aspectos referentes aos processos e a infraestrutura onde estes são executados, mas sim monitorar e acompanhar a execução do *workflow* científico através de seus resultados, notificando o cientista sobre eventos importantes através de dispositivos móveis e redes sociais. Balis *et al.* (2008) e Gil *et al.* (2007) apresentam uma série de necessidades de monitoramento e notificação para *workflows* científicos.

6. Considerações Finais

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 um *workflow* pode ser executado 100.000 vezes 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. Neste artigo, apresentamos uma abordagem para monitoramento e notificação de eventos na execução paralela de *workflows* científicos chamada SciLightning. Nossa abordagem utiliza a proveniência gerada em tempo de execução e eventos configurados pelo cientista. Utilizando a proveniência de tempo de execução, o SciLightning é capaz de identificar e notificar eventos durante a execução do *workflow* como erros em atividades, produção de arquivos e término do *workflow*.

Nossa avaliação experimental mostrou que o SciLightning é capaz de identificar e notificar 100% dos eventos. Com a exceção da notificação de execução atípica, todas as notificações tiveram precisão de 100%, o que mostrou que o SciLightning é 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 ou outras, contanto que, os dados de proveniência sejam fornecidos em tempo real. Notificar eventos do *workflows* em tempo de execução é requisito para possibilitar interferências na execução de experimentos como ajustes finos de parâmetros, e a adaptação do experimento durante sua execução (Dias *et al.* 2011). De outra forma, tal adaptabilidade pode ser difícil ou mesmo impossível de ser pré-programada.

Referências

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

Balis, B., Bubak, M., Łabno, B., (2008), "Monitoring of Grid scientific workflows", *Sci. Program.*, v. 16, n. 2-3 (abr.), p. 205–216.

Cruz, S. M. S. da, Silva, F. N. da, Gadelha Jr., L. M. R., Cavalcanti, M. C. R., Campos, M. L. M., Mattoso, M., (2008), "A Lightweight Middleware Monitor for Distributed Scientific Workflows". In: *CCGRID '08*, p. 693–698

Dean, J., Ghemawat, S., (2010), "MapReduce: a flexible data processing tool", *Commun. ACM*, v. 53 (jan.), p. 72–77.

Dias, J., Ogasawara, E., Oliveira, D., Porto, F., Coutinho, A., Mattoso, M., (2011), "Supporting Dynamic Parameter Sweep in Adaptive and User-Steered Workflow". In: *6th WORKS*, p. 31–36, Seattle, WA, USA.

Freedman, D., Pisani, R., Purves, R., (2007), *Statistics, 4th Edition*. 4th ed. W. W. Norton.

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

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., et al., (2007), "Examining the Challenges of Scientific Workflows", *Computer*, v. 40, n. 12, p. 24–32.

Ocaña, K. A. C. S., Oliveira, D., Ogasawara, E., Dávila, A. M. R., Lima, A. A. B., Mattoso, M., (2011), "SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes", *Advances in Bioinformatics and Computational Biology*, , chapter 6832, Berlin, Heidelberg: Springer, p. 66–70.

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

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

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

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

Zvelebil, Marketa, e Jeremy Baum. 2007. Understanding Bioinformatics. 1o ed. Garland Science.