paper:77

Captura e Consulta a Dados de Proveniência Retrospectiva Implícita Intra-Atividade

Wellington Oliveira[1,2], Vitor C. Neves[1], Kary Ocaña[3], Leonardo Murta[1], Daniel de Oliveira[1], Vanessa Braganholo[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brasil

[2]Departamento Acadêmico de Ciência da Computação – Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais – Rio Pomba – MG – Brasil

[3]COPPE - Universidade Federal do Rio de Janeiro (UFRJ)

{wellmor, vcneves, leomurta, danielcmo, vanessa}@ic.uff.br;
kary@cos.ufrj.br

Abstract. *Executing scientific experiments in large scale generates huge amounts of data that should be captured for future query. However, most of the available tools collect only part of the provenance (i.e., only that which is explicitly defined in the workflow specification), leading to loss of important information to the analysis of the experiment. This work presents an approach for collecting, storing, and querying intra-activity provenance obtained by monitoring the directories and files manipulated by activities of the workflow. Therefore, information not referenced in the workflow specification are captured and related with previously specified provenance.*

Resumo. *A execução de experimentos científicos em larga escala gera uma grande quantidade de dados que devem ser capturados para consulta futura. Porém, a maioria das ferramentas disponíveis coletam apenas parte da proveniência (i.e., aquela que é explicitada na especificação do workflow), levando à perda de informações importantes para a análise do experimento. Este trabalho apresenta uma abordagem para coleta, armazenamento e consulta da proveniência intra-atividade obtida pelo monitoramento dos diretórios e arquivos manipulados pelas atividades do workflow. Assim, informações não referenciadas na especificação do workflow são capturadas e relacionadas com proveniência previamente especificada.*

1. Introdução

A execução de experimentos científicos modelados como *workflows* produz resultados que necessitam de validação e que sejam passíveis de reprodução por terceiros [Freire *et al.* 2008]. Para tal, é necessário consultar o histórico das transformações dos dados desde sua origem até os resultados gerados na execução de um *workflow* científico. Esse histórico é conhecido como proveniência [Freire *et al.* 2008]. A proveniência pode ser classificada como prospectiva, quando relacionada à especificação do *workflow*, ou retrospectiva, quando associada à execução do *workflow* [Freire *et al.* 2008]. Podemos ainda subdividir a proveniência retrospectiva em explícita, que diz respeito aos elementos referenciados na configuração do *workflow*, e implícita, que trata de elementos não referenciados na definição do *workflow* [Marinho *et al.* 2011]. Por

exemplo, se uma atividade do *workflow* invoca um *script* externo que recebe como parâmetro um arquivo f_1, mas também modifica os arquivos f_2 e f_3, Sistemas de Gerência de *Workflows* Científicos (SGWfC), tais como SciCumulus [Oliveira *et al.* 2010], Vistrails [Callahan *et al.* 2006] e Kepler [Ludäscher *et al.* 2006], somente capturariam a alteração do arquivo f_1. Todo o histórico da modificação dos arquivos f_2 e f_3 será perdido. Neste caso temos um fluxo de dados implícito (*i.e.* proveniência retrospectiva implícita) em relação a f_2 e f_3.

Tomemos como exemplo concreto o *workflow* SciPhy da área de bioinformática, apresentado na Figura 1. O SciPhy possui um fluxo de dados implícito gerado durante sua execução. O SciPhy recebe como entrada sequências de DNA, RNA ou aminoácidos e gera uma árvore filogenética, que representa a semelhança entre organismos [Ocaña *et al.* 2011]. Conforme pode ser observado na Figura 1, os arquivos *MFS*, *MAS*, *PH*, *Modelo*, *EM*, e *Árvore Filogenética* são acessados e/ou modificados por uma ou mais atividades. Intuitivamente, uma primeira estratégia de captura da proveniência seria verificar a transformação de cada arquivo no final da execução da atividade que o manipula. Essa estratégia é interessante do ponto de vista prático, mas pode levar à perda de informações importantes. Por exemplo, conforme requisitos pré-determinados pelo cientista, a atividade *Alinhamento de Sequência* poderia organizar as sequências no formato *Fasta* (arquivo *MFS*) por tamanho e/ou eliminar algumas delas (*i.e.*, aquelas repetidas ou muito similares) antes de alinha-las (na atividade *Alinhamento de Sequência*). Nesse caso, o SGWfC não estaria ciente dessa modificação. Outro exemplo que pode ser observado é na leitura e manipulação do arquivo *EM* pela atividade *Busca do Modelo Evolutivo*. Nesta atividade podem ocorrer duas operações sobre o mesmo arquivo: uma operação de limpeza sobre os dados do arquivo e outra operação de gravação do modelo evolutivo extraído do arquivo *Modelo*. Assim, em uma estratégia de coleta de dados de proveniência intra-atividade, poderíamos coletar não somente a última transformação do arquivo (*i.e.*, última versão do arquivo produzida pela atividade), mas também todo o fluxo implícito (*i.e.*, sequência de modificações) ocorrido durante a execução de cada atividade.

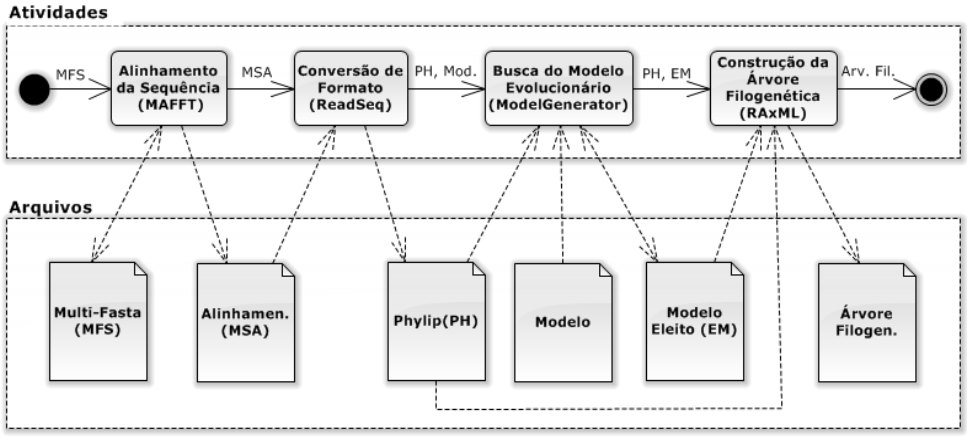

Figura 1. Atividades e arquivos de uma execução do *workflow* SciPhy

Esta captura da proveniência intra-atividade permite coletar e analisar a proveniência gerada sob as transformações de todos os arquivos consumidos e produzidos por cada atividade em um determinado diretório raiz, em seus subdiretórios ou em um espaço de trabalho gerenciado (*workspace*). Este registro em um nível mais detalhado possibilita entender melhor o funcionamento interno da atividade (mesmo

sem conhecer seu código fonte) e verificar a compatibilidade do processamento dos dados com os objetivos da atividade. Após a análise dos dados, é possível corrigir eventuais erros na configuração da atividade, definindo melhor seus dados de entrada ou a ordem de execução de cada operação de criação, modificação ou exclusão.

Para capturar a proveniência, algumas abordagens têm utilizado mecanismos de Gerência de Configuração de *Software* (GCS) [Koop *et al.* 2010; Neves *et al.* 2013]. A Gerência de Configuração é a área responsável pela gerência da evolução do *software* [Estublier 2000]. Estas abordagens atuam no máximo na coleta da proveniência obtida sob a perspectiva da atividade (unidade de versionamento). Entretanto, não permitem capturar todas as transformações dos artefatos manipulados durante a execução de cada atividade, conforme ocorre na execução do SciPhy (Figura 1). Apenas o resultado final da execução de uma atividade é capturado.

Outras abordagens [Frew *et al.* 2008; Holland *et al.* 2008] trabalham sobre os dados de proveniência em um grão mais fino que a atividade do *workflow*, coletando a proveniência a partir dos processos de sistema operacional que manipulam arquivos e através do próprio sistema de arquivos. Entretanto, a proveniência coletada fica desconectada dos dados de proveniência prospectivos (representação conceitual do *workflow*). Associar esses dados à proveniência prospectiva pode não ser uma tarefa trivial. Além disso, esse tipo de trabalho pode ser computacionalmente intenso, uma vez que o sistema operacional trabalha sobre múltiplos processos de forma simultânea.

Dessa forma, esse artigo apresenta uma abordagem de captura da proveniência retrospectiva intra-atividade gerada pelo fluxo de dados implícitos ou explícitos durante toda a execução de cada atividade, e não apenas ao final de cada atividade. A solução proposta, denominada ProvWatcher, relaciona a proveniência capturada com a proveniência prospectiva e retrospectiva sob a gerência dos SGWfCs. Para tal, ProvWatcher monitora diretórios e arquivos manipulados pelas atividades do workflow e registra as modificações em arquivos em um banco de dados relacional em tempo real, o que possibilita consultá-las e analisá-las durante o curso de execução do *workflow*. A abordagem proposta nesse artigo captura a proveniência implícita de forma análoga a proposta por Neves *et al.* [2013], entretanto captura as edições parciais dos arquivos feitos por cada atividade do *workflow*. Dessa forma, pode ser obtido um histórico mais detalhado sobre a edição de arquivos durante a execução de cada atividade.

O artigo está estruturado como segue. A Seção 2 oferece um referencial teórico sobre os conceitos relativos à proveniência de dados científicos. Na Seção 3 é apresentada a abordagem para a captura de proveniência sobre diretórios e arquivos. Um estudo de caso com o uso da abordagem é apresentado na Seção 4. A Seção 5 discute os trabalhos relacionados. Por fim, a Seção 6 conclui o artigo, apontando possibilidades de trabalhos futuros.

2. Proveniência de Dados Científicos

O termo "proveniência de dados científicos" pode ser definido como a origem ou linhagem do dado que ajuda a compreender o resultado de um experimento científico [Freire *et al.* 2008]. Com esse entendimento é possível obter resultados mais confiáveis e também garantir a reprodutibilidade do experimento. De acordo com Freire *et al.* [2008], a proveniência obtida de *workflows* pode ser classificada como: (i) *Proveniência Prospectiva*, que é aquela que captura a especificação de tarefas computacionais, e

corresponde aos passos a serem seguidos para chegar a um resultado, e (ii) *Proveniência Retrospectiva*, que é aquela que captura os passos executados e as informações sobre o ambiente utilizado para derivar um resultado, consistindo em um histórico estruturado e detalhado de uma execução de tarefas computacionais.

Freire *et al.* [2008] classifica também os mecanismos de captura de proveniência em três tipos: (i) Baseados em *workflow*, que monitoram o *workflow* como um todo, são construídos para SGWfC e dependentes deles (fortemente acoplados aos mesmos); (ii) Baseados em atividades, em que as atividades coletam sua própria proveniência. Esses mecanismos podem ser independentes de SGWfC, mas exigem adaptação das atividades do *workflow*. Por fim, os mecanismos de captura também podem ser (iii) Baseados em chamadas ao Sistema Operacional, que capturam somente a proveniência retrospectiva em grão fino, mas em geral não capturam a proveniência prospectiva, ou, quando o fazem, não relacionam proveniência prospectiva e retrospectiva. Devido à grande quantidade de informação gerada, o processo de captura e consulta pode ser complexo.

A proveniência retrospectiva também pode ser subdividida em explícita e implícita [Marinho *et al.* 2011]. A proveniência retrospectiva explícita refere-se às informações que foram devidamente explicitadas na especificação do *workflow*. Por outro lado, a proveniência retrospectiva implícita refere-se aos fluxos de dados implícitos, ou seja, os fluxos que não foram explicitados na especificação do *workflow*. Este tratamento de proveniência implícita permite obter informações como a identificação da atividade que criou ou modificou um determinado arquivo, a data em que o arquivo foi alterado, etc. A Figura 1 ilustra a ocorrência destes dois tipos de fluxos de dados: um fluxo de dados e processos que ocorre no domínio do SGWfC e é coletado por ele (proveniência retrospectiva explícita) e outro que ocorre no domínio do SO e que normalmente não é coletado pelo SGWfC (proveniência retrospectiva implícita). Assim, na especificação do SciPhy é definido que a atividade *Conversão de Formato* será executada logo após a atividade *Alinhamento da Sequência*, mas não há informação sobre o compartilhamento das informações contidas nos arquivos nem sobre os processos de edição dos arquivos que ocorrem durante a execução de cada uma delas.

3. ProvWatcher: Captura da Proveniência Intra-Atividade

A análise do processo de transformação dos dados gerados e consumidos por uma atividade é de fundamental importância para sua manutenção e evolução. Dado que uma atividade pode manipular um ou vários arquivos e cada um destes arquivos pode ser modificado mais de uma vez durante a execução do *workflow*, faz-se necessário realizar o monitoramento dos diretórios onde estes dados se encontram. Assim, para monitorar e recuperar estas informações, foi desenvolvida a abordagem chamada ProvWatcher. Nessa abordagem o diretório onde um determinado *workflow* é executado é monitorado em tempo real durante a execução de cada uma de suas atividades. A cada novo evento de criação, alteração ou exclusão de arquivos, os dados de proveniência são capturados, independente se são referenciados na especificação do workflow ou não. Esses dados (nome do arquivo, diretório, tipo de modificação e data) são armazenados no repositório de proveniência e relacionados à atividade que está sendo executada naquele momento.

A Figura 2 exibe uma visão geral do ProvWatcher. Nessa visão geral, o *SGWfC* gerencia o *workflow* científico e realiza a coleta da proveniência prospectiva e retrospectiva explícita. O *Repositório de Proveniência* armazena a proveniência

prospectiva e retrospectiva capturada pelo *SGWfC* e a proveniência retrospectiva implícita capturada pelo componente *ProvWatcher*. As setas pontilhadas indicam a comunicação entre os componentes da arquitetura em diferentes domínios (*Domínio do Workflow* e *Domínio do Sistema de Arquivos*), enquanto as setas cheias indicam a comunicação entre os componentes da arquitetura dentro do mesmo domínio. O *Diretório Raiz* é onde se encontram os arquivos manipulados pelos programas mapeados na especificação do *workflow*. Os *Arquivos 1 ... N* representam os arquivos de diversos formatos manipulados pelos programas definidos pelo *SGWfC*.

O *ProvWatcher* é responsável pelo monitoramento do *Diretório Raiz*, pela coleta da proveniência retrospectiva implícita e pela associação da proveniência implícita com a proveniência capturada pelo *SGWfC*. Finalmente, o *ProvDB* realiza a leitura e escrita de dados no *Repositório de Proveniência*. Apesar do *ProvWatcher* ser independente do *SGWfC*, seu componente *ProvDB* é desenvolvido conforme o modelo do repositório de dados do *SGWfC* em questão. O *Repositório de Proveniência* utilizado na arquitetura pode adotar ou não padrões como o PROV [Moreau and Missier 2013] ou OPM [Moreau *et al.* 2011]. Caso seja necessário realizar a conversão de dados de proveniência para o padrão PROV, ferramentas de intercâmbio de proveniência entre SGWfC heterogêneos podem ser utilizadas [Oliveira *et al.* 2014].

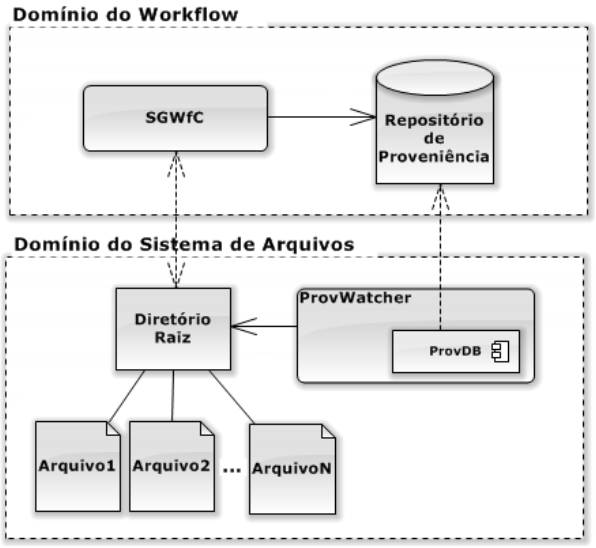

Figura 2. Arquitetura da abordagem ProvWatcher

O *ProvWatcher* e o *ProvDB* foram desenvolvidos na linguagem Java devido ao fato dela ser multiplataforma e possuir uma API própria para notificação de mudanças de arquivos denominada *WatchService* [Oracle 2014]. Essa API opera sobre a *Java Virtual Machine* (JVM) e fornece uma abstração do sistema de arquivos e do SO utilizado, facilitando a portabilidade da aplicação entre diferentes SO. Por outro lado, ela não é capaz de capturar acessos de leitura aos arquivos e diretórios. Durante o monitoramento iterativo dos diretórios do *workflow,* para cada notificação disparada pela API *WatchService*, o *ProvWatcher* realiza uma chamada ao *ProvDB* por meio de sua interface. Em seguida, o *ProvDB* realiza a leitura do repositório de proveniência e retorna o identificador do *workflow* e da atividade em execução. Após a consulta, o *ProvWatcher* combina os dados de proveniência retrospectiva explícita com os dados de proveniência implícita recuperados no monitoramento do diretório. Por fim, o *ProvWatcher* envia os dados combinados para o componente *ProvDB* que os armazena

na tabela *changefile* criada pelo *ProvDB* no início da sua execução. A tabela *changefile* é uma extensão adicionada pelo *ProvDB* ao modelo de proveniência utilizado pelo *SGWfC* (se o SGWfC também representar a proveniência usando o modelo relacional).

Para ilustrar o acoplamento do *ProvDB* com um *SGWfC*, a Figura 3 exibe o relacionamento da tabela *changefile* com as tabelas *hworkflow* e *hactivity*, utilizadas pelo *SGWfC* SciCumulus para armazenar proveniência. A tabela *hworkflow* armazena dados sobre os workflows, enquanto a tabela *hactivity* armazena dados sobre as atividades dos workflows. A Tabela 1, Tabela 2 e a Tabela 3 descrevem os atributos das relações *changefile, hworkflow e hactivity,* respectivamente.

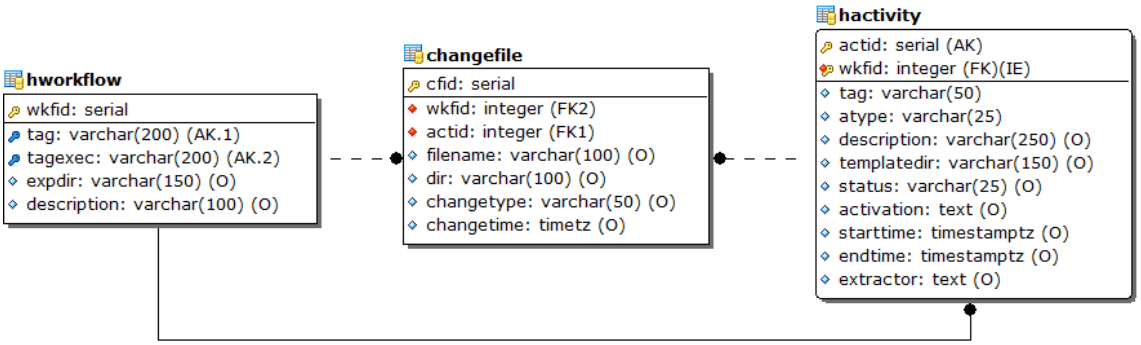

Figura 3. Tabelas hworkflow, changefile e hactivity com seus atributos e relacionamentos.

A estrutura do *ProvWatcher* permite o monitoramento de arquivos de diferentes SGWfC por meio da integração do componente de leitura e escrita (*ProvDB*), desenvolvido conforme o esquema do repositório de dados de proveniência em questão. Desta forma, através de diferentes configurações do componente *ProvDB*, nossa abordagem pode ser utilizada para capturar a proveniência implícita de variados *SGWfC*. Assim, torna-se possível capturar a proveniência retrospectiva intra-atividade, combiná-la com outros tipos de proveniência (retrospectiva explícita e prospectiva) e disponibilizá-la para uma análise mais profunda e fiel ao histórico de mudanças ocorridas durante a execução de cada atividade. Para realizar a análise dos dados de proveniência gerados, pode-se criar consultas em SQL que retornam informações de proveniência de diferentes tipos e granularidades, devidamente relacionadas entre si.

Tabela 1. Descrição dos atributos da tabela *changefile*.

Atributo	Descrição
cfid	Identificador da mudança ocorrida no arquivo
wkfid	Identificador do *workflow* em execução ou executado
actid	Identificador da atividade em execução ou executada
filename	Nome do arquivo modificado
dir	Diretório onde ocorreu a modificação
changetype	Tipo de modificação ocorrida
changetime	Data e horário da modificação

Tabela 2. Descrição dos atributos da tabela *hworkflow*.

Atributo	Descrição
wkfid	Identificador do *workflow* em execução ou executado
tag	Nome do *workflow*
tagexec	Nome da execução do *workflow*
expdir	Diretório onde os dados produzidos serão armazenados
description	Descrição do *workflow*

Tabela 3. Descrição dos atributos da tabela *hactivity*.

Atributo	Descrição
actid	Identificador da atividade em execução ou executada
wkfid	Identificador do *workflow* em execução ou executado
tag	Nome da atividade
atype	Tipo de atividade
description	Descrição da atividade
templatedir	Diretório do *template* de script da atividade
status	Estado atual da atividade em execução ou executada
activation	Comando ou script de execução da atividade
starttime	Data e hora de início da execução da atividade
endtime	Data e hora de término da execução da atividade
extractor	Nome e diretório do programa extrator de dados

4. Estudo de Caso

Para avaliar a viabilidade da abordagem ProvWatcher foi desenvolvido um estudo de caso utilizando o mecanismo de execução de *workflow* em ambiente de nuvem denominado SciCumulus [Oliveira *et al.* 2010]. Além de executar o *workflow* em paralelo, o SciCumulus captura os dados de proveniência durante a execução de cada atividade. Estes dados são persistidos em um banco de dados relacional PostgreSQL e, a partir dele, consultas em SQL podem ser realizadas para a obtenção de informações sobre a origem dos resultados e suas correlações.

No experimento aqui apresentado, o SciCumulus executou o *workflow* SciPhy num ambiente real centralizado. Um diretório raiz denominado *exp_SciPhy* foi criado no diretório raiz ("*/*") do sistema operacional Linux (Ubuntu 13.04). Os arquivos utilizados pelo *workflow* SciPhy foram então transferidos de um servidor remoto para o diretório local "*exp_SciPhy"*. Para cada atividade em execução, o SciCumulus altera o estado (campo *status* de *hactivity* na Figura 3) de cada atividade no banco de dados de proveniência para "RUNNING" em seu início, e esse estado é novamente alterado para "FINISHED" ao fim da execução. Para capturar a proveniência implícita gerada, foi necessário registrar o diretório no ProvWatcher e configurar o componente ProvDB para a conexão e acesso ao banco de dados do SciCumulus.

Assim que o SciPhy é executado, o ProvWatcher inicia o processo de monitoramento e captura dos eventos de manipulação dos arquivos utilizando o método não bloqueante *take()* da API *WatchService*. Quando um novo evento é capturado, o ProvWatcher faz uma chamada ao componente ProvDB, solicitando o identificador do *workflow* e da atividade em execução (*wkfid* e *actid,* respectivamente). Então, o ProvDB consulta a base de dados do SciCumulus, retorna as informações solicitadas pelo ProvWatcher e cria uma nova tabela (*changefile*) para armazenar a proveniência capturada (somente na primeira iteração). Em seguida, o ProvWatcher relaciona as informações capturadas (nome do arquivo, diretório, tipo de modificação e data e horário) com as informações da atividade que está sendo executada naquele momento (com estado "RUNNING" na tabela *hactivity* do SciCumulus) e passa estas informações ao ProvDB. Após a obtenção destes dados, o ProvDB persiste-os na tabela *changefile*. Todo processo é repetido até o final da execução do *workflow*.

Durante a execução do SciPhy, os cientistas podem realizar consultas SQL sobre o banco de dados para obter informações sobre a manipulação de dados pelos processos que estão sendo executados. A Tabela 4 exibe dois exemplos de consulta que podem ser

executados sobre a base de dados do SciCumulus em tempo de execução. A consulta do item 1 retorna informações da atividade em execução e das modificações realizadas até aquele momento sobre os respectivos arquivos utilizados. Além de obter informações importantes para o entendimento dos processos internos de cada atividade, este tipo de consulta também pode apoiar o processo de direcionamento do *workflow,* conforme a análise realizada sobre os mesmos em tempo de execução. A consulta do item 2 retorna os arquivos que foram modificados mais de uma vez. Esse tipo de consulta não pode ser executado nas abordagens atuais. Nela podemos identificar quais arquivos foram modificados ao longo da execução das atividades, independente da especificação do workflow. Assim, o cientista pode depurar o *workflow*, a fim de verificar inconsistências e prováveis pontos de melhoria. Esta consulta também nos permite obter a proveniência em diferentes grãos, relacionando a proveniência implícita capturada no nível intra-atividade (*changefile*) com a proveniência prospectiva e retrospectiva no nível da própria atividade (*hactivity*) e do workflow (*hworkflow*). O resultado obtido pela execução do ProvWatcher e o retorno das consultas acrescentam um maior nível de detalhamento sobre a proveniência retrospectiva, uma vez que estas informações não eram capturadas e disponibilizadas pelo SciCumulus. Assim, os dados gerados pelo monitoramento do ProvWatcher puderam satisfazer as consultas propostas possibilitando a validação do experimento executado.

Tabela 4. Consultas sobre a base de dados de proveniência

Item	Consulta
1	**Select** w.wkfid. w.tag, a.actid, a.tag, a.status, c.filename, c.dir, c.changetype, c.changetime **from** hactivity a, h*workflow* w, **changefile c** **where** a.status = 'RUNNING' **and** w.wkfid = a.wkfid **and** a.actid = c.actid **order by** c.changetime
2	**Select** w.wkfid, w.tag, w.tagexec, c.filename, count(c.filename) **from** h*workflow* w, hactivity a, **changefile c** **where** w.wkfid = a.wkfid **and** a.actid = c.actid **and** w.wkfid = 90 **group by** w.wkfid, w.tag, w.tagexec, c.filename **HAVING COUNT**(c.filename)>1

5. Trabalhos Relacionados

Nas abordagens PASS [Holland *et al.* 2008] e ES3 [Frew *et al.* 2008], a proveniência é capturada no nível do SO. Porém, apesar de coletar informações sobre a proveniência retrospectiva implícita, estas abordagens recuperam uma grande quantidade de informações que dificultam o trabalho de análise. Além disto, elas dependem de um sistema operacional específico para serem executadas e exigem um pós-processamento para interligar a proveniência implícita com a proveniência retrospectiva. Por outro lado, ProvWatcher captura a proveniência retrospectiva implícita e a interliga à proveniência prospectiva por meio do relacionamento existente entre a tabela *changefile* e as tabelas que representam os workflows e as atividades no banco de dados de proveniência do SGWfC (*hworkflow* e *hactivity* no caso do SciCumulus). O processo de monitoramento da nossa abordagem coleta os dados sobre diretórios e arquivos diretamente relacionados à atividade que está sendo executada naquele momento. Isto

implica numa quantidade menor de informações, porém mais especializadas, favorecendo o processo de análise.

Algumas abordagens têm utilizado técnicas e ferramentas de Gerência de Configuração de *Software* (GCS) para coletar a proveniência retrospectiva [Koop *et al.* 2010, Neves *et al.* 2013]. Koop *et al.* [2010] propõem um arcabouço que utiliza o GIT com o objetivo de criar ligações fortes (chamados de *strong links*) entre o experimento e seus dados de proveniência. Nesta abordagem é necessário informar na especificação do *workflow* (proveniência prospectiva) quais arquivos deverão ser capturados, o que é dispensado na abordagem proposta nesse artigo. Assim, Koop *et al.* [2010] capturam a proveniência retrospectiva explícita, enquanto nossa abordagem é capaz de capturar a proveniência retrospectiva implícita.

Neves *et al.* [2013] apresentam uma proposta de monitoramento através do versionamento de artefatos após a execução de cada atividade. Nesta abordagem, Neves *et al.* [2013] utilizam o sistema de gerência de configuração GIT. Com ele um *workspace* é criado para armazenar os arquivos (*clonados* a partir do repositório central) que serão gerados e/ou consumidos pelas atividades que compõe o *workflow*. Cada execução de um *workflow* é associada a um novo ramo e no final da execução de cada atividade um *commit* é disparado para gravar todos os eventos de leitura e escrita sobre os dados, gerando novas versões. Após a execução de todas as atividades o comando *push* é executado para retornar os dados modificados ao repositório central. Isto possibilita a obtenção e rastreamento do histórico dos arquivos no contexto de cada atividade através de uma análise *intra* (diferentes atividades sob um *trial*) e *inter-trial* (mesma atividade em diferentes *trials*) [Neves *et al.* 2013]. Por outro lado, este histórico não traz o detalhamento do que aconteceu durante a execução da atividade, pois captura somente a última modificação, sobre os artefatos, realizada por cada atividade. O ProvWatcher, por sua vez, captura as modificações (criação, alteração e exclusão) dos arquivos durante a execução de cada atividade (intra-atividade) sem identificar as alterações internas (linhas) do arquivo como ocorre no trabalho de Neves *et al.* [2013].

6. Conclusão e Trabalhos Futuros

A abordagem de captura de proveniência ProvWatcher faz uso de monitoramento para verificar as alterações sobre um diretório que armazena dados de entrada e saída gerados, alterados ou consumidos por atividades de *workflows*. Cada ação realizada sobre um determinado arquivo é persistida em um banco de dados relacional para a realização de consultas durante e/ou após a execução do *workflow*. Esta abordagem permite que o cientista tenha um maior controle sobre o fluxo de informações de entrada e saída dentro de cada atividade. Ela também garante um controle em granularidade mais fina do que a própria atividade, uma vez que podemos conhecer as ações realizadas no decorrer de sua execução.

A abordagem proposta nesse artigo foca em capturar informações de proveniência relacionadas às operações de criação, modificação e exclusão de arquivos, (uma vez que a API *WatchService* não fornece apoio a operações de acesso) e relaciona-las aos dados de proveniência prospectiva e retrospectiva do *workflow*. Encontra-se em andamento o desenvolvimento de uma nova versão do ProvWatcher onde serão contemplados os acessos de leitura aos arquivos de dados, além da captura do conteúdo das versões intermediárias dos arquivos.

Agradecimentos. Os autores agradecem ao CNPq e à FAPERJ pelo financiamento parcial deste trabalho.

Referências

Callahan, S. P., Freire, J., Santos, E., et al. (2006). VisTrails: Visualization Meets Data Management. In *SIGMOD*. ACM.

Estublier, J. (2000). Software configuration management: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering*. ACM.

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

Frew, J., Metzger, D. and Slaughter, P. (2008). Automatic capture and reconstruction of computational provenance. *Concurrency and Computation: Practice and Experience*, v. 20, n. 5, p. 485–496.

Holland, D. A., Seltzer, M. I., Braun, U. and Muniswamy-Reddy, K.-K. (2008). PASSing the provenance challenge. *Concurrency and Computation: Practice and Experience*, v. 20, n. 5, p. 531–540.

Koop, D., Santos, E., Bauer, B., et al. (2010). Bridging Workflow and Data Provenance Using Strong Links. In: Gertz, M.; Ludäscher, B. [Eds.]. *Scientific and Statistical Database Management*. LNCS. Springer Berlin Heidelberg. p. 397–415.

Ludäscher, B., Altintas, I., Berkley, C., et al. (2006). Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, v. 18, n. 10, p. 1039–1065.

Marinho, A., Mattoso, M., Werner, C., Braganholo, V. and Murta, L. (2011). Challenges in managing implicit and abstract provenance data: experiences with ProvManager. In *TaPP*. USENIX Association.

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

Moreau, L. and Missier, P. (2013). PROV-DM: The PROV Data Model. http://www.w3.org/TR/2013/REC-prov-dm-20130430/, [acessado em 17 de fevereiro de 2014].

Neves, V. C., Braganholo, V. and Murta, L. (2013). Implicit Provenance Gathering through Configuration Management. In *SE-CSE*. IEEE.

Ocaña, K. A. C. S., Oliveira, D. De, Ogasawara, E., et al. (2011). SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes. In: Souza, O. N. De; Telles, G. P.; Palakal, M. [Eds.]. *Advances in Bioinformatics and Computational Biology*. LNCS. Springer Berlin Heidelberg. p. 66–70.

Oliveira, D., Ogasawara, E., Baião, F. and Mattoso, M. (2010). SciCumulus: a lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In *3rd International Conference on Cloud Computing*. IEEE.

Oliveira, W., Oliveira, D. and Braganholo, V. (2014). Experiencing PROV-Wf for Provenance Interoperability in SWfMSs. In *IPAW*.

Oracle (2014). Watching a Directory for Changes. http://docs.oracle.com/javase/tutorial/essential/io/notification.html, [acessado em 18 de maio de 2014].