ProvManager: Uma Abordagem para Gerenciamento de Proveniência de Workflows Científicos

Aluno: Anderson Souza Marinho[1*]
Orientadora: Cláudia Maria Lima Werner[1]
Co-orientador: Leonardo Gresta Paulino Murta[2]

[1] Programa de Engenharia de Sistemas e Computação (PESC)
COPPE/UFRJ, Universidade Federal do Rio de Janeiro
Rio de Janeiro, RJ, Brasil

[2] Instituto de Computação (IC)
Universidade Federal Fluminense
Niterói, RJ, Brasil

{andymarinho,werner}@cos.ufrj.br,leomurta@ic.uff.br

Nível: Mestrado
Ano de Ingresso: 2008
Aprovação da Proposta: Abril de 2009
Previsão de Conclusão: Junho de 2010

Resumo. *De maneira análoga a controle de execução de processos em ES, workflows científicos necessitam de recurso de rastreabilidade, conhecido como proveniência, para armazenar o histórico de execução de um experimento. Alguns workflows científicos, porém, estão passando a ser executados em ambientes distribuídos e, com isso, a definição de abordagens de proveniência independentes de sistemas de workflow vem se tornando necessária. Este tipo de abordagem é interessante, pois permite o armazenamento e o acesso aos dados de proveniência de maneira integrada, mesmo em experimentos que utilizam diferentes sistemas de gerenciamento de workflow. Com o intuito de prover funcionalidades de proveniência, as atuais abordagens sobrecarregam os cientistas com tarefas computacionais, tais como adaptações de script e implementações de funcionalidades extras. No entanto, esta estratégia não é uma boa solução, pois a maioria dos usuários (cientistas) que utilizam esses sistemas de workflow não possui habilidade computacional elevada. Assim, o objetivo deste trabalho é definir uma abordagem de proveniência que facilite a captura e a análise de dados de proveniência de maneira geral, principalmente em cenários de ambientes distribuídos.*

Palavras-chave*: Proveniência, workflow científico, experimento, ambiente distribuído.*

[*] Com auxílio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

1. Introdução

Experimentação científica é uma das formas usadas pelos cientistas para investigar fenômenos, tendo como objetivo adquirir novos conhecimentos ou corrigir e integrar conhecimentos previamente estabelecidos (Wilson 1991). Nas últimas décadas, com o avanço da computação, os experimentos científicos passaram a utilizar ferramentas computacionais para facilitar a sua execução. A partir dessa disponibilidade de recursos computacionais, ambientes, objetos e participantes passaram a ser simulados, surgindo a categoria de experimentos *in silico* (Travassos e Barros 2003). Entretanto, os experimentos passaram a possuir uma grande quantidade de programas e a sua manipulação tornou-se mais complexa (Mattoso et al. 2008). O conceito de *workflow* passa a ser utilizado com o objetivo de facilitar a abstração de um experimento, permitindo uma composição estruturada de programas como uma seqüência de atividades que visa um determinado resultado (Hollingsworth 1995). Junto a isso, com o propósito de automatizar a construção e execução dos chamados *workflows* científicos, foram criados sistemas gerenciadores de *workflows* científicos (*SGWfC*). Um *SGWfC* possibilita o cientista realizar uma espécie de programação em alto nível, através do encadeamento de programas (ou atividades) que seguem um determinado fluxo e lógica. Com isso, como em controle de processos em ES, *SGWfC* necessita disponibilizar recursos de rastreabilidade que vinculem programas e dados produzidos, com o intuito de possibilitar uma validação posterior. Esse recurso em *workflow* científico é conhecido como **proveniência** (Freire et al. 2008).

Proveniência fornece informação histórica acerca dos dados manipulados em um *workflow* (Freire et al. 2008). Essa informação histórica descreve os dados que foram gerados, apresentando os seus processos de transformação a partir de dados primários e intermediários. O controle deste tipo de informação é extremamente importante, uma vez que proporciona aos cientistas uma variedade de oportunidades de trabalho. Por exemplo, a partir de proveniência, é possível garantir a qualidade dos dados gerados, já que é possível observar os dados que foram utilizados (seus ancestrais) e determinar se estes são confiáveis ou não. Outros exemplos são a possibilidade de realização de auditorias para verificar quais recursos estão sendo realmente utilizados, a repetição da derivação dos dados sem a necessidade da reexecução completa do *workflow*, a atribuição de responsabilidades dos dados gerados, entre outras aplicações (Simmhan et al. 2005).

Para se obter benefícios das informações de proveniência, é necessário que estas sejam capturadas, modeladas e armazenadas para posterior consulta. O gerenciamento de proveniência é uma questão em aberto que já foi, de certa forma, abordada por vários *SGWfC* e diversos trabalhos na literatura (Simmhan et al. 2005). Um dos problemas ainda em aberto é a falta de concordância do que deve ser capturado para proveniência, além de uma definição clara de como essa captura deve ser feita. A captura de dados de proveniência se torna mais complexa quando o *workflow* é executado em diferentes ambientes de execução, ou seja, em um ambiente distribuído. Neste cenário, as informações de proveniência devem ser coletadas e armazenadas a partir de fontes distintas. Em cenários mais críticos, um experimento é fragmentado em diversos *workflows* menores para serem executados em *SGWfC* distintos.

A partir do cenário apresentado e dos problemas destacados, o objetivo deste trabalho é definir uma abordagem, chamada *ProvManager*, que viabilize o gerenciamento

de proveniência de *workflows*. Em especial, *workflows* que são executados em ambientes distribuídos. O objetivo é fazer com que o gerenciamento de proveniência suba mais um nível, passando do gerenciamento no nível de *workflow* para o gerenciamento de proveniência no nível de experimento.

2. Abordagem Proposta

Visando a aplicação de um sistema de proveniência em um ambiente distribuído, a abordagem *ProvManager* deve ser independente de qualquer tecnologia que dê suporte a execução de *workflows*, ou seja, independente de qualquer *SGWfC*, escalonador de atividades, máquinas de execução, etc. Esta independência pode ser garantida através da utilização de um mecanismo de captura de proveniência adequado. Mecanismos de captura que trabalham acoplados ao *SGWfC*, por exemplo, não são uma boa escolha, pois são dependentes do *SGWfC* e não podem ser reutilizados em diferentes sistemas (Freire et al. 2008). Portanto, a abordagem *ProvManager* adotou um mecanismo de captura que trabalha no nível de atividade (Marinho et al. 2009). Neste nível, cada atividade do *workflow* é responsável por capturar suas próprias informações de proveniência e, com isso, a independência de *SGWfC* é garantida. Entretanto, o problema de trabalhar neste nível é que as atividades precisam ser adaptadas para darem suporte ao mecanismo de captura. Nas abordagens existentes na literatura, essa adaptação é deixada a cargo do cientista, porém, geralmente, a maioria dos cientistas não possui habilidades computacionais suficientes para tal. Para contornar essa deficiência, *ProvManager* propõe um mecanismo automático de configuração de *workflows*.

A Figura 1 apresenta uma visão geral do funcionamento do *ProvManager*. O primeiro passo que o cientista deve realizar é configurar o *workflow* para permitir a coleta de proveniência. Essa etapa é delegada para o *ProvManager* através do seu mecanismo de configuração. Esse mecanismo recebe a especificação do *workflow* e configura cada atividade adicionando o mecanismo de captura de proveniência. Durante essa configuração, informações de proveniência relacionadas ao tempo de desenvolvimento do *workflow*, conhecidas como proveniência prospectiva (Freire et al. 2008), são capturadas e coletadas no *ProvManager*. Após a configuração, as informações de proveniência relacionadas ao tempo de execução do *workflow*, conhecidas como proveniência retrospectiva (Freire et al. 2008), são transferidas de cada atividade do *workflow* para o repositório central de proveniência via uma API de serviços web. Após o término da execução do experimento, o cientista possui um local integrado para visualizar e analisar os dados de proveniência.

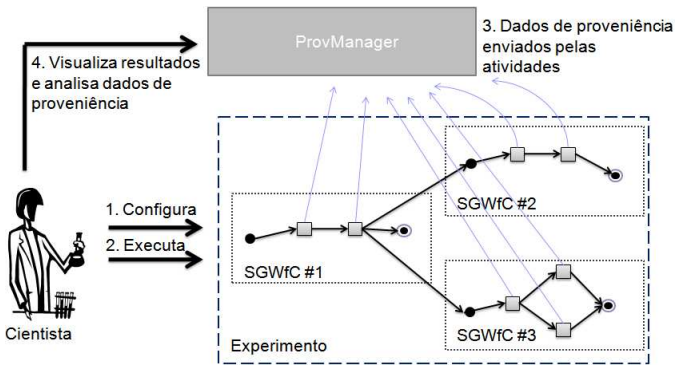

Figura 1 - Visão geral do funcionamento da abordagem

Analisando esse processo, algumas questões merecem ser tratadas com mais atenção. A primeira delas é sobre a especificação do *workflow*, que pode ser escrita de maneira diferente de acordo com o *SGWfC* utilizado. Além disso, cada *SGWfC* possui características próprias como, por exemplo, suporte a determinados tipos de atividades, formas de execuções distintas, etc. Por esse motivo, a arquitetura do *ProvManager* está sendo definida para ser flexível e extensível, a fim de dar suporte as características específicas de diferentes *SGWfC*. O *ProvManager* irá dar suporte a inserção de cartuchos (*plugins*) que serão responsáveis pela configuração de *SGWfC* específicos, tais como VisTrails, Kepler, Taverna, dentre outros.

Outra questão complexa é a adaptação das atividades para viabilizar o mecanismo de captura de proveniência. Mais uma vez, a resposta depende do *SGWfC*. Alguns sistemas escrevem o código da atividade diretamente na especificação do *workflow*, outros trabalham apenas com serviços web ou executáveis. Em piores casos, algumas atividades não disponibilizam código fonte ou são executadas remotamente, não podendo, assim, ser alteradas. Com isso, uma solução é realizar essa adaptação de maneira indireta através da construção de *wrappers* que encapsulem as atividades originais e acrescentem o mecanismo de captura de proveniência, com é ilustrado na Figura 2.a. Esses *wrappers* podem ser implementados a partir da utilização do recurso de atividade composta, fornecido pela maioria dos *SGWfC*. Uma atividade composta agrega um conjunto de atividades, formando um *sub-workflow*. Assim, para cada atividade do *workflow*, uma atividade composta é criada, contendo a própria atividade original, além da inclusão das atividades de coleta de proveniência (ACP), que capturam os dados de proveniência da atividade original e os publicam via serviços web no *ProvManager*. Vale destacar que os cartuchos de *SGWfC* inseridos no *ProvManager* são responsáveis por gerar essas atividades compostas em um *SGWfC* específico.

A Figura 2.b ilustra um exemplo de adaptação de um *workflow* escrito em Kepler utilizando a solução de atividades compostas. Do lado esquerdo, encontra-se o *workflow* original, formado por três atividades, que serão substituídas por atividades compostas. A adaptação de uma dessas atividades (atividade A) é ilustrada no *workflow* do lado direito da Figura 2.b. É possível notar que algumas ACP são configuradas para serem executadas antes da atividade A, enquanto que outras são configuradas para serem executadas após. Essa característica depende do tipo de proveniência que a ACP publica. Uma ACP que publica o início da execução de uma atividade, por exemplo, precisa ser executada antes da atividade original no *sub-workflow*. Por outro lado, uma ACP que publica o término de execução de uma atividade precisa ser executada após a atividade original. É possível observar também que duas ou mais ACP podem ser executadas em paralelo, otimizando, assim, a execução do *workflow*.

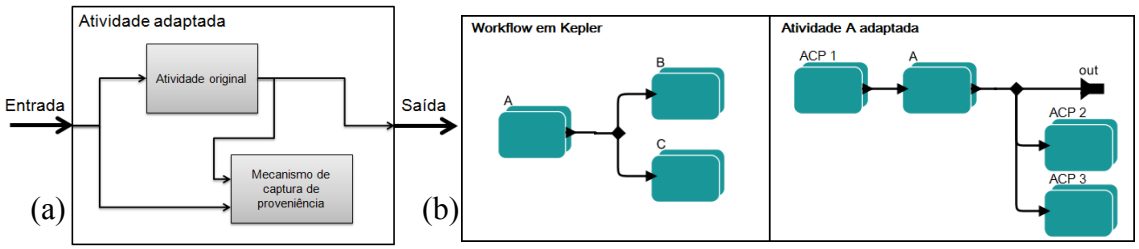

Figura 2 - Estrutura *wrapper* (a) e *workflow* Kepler adaptado (b)

3. Trabalhos Relacionados

Na literatura, alguns trabalhos (Groth et al. 2006, Lin et al. 2008, Simmhan et al. 2006) possuem as mesmas preocupações que este trabalho e defendem uma abordagem de proveniência que seja independente de *SGWfC*. Além disso, existe uma iniciativa de Moreau *et al.* (2007), que estabelece um modelo de proveniência padrão, chamado *Open Provenance Model* (OPM), que define uma representação genérica de proveniência. Entretanto, estes trabalhos não demonstram ter uma real preocupação com o cientista no que se diz respeito à facilidade de aplicação como a abordagem *ProvManager*. Por exemplo, estes trabalhos não procuram diminuir os esforços do cientista na etapa de configuração do mecanismo de proveniência. Além disso, esses trabalhos não apresentam a preocupação do gerenciamento de proveniência no nível de experimento.

Por exemplo, Simmhan *et al.* (2006) desenvolveram um *framework* para captura de proveniência em *workflows* científicos. Este *framework* tem como principal característica a independência de *SGWfC*, porém foi concebido apenas para *workflows* em grade (Grid) e serviços web. O seu mecanismo de captura funciona no nível de atividade e as informações coletadas por cada serviço do *workflow* são enviadas para um repositório central através de serviços web. O grande problema deste *framework* é o encargo imposto ao cientista de implementar manualmente o mecanismo de captura e de publicação de proveniência em cada atividade pertencente ao *workflow*.

Já o trabalho de Groth *et al.* (2006) foi um dos primeiros trabalhos a se preocupar com a definição de um mecanismo de proveniência que fosse independente de *SGWfC*. Nesse trabalho, uma arquitetura de proveniência independente de *SGWfC* foi definida. Esta arquitetura possui o seu próprio modelo de proveniência, que define informações de proveniência exclusivas de cada atividade do *workflow*, como por exemplo, o fluxo interno de dados e os estados de uma atividade. Devido a essa particularidade, a arquitetura estabeleceu que o mecanismo de captura deve ser aplicado no nível de atividade. No entanto, a arquitetura não define nenhum mecanismo que auxilie o cientista na adaptação das atividades do *workflow*. O cientista é obrigado a fazer essa adaptação manualmente. Tendo em vista esse problema, foi definida uma metodologia, denominada PrIMe (Munroe et al. 2006), que tem como objetivo facilitar o desenvolvimento de aplicações que possuam funcionalidades de proveniência. Embora seja uma boa iniciativa, o problema de certo modo ainda persiste, já que o cientista precisa adaptar as atividades manualmente. Prime é apenas uma metodologia.

4. Considerações Finais

Este artigo apresentou *ProvManager*, uma proposta de abordagem de gerenciamento de proveniência que é independente de tecnologia de *SGWfC*. *ProvManager* está sendo concebido principalmente para permitir um gerenciamento de proveniência de *workflows* que são executados em ambientes distribuídos, onde às vezes mais de um *SGWfC* é utilizado para executar um único experimento. Além disso, *ProvManager* visa facilitar o trabalho do cientista durante o uso. Um exemplo é o mecanismo proposto que ajuda o cientista a configurar automaticamente o mecanismo de captura de proveniência no *workflow*. Vale destacar que essa abordagem pode ser generalizada e aplicada em outros domínios, além do de experimentação científica. No entanto, isso está fora do escopo do trabalho.

Este trabalho está dividido em quatros macro atividades. A primeira delas, parcialmente concluída, é a revisão bibliográfica. A segunda etapa, em fase de desenvolvimento, consiste na definição do modelo de dados de proveniência. Este modelo será definido usando o OPM como base, porém outras informações de proveniência serão acrescentadas, porque este último ainda não é um modelo de proveniência completo. A terceira etapa consiste na implementação da abordagem. No momento, estamos definindo a arquitetura e decidindo questões importantes como, por exemplo, a utilização de uma base de conhecimento Prolog ou OWL/RDF para o armazenamento dos dados de proveniência. A última etapa consiste na avaliação da abordagem. Essa avaliação será desmembrada em diversos aspectos. O primeiro deles será uma avaliação sobre o modelo de proveniência definido, verificando se este abrange todas ou as mais importantes informações de proveniência em um experimento. Outra avaliação é sobre a questão de desempenho, analisando quanto de sobrecarga é acrescentada na execução do *workflow* com a inserção das ACP, uma vez que existem *workflows* mais complexos que demoram semanas para serem executados e lidam com gigabytes de dados. Uma última avaliação seria uma avaliação mais geral, verificando se realmente a abordagem auxilia o cientista no gerenciamento dos dados de proveniência no nível de experimento. Essa avaliação pode ser feita a partir da comparação de outros sistemas de proveniência existentes. Por fim, workflows reais do domínio de Engenharia de petróleo serão utilizados para avaliar a abordagem.

Referências

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

Groth, P., Jiang, S., Miles, S., Munroe, S., et al., 2006, *An Architecture for Provenance Systems*, Technical report, Electronics and Computer Science, University of Southampton.

Hollingsworth, D., 1995, "Workflow Management Coalition: The Workflow Reference Model", *The Workflow Management Coalition*

Lin, C., Lu, S., Lai, Z., Chebotko, A., Fei, X., Hua, J., Fotouhi, F., 2008, "Service-Oriented Architecture for VIEW: A Visual Scientific Workflow Management System". In: *SCC '08*, p. 335-342

Marinho, A., Murta, L., Werner, C., Braganholo, V., Da Cruz, S., Mattoso, M., 2009, "A Strategy for Provenance Gathering in Distributed Scientific Workflows". In: *IEEE 2009 Third International Workshop on Scientific Workflows (SWF 2009)* , L.A., USA.

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

Moreau, L., Freire, J., Myers, J., Futrelle, J., Paulson, P., 2007, *The Open Provenance Model*, Technical report, Electronics and Computer Science, University of Southampton.

Munroe, S., Miles, S., Moreau, L., Vázquez-Salceda, J., 2006, "PrIMe: a software engineering methodology for developing provenance-aware applications". In: *Proceedings of the 6th international workshop on Software engineering and middleware*, p. 39-46, Portland, USA.

Simmhan, Y. L., Plale, B., Gannon, D., 2005, *A Survey of Data Provenance Techniques*, Technical report, Computer Science Department, Indiana University.

Simmhan, Y. L., Plale, B., Gannon, D., 2006, "A Framework for Collecting Provenance in Data-Centric Scientific Workflows". In: *ICWS*, p. 427-436, Chicago, USA.

Travassos, G., Barros, M., 2003, "Contributions of in virtuo and in silico experiments for the future of empirical studies in software engineering". In: *Workshop on Empirical Software Engineering: The Future of Empirical Studies in Software Engineering*, p. 117-130, Fraunhofer IRB Verlag, Roman Castles, Italy.

Wilson, E. B., 1991, *An Introduction to Scientific Research*. Dover Publications.