Rumo à Integração da Álgebra de Workflows com o Processamento de Consulta Relacional*

João Antonio Ferreira[1], Jorge Soares[1], Fabio Porto[2], Esther Pacitti[3], Rafaelli Coutinho[1], Eduardo Ogasawara[1]

[1]CEFET/RJ - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

[2]LNCC - Laboratório Nacional de Computação Científica

[3]Inria & University of Montpellier

joao.parana@acm.org, jorge@eic.cefet-rj.br,fporto@lncc.br,

Esther.Pacitti@inria.fr,rafaelli.coutinho@cefet-rj.br, eogasawara@ieee.org

Abstract. *Workflows emerged as a basic abstraction for structuring data analysis experiments in the current Data Intensive Scalable Computing (DISC) scenario. In many situations, these workflows are intensive, either computationally or in relation to data management, requiring execution in high-performance processing environments. However, parallelizing the execution of workflows commonly requires laborious programming, in an ad hoc manner and in a low level of abstraction, which makes it difficult to explore optimization opportunities. Some algebraic approaches have been developed to mitigate such limitation. This work moves in the direction converging the workflow algebra with relational query processing.*

Resumo. *Os workflows emergiram como uma abstração básica para estruturar experimentos de análise de dados no atual cenário de DISC (*Data Intensive Scalable Computing*). Em muitas situações, estes workflows são intensivos, seja computacionalmente ou em relação à manipulação de dados, exigindo a execução em ambientes de processamento de alto desempenho. Entretanto, paralelizar a execução de workflows comumente requer programação trabalhosa, de modo ad hoc e em baixo nível de abstração, o que torna difícil a exploração das oportunidades de otimização. Algumas abordagens algébricas foram desenvolvidas visando mitigar tal limitação. Este trabalho caminha na direção de convergir a álgebra de workflows com o processamento de consultas relacionais.*

1. Contexto

Apesar de alguns sistemas de workflows possuírem recursos para execução paralela, paralelizar um workflow de larga escala é uma tarefa difícil, *ad hoc* e trabalhosa. Na maioria das soluções existentes, cabe aos usuários dos sistemas decidirem a ordem e as dependências entre as atividades além das estratégias de paralelização. Estas decisões, em muitos casos, restringem as oportunidades de otimização da execução do workflow que poderiam levar a melhorias significativas de desempenho [Ogasawara et al., 2011], principalmente

*Os autores agradecem à FAPERJ, à CAPES e ao CNPq pelo financiamento parcial do projeto.

quando tais workflows são expressos por funções definidas por usuário (UDF) [Rheinländer et al., 2017].

A adoção de uma abordagem algébrica para especificar workflows vem sendo amplamente estudada e permite realizar a otimização da execução paralela de workflows de modo sistemático [Ogasawara et al., 2011; Fegaras, 2017], *i.e.*, de modo que o desenvolvedor não tenha que se preocupar com codificações paralelas, mas tão somente com a especificação do workflow considerando informação do domínio [Liu et al., 2015; Rheinlander et al., 2015].

Na álgebra de workflows, os dados são uniformemente representados por meio de relações e as atividades são regidas por operações algébricas que possuem semântica sobre a produção e o consumo dos dados. Considera-se, nesta álgebra, que atividades consomem e produzem relações. Isso traz uma uniformidade no modelo de dados e possibilita a geração de workflows prontos para execução em paralelo. As relações são definidas como um conjunto de tuplas de dados primitivos (*i.e.*, inteiro, real, alfanumérico, data, etc) ou referência a arquivo (via URI: *Uniform Resource Identifier*). No que tange ao tratamento de arquivos, o seu formato é considerado uma caixa preta. Desta forma, tem-se uma uniformização do tratamento dos arquivos, sejam eles textuais, semiestruturados ou binários. Assim, é de responsabilidade das atividades no workflow saber consumir ou produzir estes dados [Ogasawara et al., 2011].

Cada relação R possui um esquema $\mathcal{R}$ e pode ser especificada como $R(\mathcal{R})$. Dada uma relação $R(\mathcal{R})$, representa-se $atr(R)$ como um conjunto de atributos de R e $key(R)$ como o conjunto contendo os atributos chave de R. De modo análogo à álgebra relacional, relações podem ser manipuladas por operações de conjunto: união ($\cup$), interseção ($\cap$) e diferença ($-$), desde que os seus esquemas sejam compatíveis (aridade da relação e domínio de cada atributo). Pode-se atribuir uma relação a variáveis de relação para posterior reuso usando a atribuição $\leftarrow$ (ex.: $T \leftarrow R_1 \cup R_2$) [Elmasri and Navathe, 2015].

No escopo deste trabalho, uma atividade compreende uma UDF (encapsulando a invocação de um programa ou a execução de uma expressão da álgebra relacional) e esquemas de relações, tanto de entrada quanto de saída. As atividades do workflow são regidas por operações algébricas que especificam a razão de consumo e produção entre as tuplas. Esta característica possibilita um tratamento uniforme para as atividades, viabilizando a realização de transformações algébricas. A álgebra inclui seis operações (resumidas na Tabela 1): Map, $SplitMap$, $Reduce$, $Filter$, $SRQuery$ e $MRQuery$. A maioria das operações algébricas consome uma única relação, com exceção da operação $MRQuery$ que consome uma sequência de relações. As primeiras quatro operações são usadas para apoiar atividades que executam programas encapsulados por meio de UDF. As duas últimas operações são usadas para executar atividades que processam expressões de álgebra relacional. Isso significa que a UDF deve ser compatível com a operação da atividade em termos de interface para consumo e produção de dados [Hsu et al., 2010].

Neste contexto, os workflows podem ser otimizados para execução paralela por meio de transformações algébricas. Cada transformação aplicada, apesar de garantir que o workflow produza o mesmo resultado, traz uma diferença em termos de custo computacional. Em outras palavras, expressões algébricas equivalentes produzem diferentes planos de execução do workflow. Pode-se avaliar o custo destes planos por meio de uma

Tabela 1. Resumo das operações algébricas

Operação	UDF	Operandos adicionais	Resultado	Cons. × Prod. tuplas
Map	programa	Relação R	Relação S	$1:1\ por\ \|R\|$
SplitMap	programa	Relação R	Relação S	$1:m\ por\ \|R\|$
Reduce	programa	Relação R e atrs	Relação S	$n:1\ por\ \|\pi_{atrs}\|$
Filter	programa	Relação R	Relação S	$1:(0-1)\ por\ \|R\|$
SRQuery	exp. relacional	Relação R	Relação S	$n:m$
MRQuery	exp. relacional	Relações $(R_1 \cdots R_i)$	Relação S	$(n_1 \cdots n_i):m$

função de custo. Desta forma, a abordagem algébrica possibilita a visualização do problema de execução paralela de workflow de modo análogo à otimização de consultas em bancos de dados relacionais.

2. Otimização de Workflows

Considere um workflow representado em uma extensão de *XML Process Definition Language* (XPDL)[1] e visualmente apresentado pelo grafo da Figura 1.a. Tal representação tem uma correspondência direta com as expressões da álgebra de workflows descritas na Figura 1.b. Cada relação intermediária tem um esquema específico (no exemplo foram deixados parecidos apenas para facilitar o entendimento).

A semântica presente nas operações algébricas possibilita a reescrita de workflows, gerando expressões equivalentes, porém, com desempenho de paralelismo distinto por meio de um processo de otimização de workflow (Figura 1.c). A navegação por esse espaço de solução, formado por expressões equivalentes de workflows, permite a busca por expressões eficientes. Essa busca, orientada por um modelo de custos de execução paralela, viabiliza, assim, a otimização da execução do workflow.

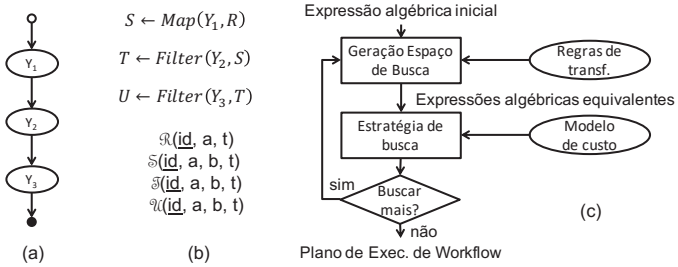

Figura 1. Workflow representado em grafo (a), Workflow representado em álgebra de workflows (b), Processo de otimização de execução de workflows (c)

No escopo deste trabalho considera-se o modelo de execução constituído por uma ativação de atividades e por uma forma simplificada de execução destas ativações, baseadas em distribuição estática de pipelines [Ogasawara et al., 2011], normalmente presentes em sistemas *Data Intensive Scalable Computing* (DISC) [Bryant, 2011], como no caso do Spark [Zaharia et al., 2016]. Uma ativação é um objeto autocontido que possui as informações necessárias para a execução de uma instância da atividade em qualquer um dos núcleos disponíveis [Ogasawara et al., 2011]. Desta forma, no tocante à execução, a

[1] A forma de representar workflows em XPDL pode ser vista em Ogasawara et al. [2013]

ativação contém as informações relacionadas à UDF, os dados (tuplas a serem consumidas e, após a execução da UDF, as tuplas produzidas) e o status de execução. A ativação é inspirada no conceito de ativação em banco de dados [Bouganim et al., 1996]. As ativações podem consumir e produzir conjunto de tuplas; entretanto, uma ativação contém a unidade mínima de dados necessários para que, ainda assim, a execução da instância de uma atividade seja realizada. A razão entre o consumo e a produção de tuplas em uma ativação varia de acordo com a operação algébrica que rege aquela atividade (Tabela 1). A saída de uma atividade é a união de todas as tuplas produzidas por suas ativações.

As execuções das ativações podem ser divididas em três etapas: instrumentação da entrada, invocação de programa e extração de saída. A instrumentação da entrada extrai os valores das tuplas de entrada e prepara para a execução do programa, configurando os valores das entradas dos parâmetros de acordo com os tipos de dados esperados. A invocação de programa despacha e monitora o programa associado à atividade. A extração de saída coleta os dados da saída do programa executado, transformando-os nas tuplas de saída.

Ignorando-se neste momento o modelo de execução (materialização *versus* pipeline) [Elmasri and Navathe, 2015], define-se como otimização de workflow, o processo a partir do qual se exploram as possíveis configurações para se obter uma que minimize o custo computacional. A Figura 1.c descreve o processo de otimização de workflow, que remonta ao processo tradicional de otimização de consultas em banco de dados, no qual foca-se na identificação de um plano de execução de consulta que minimize uma função de custo. A otimização de consultas tipicamente restringe o tamanho do espaço de busca que se pode considerar, por meio de heurísticas que são comumente aplicadas, como as que realizam as operações de projeção e seleção nas relações base de modo a minimizar o número de tuplas intermediárias antes das operações de junções. No modelo relacional, as permutações de uma árvore de junção são comumente as mais importantes para influenciar as consultas relacionais [Tamer Ozsu and Valduriez, 2011], particularmente focando na redução de tuplas das relações intermediárias. Por conta disto, assume-se a aplicação de heurísticas que se concentram na otimização das árvores de junções. Nesta perspectiva, a otimização de workflows proposta neste trabalho possui várias semelhanças em relação às tradicionais otimizações de consultas em banco de dados. O maior objetivo da otimização de workflows é reduzir o tamanho das relações intermediárias que servem como entrada para atividades que invocam UDF com custo computacional elevado[2]. Assim, de modo análogo ao processamento de consultas, no qual a ordem das junções direciona a otimização, na abordagem algébrica para workflows a ordem de execução das UDF direciona a otimização de workflows.

Cabe salientar que as dependências em workflows impõem restrições às transformações algébricas que podem ser aplicadas durante o processo de otimização. Por exemplo, se uma atividade Y_i produz dados que são consumidos por atividades Y_j, a atividade Y_j não pode ser posicionada em uma ordem que anteceda a atividade Y_i. Sempre que possível, antecipa-se a execução de atividades que reduzam a quantidade de tuplas nas relações intermediárias e, correspondentemente, adiam-se as atividades que produzem mais dados.

[2]O custo computacional da execução de uma UDF pode ser conhecido por meio de proveniência

Este cenário de analogia torna-se mais interessante quando se consegue uniformizar o modelo relacional com o de workflows por meio de transformações algébricas. Tais ações possibilitam usufruir do conhecimento de otimização de consultas no contexto de workflows. Desta forma, este trabalho aponta potenciais benefícios ao se formalizar o mapeamento da álgebra de workflows na álgebra relacional. Tal abordagem possibilita elaborar um processador de workflows que tenha o modelo de otimização de consultas como um dos seus alicerces. A Tabela 2 apresenta a relação entre álgebra de workflows e as UDF invocadas nos workflows, e a sua tradução para expressões da álgebra relacional. No caso do Map há uma tradução para a invocação da projeção da álgebra relacional. Neste caso, tem-se o mapeamento dos atributos invocados, que por simplicidade de notação está marcado como asterisco na base de tupla a tupla. De modo análogo, o $SplitMap$ foi mapeado para uma projeção especializada ($\overline{\pi}$) apenas para deixar claro que a UDF realiza consumo tupla a tupla, produz um conjunto de tuplas na saída e que este operador estendido deve ser capaz de considerar esta diferença. O operador $Reduce$ é semelhante às UDF de agrupamento de dados. Finalmente, o operador $Filter$, apesar de operar na seleção, também é um operador tupla a tupla, por ser invocado no predicado da seleção.

Tabela 2. Mapeamento da álgebra de workflows para álgebra relacional

Operação	Álgebra de Workflow	Álgebra Relacional
Map	$S \leftarrow Map(Y, R)$	$S \leftarrow \pi_{u_Y(*)}R$
SplitMap	$S \leftarrow SplitMap(Y, R)$	$S \leftarrow \overline{\pi}_{u_Y(*)}R$
Reduce	$S \leftarrow Reduce(Y, atrs, R)$	$S \leftarrow {}_{atrs}\Gamma_{u_Y(*)}R$
Filter	$S \leftarrow Filter(Y, R)$	$S \leftarrow \sigma_{u_Y(*)}R$

3. Estudo de caso

Considere a execução do workflow da Figura 1, na sua forma original mostrada em (b): (i) $S \leftarrow Map(Y_1, R)$, (ii) $T \leftarrow Filter(Y_2, S)$ e (iii) $U \leftarrow Filter(Y_3, T)$. Levando em conta o esquema das relações do workflow e traduzindo-se tais expressões para álgebra relacional, tem-se: (i) $S \leftarrow \pi_{id,a,b,t}(\pi_{U_{Y_1}(id,a,t)}R)$, (ii) $T \leftarrow \pi_{id,a,b,t}(\sigma_{U_{Y_2}(id,a,b,t)}S)$ e (iii) $U \leftarrow \pi_{id,a,b,t}(\sigma_{U_{Y_3}(id,a,b,t)}T)$.

Com a introdução da semântica dos atributos consumidos e produzidos pelas UDF e a relação de tuplas produzidas nestes contextos, pode-se aplicar as transformações algébricas conhecidas e escrever: $U \leftarrow \pi_{id,a,b,t}(\sigma_{U_{Y_2}(id,a,b,t)}(\sigma_{U_{Y_3}(id,a,b,t)}S))$, retirando-se a relação intermediária T (formando-se um pipeline). Pode-se, até mesmo, inverter a ordem das operações $W \leftarrow \pi_{id,a,b,t}(\sigma_{U_{Y_3}(id,a,b,t)}S)$ e $U \leftarrow \pi_{id,a,b,t}(\sigma_{U_{Y_2}(id,a,b,t)}W)$, materializando-se intermediariamente a relação W antes de produzir U. Note que estas escolhas passam a ser viáveis de serem feitas devido a uniformização da álgebra de workflows com a álgebra relacional e a computação das complexidades por meio de funções de custo. Além disto, esta representação, de modo análogo ao relacional, viabiliza as decisões de modelo de execução (pipeline *versus* materialização) independentemente da especificação do workflow pelos especialistas do domínio.

Foi feita uma avaliação experimental preliminar por meio de dados sintéticos a partir do workflow da Figura 1.b usando o Chiron [Ogasawara et al., 2013]. Considerou-se que a atividade Y_2 apresentava seletividade de 100% (todas as tuplas são aceitas), que

ela é computacionalmente intensiva e que ao mesmo tempo, a atividade Y_3 variava o seu fator de seletividade de 20% a 100% nas diferentes execuções do workflow. A alteração da ordem das atividades Y_2 para Y_3 se mostrou benéfica. Observe que a razão da seletividade variar vem do fato das múltiplas execuções do workflow (sensibilizado pelo atributo b calculado na atividade Y_1). A versão otimizada traz vantagens quando o fator de seletividade de Y_3 é baixo. Este experimento foi executado em um *cluster* com 64 núcleos, com número de tuplas inicial igual a 16.384 e custo médio de execução da atividade Y_1, Y_2 e Y_3, respectivamente, iguais a 1.0, 4.0 e 0.25 segundos. Desta forma, houve uma melhoria no tempo de execução que passou de 22.4 para 8.7 minutos, trazendo uma redução de mais de 60%. Em cenário de larga escala, pode-se considerar que tal abordagem traz possibilidades de avanços na execução dos workflows.

Como trabalhos futuros, pretende-se formalizar de modo mais rigoroso o comportamento das expressões algébricas e as diferentes UDF provinda das diferentes operações da álgebra de workflows. Pretende-se conduzir testes de desempenho de escalabilidade, mostrando tanto volumes de dados em baixa escala quanto em larga escala e discutir o custo computacional envolvido.

Referências

Bouganim, L., Florescu, D., and Valduriez, P. (1996). Dynamic Load Balancing in Hierarchical Parallel Database Systems. In *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB '96, pages 436–447, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bryant, R. (2011). Data-intensive scalable computing for scientific applications. *Computing in Science and Engineering*, 13(6):25–33.

Elmasri, R. and Navathe, S. B. (2015). *Fundamentals of Database Systems*. Pearson, Boston und 24 andere, 7 edition.

Fegaras, L. (2017). An algebra for distributed Big Data analytics. *Journal of Functional Programming*.

Hsu, M., Chen, Q., Wu, R., Zhang, B., and Zeller, H. (2010). Generalized UDF for analytics inside database engine. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6184 LNCS:742–754.

Liu, J., Pacitti, E., Valduriez, P., and Mattoso, M. (2015). A Survey of Data-Intensive Scientific Workflow Management. *Journal of Grid Computing*, pages 1–37.

Ogasawara, E., de Oliveira, D., Valduriez, P., Dias, J., Porto, F., and Mattoso, M. (2011). An algebraic approach for data-centric scientific workflows. In *Proceedings of the VLDB Endowment*, volume 4, pages 1328–1339.

Ogasawara, E., Dias, J., Silva, V., Chirigati, F., Oliveira, D. d., Porto, F., Valduriez, P., and Mattoso, M. (2013). Chiron: a parallel engine for algebraic scientific workflows. *Concurrency and Computation: Practice and Experience*, 25(16):2327–2341.

Rheinlander, A., Heise, A., Hueske, F., Leser, U., and Naumann, F. (2015). SOFA: An extensible logical optimizer for UDF-heavy data flows. *Information Systems*, 52:96–125.

Rheinländer, A., Leser, U., and Graefe, G. (2017). Optimization of complex dataflows with user-defined functions. *ACM Computing Surveys*, 50(3).

Tamer Ozsu, M. and Valduriez, P. (2011). *Principles of Distributed Database Systems*. Springer, New York, 3 edition.

Zaharia, M., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., and Venkataraman, S. (2016). Apache spark: A unified engine for big data processing. *Communications of the ACM*, 59(11):56–65.