SGProv: Mecanismo de Sumarização para Múltiplos Grafos de Proveniência[1]

Daniele El-Jaick, Marta Mattoso e Alexandre A. B. Lima

Universidade Federal do Rio de Janeiro, Brazil
{deljaick, marta, assis}@cos.ufrj.br

Resumo. Os Sistemas de Gerência de Workflows Científicos (SGWfC) têm o objetivo de automatizar a construção e execução de experimentos científicos. Várias execuções de workflows são necessárias para realizar um experimento. O rastro de proveniência, coletado pelos SGWfC durante estas execuções, é importante para que os cientistas possam compreender, reproduzir e analisar seus experimentos. Um rastro de proveniência contém o histórico da derivação dos dados, assim, pode ser representado sob a forma de um grafo direcionado e acíclico. Cada execução de um workflow gera um grafo de proveniência. Após várias execuções, por exemplo, explorando parâmetros, inúmeros grafos são gerados. A base de proveniência, portanto, requer um espaço de armazenamento considerável e consultá-la envolve a manipulação de um grande volume de grafos. Consultas típicas de proveniência percorrem os diversos grafos para obter o caminho de derivação (linhagem) dos dados da consulta. Este trabalho propõe um mecanismo de sumarização para grafos de proveniência (SGProv), usando um banco de dados de grafos para armazenar e consultar esses grafos. O objetivo é gerar um único grafo sumário que represente todos os grafos de proveniência gerados durante um experimento, mas com tamanho reduzido e eliminando dados repetidos. Esta abordagem de sumarização visa reduzir o tempo de processamento de consultas de proveniência utilizando apenas o grafo sumário para respondê-las sem precisar reconstruir os grafos originais. Consultas típicas sobre dados de proveniência de execução de workflows mostraram o potencial da nossa solução.

Categories and Subject Descriptors: [H. **Information Systems**]: [H.2. **Database Management**]

1. INTRODUÇÃO

Os experimentos científicos assistidos por computadores são baseados em simulações realizadas pelo encadeamento de programas. Este encadeamento é comumente representado por meio de workflows científicos, que podem ser definidos como uma especificação formal dos passos executados em um experimento científico. A natureza exploratória dos experimentos científicos faz com que um mesmo workflow seja executado sob diversas concepções. Um cientista pode precisar analisar os resultados da execução com diferentes combinações de parâmetros ou com a substituição de programas. Assim, para um único experimento, inúmeras execuções de workflows são realizadas [Mattoso *et al.* 2010].

Cada execução de um workflow gera um rastro de proveniência com a descrição de todas as transformações de dados que ocorreram durante a execução: dados de entrada e resultados intermediários e finais. Os rastros de proveniência são baseados em objetos (dados e programas) e seus relacionamentos (dependências) e são representados na forma de um grafo direcionado acíclico (*directed acyclic graph* – DAG) [Aggarwal e Wang 2010]. Independente da complexidade do workflow, o grafo de proveniência resultante será um DAG. A presença de execuções condicionais e paralelismo, por exemplo, afeta a complexidade do workflow, mas não do DAG. Uma base de dados

[1] Esse artigo foi parcialmente financiado pelo CNPq, Capes e Faperj.

de proveniência é, portanto, composta por vários DAG cujos nós possuem estruturas variáveis e que, muitas vezes, não são previamente conhecidas. Os nós possuem também muitas dependências entre si. Estas características fazem com que o armazenamento e a consulta aos dados de proveniência ainda sejam desafios significativos [Anand *et al.* 2010]. Consultas típicas de proveniência incluem obter a "linhagem" (caminho de derivação) de resultados (finais ou intermediários) de uma execução do workflow. Executá-las eficientemente (curto tempo de resposta) e obter um grafo, ou um conjunto de grafos, como resposta são desafios [Woodman *et al.* 2011].

A abordagem mais usada na literatura para tratar o problema de armazenamento e consulta de dados de proveniência é utilizar um banco de dados relacional como em [Huahai e Singh 2008] e [Anand *et al.* 2012]. O principal problema desta abordagem é a complexidade da especificação e o tempo de processamento das consultas que, na maioria das vezes, envolvem muitas junções entre relações, especialmente em grafos muito volumosos e com muitas arestas entre seus nós. O esquema (descrição da base de dados) rígido do modelo relacional também representa um problema para a sua utilização, uma vez que os dados de proveniência, geralmente, possuem esquema flexível [Davidson e Freire 2008].

Este artigo analisa o uso de um banco de dados de grafos (BDG) para armazenar e consultar grafos de proveniência, pois seu modelo de dados é adequado para as características de consultas desta aplicação. Este modelo é composto por nós, arestas e atributos, que podem ser associados tanto aos nós quanto às arestas. As arestas possuem tipos, sendo possível a existência de várias arestas de tipos diferentes entre quaisquer pares de nós. Não existe esquema rígido que defina previamente os atributos relativos a cada nó ou aresta, tornando o armazenamento de dados bastante flexível. Os BDG utilizam operações clássicas da teoria dos grafos para processar os grafos armazenados, tais como travessias, busca em largura e em profundidade e determinação do menor caminho entre dois nós [Angles e Gutierrez 2008]. Desta forma, usar um BDG para percorrer e recuperar dados de vários grafos de proveniência torna as consultas mais simples e eficientes em relação a um banco de dados relacional, pois não envolve o uso de junções entre relações.

Mesmo com a utilização de um BDG, persiste o problema da manipulação do grande volume de dados dos grafos de proveniência, que pode comprometer o tempo de processamento das consultas. Uma abordagem para tratar grafos volumosos é a sumarização [Yu *et al.* 2010]. [Navlakha *et al.* 2008] propõem um grafo sumário gerado através da sumarização de áreas densas do grafo, de subgrafos completos e bipartidos e de cliques. [Yu *et al.* 2010] propõem um grafo sumário onde: nós de um super-nó devem ter os mesmos atributos com os mesmos valores; uma super-aresta é criada se cada nó de um super-nó tiver pelo menos uma aresta com um nó de outro super-nó. [Liu e Yu 2011] propõem um grafo sumário onde: nós de um super-nó devem possuir atributos em comum, com os mesmos valores (ou valores similares) e o mesmo número (ou número similar) de arestas para super-nós vizinhos. Entretanto, as soluções existentes se dedicam a sumarizar um único grafo que tenha um grande volume de nós, enquanto as consultas de proveniência são realizadas sobre um grande volume de grafos, não necessariamente com muitos nós.

Este artigo propõe um mecanismo de sumarização para múltiplos grafos de proveniência (SGProv) resultantes de execuções de workflows durante um experimento científico computacional. Seu objetivo é gerar um único grafo sumário que represente todos os grafos gerados durante um experimento, de tal modo que caminhos e nós repetidos em mais de um grafo sejam armazenados uma única vez e suas diferenças sejam evidenciadas. O volume de dados do grafo sumário é reduzido em relação ao conjunto de grafos original e as consultas podem ser aplicadas somente a ele, melhorando o tempo de processamento. Para avaliar o SGProv, foram analisados os tempos de processamento de consultas típicas de proveniência aplicadas somente ao grafo sumário e aos grafos da base original.

Este artigo está organizado da seguinte forma: a seção 2 descreve o mecanismo de sumarização proposto, a seção 3 apresenta os resultados obtidos com o SGProv e a seção 4 apresenta as conclusões.

2. SGPROV

Esta seção descreve o modelo de dados e como o grafo sumário é gerado a partir dos grafos de proveniência.

2.1 Modelo de Dados

Inicialmente, é definido o conceito de grafo adotado pelo SGProv.

Definição 1 (Grafo): Um grafo é definido como G=(V, E, A, T), onde V={v_1, v_2,..., v_n} é seu conjunto de nós, E={e_1, e_2,..., e_m} ⊂ (VxV) é seu conjunto de arestas direcionadas, A={a_1, a_2,..., a_p} é o conjunto de atributos associados aos nós ou arestas e T={t_1, t_2,..., t_q} é o conjunto de tipos de arestas. Cada nó $v_i \in V$ tem pelo menos um atributo $a_x \in A$. O valor de a_x no nó v_i é representado por Val(v_i, a_x). Cada aresta $e_j \in E$ possui um único tipo t_k definido por Tipo(e_j), onde $t_k \in T$. O valor do atributo a_y pertencente a uma aresta e_j é representado por Val(e_j, a_y). Como as arestas são direcionadas, Origem(e_j) e Destino(e_j) representam, respectivamente, seus nós de origem e de destino.

Seguindo o modelo apresentado, é definido o conceito de grafo de proveniência.

Definição 2 (Grafo de proveniência): Um grafo de proveniência é definido como G_p=(V_p, E_p, A_p, T_p), onde os nós de V_p representam programas ou dados e as arestas de E_p representam a "linhagem" dos nós. Um atributo "tipo" $\in A_p$ é obrigatório para todos os nós e arestas. Se um nó v_{pi} representa um programa, Val(v_{pi}, tipo) = "programa". Se este nó representa um dado, Val(v_{pi}, tipo) = "dado". O conjunto A_p contém também os atributos encontrados na coleta de proveniência, como, por exemplo, os nomes dos parâmetros de entrada dos programas. Nós que representam programas possuem ainda um atributo "nome" $\in A_p$. O conjunto T_p representa tipos de arestas.

2.2 Grafo Sumário

O SGProv baseia-se no fato de que nós que representam um mesmo programa tendem a ocorrer com frequência em diferentes grafos de proveniência, pois, geralmente, um programa é executado repetidas vezes com diferentes combinações de parâmetros. Os nós que representam os dados apresentam grande variação nestes grafos, pois correspondem às entradas e saídas dos programas. Com base nestas características, o SGProv agrupa os nós e as arestas do conjunto de grafos de proveniência para gerar o grafo sumário, que contém todos os nós e arestas dos grafos originais, mas sem redundâncias. Para processar os grafos armazenados, o SGProv utiliza a busca em largura, que visita os nós do grafo nível a nível, ou seja, todos os nós a uma distância *k* do nó inicial são visitados antes de qualquer nó a uma distância *k+1* do inicial [Haichuan e Kitsuregawa 2013].

Definição 3 (Grafo Sumário): Um grafo sumário é definido por G_S=(V_S, E_S, A_S, T_S). Cada nó $v_s \in V_S$ é chamado de super-nó e cada aresta $e_s \in E_S$ é chamada de super-aresta.

Definição 4 (Super-nó): Um super-nó v_{si}={v_{p1}, v_{p2},..., v_{pn}} ⊂ (V_{p1} U V_{p2} U...U V_{px}), onde Val(v_{p1}, tipo) = Val(v_{p2}, tipo)=...= Val(v_{pn}, tipo) = "programa" e Val(v_{p1}, nome) = Val(v_{p2}, nome) =...= Val(v_{pn}, nome).

Definição 5 (Super-aresta): Uma super-aresta liga dois super-nós se existir pelo menos uma aresta de um nó de um super-nó para um nó do outro super-nó. Uma super-aresta e_{si}={e_{p1}, e_{p2},..., e_{pm}}⊂ (E_{p1} U E_{p2} U...U E_{py}), onde {Origem(e_{p1}), Origem(e_{p2}),..., Origem(e_{pm})} ⊂ Origem(e_{si}), {Destino(e_{p1}), Destino(e_{p2}),..., Destino(e_{pm})} ⊂ Destino(e_{si}) e Tipo(e_{p1}) = Tipo(e_{p2}) =...= Tipo(e_{pm}).

Os nós e as arestas presentes em um único grafo de proveniência são inseridos diretamente no sumário, fora de qualquer super-nó ou super-aresta. Isto ocorre quando um programa é usado em uma

única execução do workflow e, portanto, pertence a um único grafo G_p. Alguns super-nós e super-arestas do grafo sumário possuem um atributo "exec", que consiste em uma lista de identificadores das execuções dos workflows de que fizeram parte. Este atributo torna possível recuperar os caminhos dos grafos originais e responder as consultas de proveniência com base apenas no grafo sumário. Quando todos os super-nós e super-arestas do grafo sumário possuem um mesmo valor no seu atributo "exec" significa que fizeram parte de uma mesma execução do workflow. A ausência do atributo "exec" em um super-nó (super-aresta) indica que ele representa um programa (dependência) presente em todos os grafos de proveniência.

Na figura 1 é apresentado um exemplo do mecanismo proposto. A partir dos grafos de proveniência G_{p1} e G_{p2} é gerado o grafo sumário G_s, onde: o super-nó v_{s1} é criado, pois Val(v_{10}, tipo) = Val(v_{20}, tipo) = "programa" e Val(v_{10}, nome) = Val(v_{20}, nome) = "P2"; o super-nó v_{s2} é criado, pois Val(v_{11}, tipo) = Val(v_{21}, tipo) = "programa" e Val(v_{11}, nome) = Val(v_{21}, nome) = "P3"; e a super-aresta e_{s12} é criada, pois existem arestas (e_1, e_2) entre nós de v_{s1} e v_{s2}, onde {Origem(e_1), Origem(e_2)} ⊂ Origem(e_{s12}), {Destino(e_1), Destino(e_2)} ⊂ Destino(e_{12}) e Tipo(e_1) = Tipo(e_2).

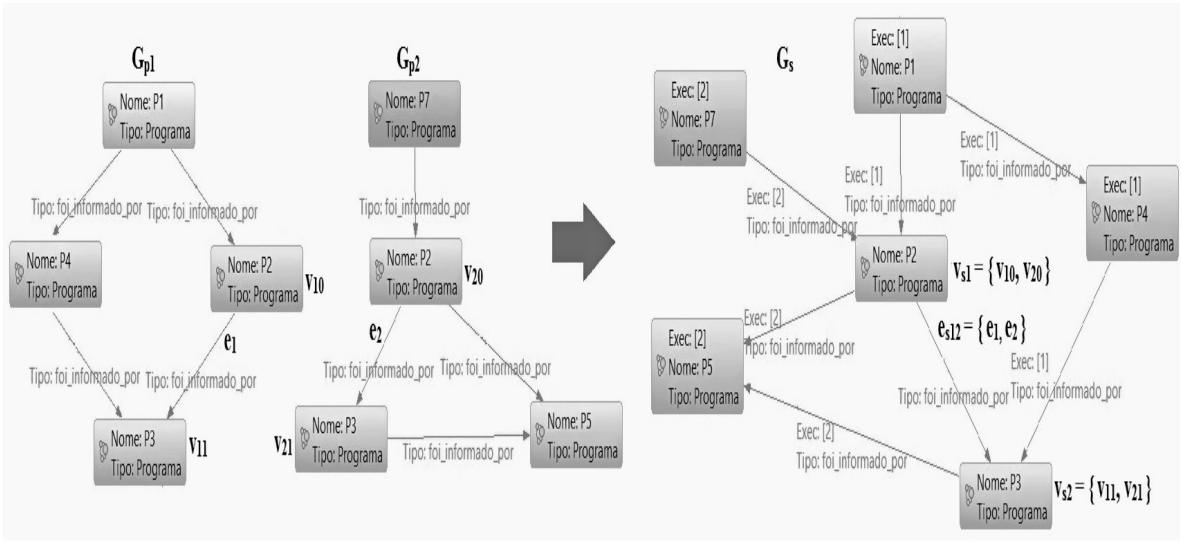

Figura 1. Exemplo do mecanismo de sumarização.

3. AVALIAÇÃO EXPERIMENTAL

O SGProv foi implementado sobre o BDG Neo4J[2] e para as consultas foi utilizada sua linguagem Cypher[3]. Os experimentos foram realizados em uma máquina com processador Intel Core i7, com 2.4 GHz e 8GB de RAM. Para realizar uma avaliação inicial da contribuição do SGProv, foi criada uma base de testes com 1000 grafos de proveniência gerados a partir de execuções de workflows científicos sintéticos. Alguns grafos possuem estruturas similares, mas não iguais, e outros possuem estruturas bem distintas entre si. Eles estão no formato Prov-DM[4] modelo proposto pela W3C[5] (Consórcio *World Wide Web*) como padrão para proveniência. Na base de testes, os nós dos grafos representam programas utilizados por um cientista no seu experimento, onde um programa pode pertencer a um ou mais grafos. Foram definidos 30 programas distintos para workflows sintéticos que

[2] http://www.neo4j.org/
[3] http://www.neo4j.org/learn/cypher
[4] http://www.w3c.org/TR/2012/WD-prov-dm-20120202
[5] http://www.w3.org/TR/prov-aq/

possuem, em média, 15 programas e, em cada execução, ocorre a substituição de programas por outros similares ou a inclusão de novos programas. A tabela 1 mostra características da base de testes, que foi armazenada em um banco de dados, e do grafo sumário produzido pelo SGProv, que foi armazenado em outro banco de dados. Os bancos de dados são gerenciados pelo Neo4J. Como foram definidos 30 programas que se repetem nos grafos de proveniência, o grafo sumário é composto por 30 nós. Isso porque o mecanismo de sumarização agrupa os nós do grafo pela igualdade entre os nomes dos programas. Como o fluxo de dados entre os programas (arestas) pode se repetir nos grafos gerados, o mecanismo de sumarização agrupa as arestas que possuem o mesmo programa de origem e de destino produzindo um número reduzido de arestas no sumário.

Tabela 1. Características da base de testes e do grafo sumário produzido após a aplicação do SGProv.

	Base de Testes	Grafo Sumário
Número de nós	8490	30
Número de arestas	10.149	305

No contexto de experimentos científicos, os cientistas precisam consultar a base de proveniência para obter informações como: a relação entre a produção e o consumo de conjuntos de dados pelos processos, histórico da derivação dos dados e resultados intermediários obtidos. Assim, é possível interpretar, avaliar e validar seus experimentos e detectar possíveis erros [Gadelha e Mattoso 2011].

Foram executadas consultas típicas de proveniência, definidas em [Woodman *et al.* 2011], sobre a base de testes e sobre o grafo sumário para avaliar a variação dos tempos de processamento nos dois casos. As consultas foram realizadas no Neo4J e foram repetidas 100 vezes. A tabela 2 mostra a média dos tempos obtidos no processamento de cada consulta.

Tabela 2. Tempos médios de processamento das consultas de proveniência.

Consultas	Base de Teste	Sumário
C1. Consultar todos os programas do grafo e seus descendentes.	110 ms	15 ms
C2. Consultar os descendentes diretos e os descendentes dos descendentes (descendentes não diretos) do nó que representa o programa cujo atributo “nome” = P5.	93 ms	2 ms
C3. Consultar o menor caminho entre os nós que representam os programas cujo atributo “nome” = P1 e cujo atributo “nome” = P4 e procurar dependências entre os programas cujo atributo “nome” = P4 e cujo atributo “nome” = P5.	-	1 ms
C4. Consultar todos os programas que produziram dados de entrada para o programa cujo atributo “nome” = P10 (consulta aos ascendentes).	35 ms	1 ms

A redução obtida com as consultas realizadas com base no grafo sumário evidenciou o potencial da sumarização. A consulta C3 aplicada à base de testes ultrapassou o tempo limite de processamento. Isso pode ser explicado pelo grande volume de grafos na base de testes e grande quantidade de nós que representam os programas P1, P4 e P5. Uma forma de minimizar este problema seria efetuar a consulta individualmente para cada grafo da base de testes. Entretanto, o tempo de processamento de cada consulta a um grafo da base de testes foi, em média, 5000 ms.

4. CONCLUSÕES

O mecanismo de sumarização reduziu consideravelmente o volume de grafos da base de testes no exemplo analisado. No exemplo, quanto mais os programas e suas dependências com outros programas se repetem nos grafos, melhor é o resultado obtido na sumarização. O custo extra corresponde à introdução do atributo ”exec”, que será medido em trabalhos futuros. Métricas para avaliar a qualidade do sumário e sua prova de correção estão em desenvolvimento. Uma forma de avaliar a qualidade é analisar a taxa de participação dos nós dos super-nós de origem e de destino em cada super-aresta. Taxa de participação maior do que 50% indica uma super-aresta forte, caso

contrário é considerada fraca. O sumário ideal deve possuir o maior número possível de super-arestas fortes [Yu *et al.* 2010]. Uma forma de verificar a correção seria reconstruir os grafos originais a partir do sumário e analisar os resultados das consultas de proveniência aplicadas à base de testes e ao grafo sumário para identificar se estas retornam os mesmos valores sem perdas nem repetições.

Os resultados obtidos nos testes mostraram uma grande redução no tempo de processamento das consultas de proveniência quando aplicadas ao grafo sumário. Por outro lado, as consultas aplicadas aos grafos da base de testes tiveram um desempenho bem menos eficiente. Entretanto, se os grafos da base de testes forem muito distintos entre si, em relação aos programas e suas dependências, o mecanismo de sumarização pode produzir um grafo sumário tão volumoso quanto os da base de testes comprometendo o tempo de processamento das consultas. Neste caso, é preciso avaliar uma forma de fazer a sumarização com base em outras características de nós e arestas dos grafos. Uma forma pode ser agrupar nós que possuem um mesmo conjunto de nós vizinhos [Liu e Yu 2011]. No entanto, a característica principal de grafos de proveniência é a grande similaridade entre eles. Há quase sempre uma grande interseção entre os programas de simulação que fazem parte da modelagem dos workflows explorados nos experimentos científicos.

Os Neo4J apresentou bom desempenho para armazenamento, recuperação e consulta dos grafos da base de dados. Entretanto, uma análise mais ampla sobre a utilização dos BDG para gerenciar grafos de proveniência requer testes com outros sistemas de gerência de banco de dados orientados a grafos.

O mecanismo de sumarização considerou programas repetidos para agrupar os nós e arestas. Entretanto, os grafos de proveniência no formato PROV-DM também possuem outros tipos de nós, como agentes e entidades (dados), e mais quatro tipos de arestas que precisam ser avaliados. Além disso, os programas podem ter vários atributos com valores distintos. Estas questões serão abordadas em trabalhos futuros.

REFERÊNCIAS

AGGARWAL, C., HAIXUN, W. *Managing and Mining Graph Data*. Springer, 2010.

ANAND, M., SHAWN B., LUDASCHER, B. Provenance Browser: Displaying and Querying Scientific Workflow Provenance Graphs. In *Proceedings of the 26th IEEE International Conference on Data Engineering*. Long Beach, USA, pp. 1201-1204, 2010.

ANAND, M., SHAWN B., LUDASCHER, B. Database Support for Exploring Scientific Workflow Provenance Graphs. In *Proceedings of the 24th International Conference on Scientific and Statistical Database Management*. Crete, Greece, pp. 343-360, 2012.

ANGLES, R., GUTIERREZ, C. Survey of Graph Database Models. Journal ACM Computing Surveys, volume 40, issue 1, article 1, 2008.

DAVIDSON, S., FREIRE, J. Provenance and Scientific Workflows: Challenges and Opportunities. In *Proceedings of the of the ACM SIGMOD International Conference on Management of Data*. New York, USA, pp. 1345-1350, 2008.

GADELHA, L., MATTOSO, M. Provenance Query Patterns for Many-Task Scientific Computing. In *Proceedings of the* 3rd *Usenix Workshop on the Theory and Practice of Provenance*. Greece, 351-370, 2011.

HAICHUAN, S., KITSUREGAWA, M. Efficient Breadth-First Search on Large Graphs with Skewed Degree Distributions. In *Proceedings of the 16th International Conference on Extending Database Technology*. Italy, pp. 311–322, 2013.

HUAHAI, H., SINGH, A. Graphs-at-a-time: Query Language and Access Methods for Graph Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. New York, USA, pp. 405-418 , 2008.

LIU, Z., YU, J. On Summarizing Graph Homogeneously. In *Proceedings of the Database Systems for Advanced Applications*. Hong Kong, China, pp. 299-310, 2011.

MATTOSO, M., WERNER, C., TRAVASSOS, G., BRAGANHOLO, V., MURTA, L., OGASAWARA, E., OLIVEIRA, F., CRUZ, S. Towards Supporting Large Scale in Silico Experiments Life Cycle. International Journal of Business Process Integration and Management, v. 5(1), pp. 79-92, 2010.

NAVLAKHA, S., RASTOGI, R., SHRIVASTAVA, N. Graph Summarization with Bounded Error. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Vancouver, Canada, pp. 419-432 2008.

WOODMAN, S., HIDEN, H., WATSON, P. Achieving Reproducibility by Combining Provenance with Service and Workflow Versioning." In *Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science*. New York, USA, pp. 127–136, 2011.

YU, P., HAN J., FALOUTSOS, C., TIAN, Y., PATEL, J. *Link Mining: Models, Algorithms, and Applications*. Springer. 2010.