Escalonamento de *Workflows* em Nuvens de Computadores com Restrições de Confidencialidade

Rodrigo A. P. Silva[1], **Esther Pacitti**[2], **Yuri Frota**[1], **Daniel de Oliveira**[1]

[1]Universidade Federal Fluminense (UFF), Niterói, RJ, Brasil
[2]Inria & LIRMM, Univ. Montpellier, Montpellier, França

{rprado,yuri,danielcmo}@ic.uff.br, Esther.Pacitti@lirmm.fr

Resumo. *Diversos cientistas tem migrado seus experimentos para a nuvem. Esses experimentos podem ser modelados como workflows científicos, e muito deles são intensivos em dados/computação. Os dados produzidos pelas execuções dos workflows são armazenados na nuvem, o que levanta a preocupação da confidencialidade dos dados, i.e., o risco do acesso não autorizado aos arquivos por conta de usuários maliciosos. Mecanismos como a dispersão dos dados e criptografia podem ser adotados para aumentar a confidencialidade dos dados. Entretanto, a adoção desses mecanismos não pode ser desacoplada do escalonamento do workflow, pois pode aumentar o tempo de execução e seu custo financeiro. Nesse artigo, apresentamos uma heurística de escalonamento de workflows denominada SaFER-GCH (workflow Scheduling with conFidEntiality pRoblem - Greedy random Constructive Heuristic), que considera restrições de confidencialidade dos dados enquanto executa o escalonamento das ativações. Experimentos com traces reais de workflows apresentaram resultados promissores.*

1. Introdução

Os *workflows* científicos têm sido aplicados como um arcabouço para modelar um experimento científico. Esses *workflows* são normalmente modelados como grafos dirigidos, nos quais os nós representam as atividades (*i.e.,* associadas à um programa) e os arcos representam as dependências de dados entre as atividades. Denominamos ativação a execução de uma atividade que consome um dado de entrada específico. Os *workflows* geralmente são modelados e executados usando mecanismos complexos chamados de Sistemas de Gerência de *Workflows* (SGWfs). Os SGWfs apoiam a composição, a execução, o monitoramento e a captura de dados de proveniência dos *workflows*. As nuvens oferecem elasticidade, o que pode ser valioso para a execução de *workflows* em larga escala. Muitos SGWfs permitem a execução de *workflows* em nuvens como o Pegasus e o SciCumulus. Um conhecido problema NP-Difícil é o de escalonar as ativações do *workflow* nas Máquinas Virtuais (VMs). Os SGWfs já oferecem algoritmos de escalonamento que exploram características das nuvens como elasticidade, escalabilidade, *etc*. Ao executar *workflows* na nuvem, os SGWfs geralmente se baseiam em áreas de armazenamento compartilhado (*buckets*) para gravar os dados produzidos, exemplos são o Pegasus e o SciCumulus que usam o serviço S3 da Amazon AWS. Com esse armazenamento centralizado em um único *bucket*, a confidencialidade dos resultados se torna uma questão em aberto, porém de suma importância [Ennajjar et al. 2017]. Os arquivos gerados pelos *workflows* podem conter dados não publicados da pesquisa ou dados confidenciais, e, se os mesmos forem armazenados em conjunto, e um usuário malicioso tiver acesso ao *bucket* em que estão armazenados, o mesmo pode inferir sobre o experimento

e seus resultados, o que não é desejável. De fato, nos últimos anos, muitos cientistas tem evitado migrar experimentos para a nuvem devido às questões de confidencialidade.

Muitas técnicas podem ser usadas para diminuir os riscos de confidencialidade, como por exemplo a dispersão dos dados. O uso de um plano de dispersão é uma das principais técnicas para garantir a confidencialidade dos dados na nuvem [Branco-Jr. et al. 2016], já que esse plano define quais arquivos podem ser armazenados em um mesmo *bucket*. As restrições de armazenamento necessárias para gerar um plano de dispersão podem ser representadas como um grafo de conflito, onde cada arco ligando dois arquivos representa um conflito (*i.e.*, esses dois arquivos não devem ser armazenados no mesmo *bucket*). Abordagens existentes já tratam do problema de dispersão de dados em *workflows* para garantir a confidencialidade. Entretanto, tais abordagens geram planos de dispersão dos dados desacoplados do escalonamento do *workflow*. Essa dispersão dos dados sem considerar o local onde ativações irão produzi-los ou consumi-los pode inserir um *overhead* desnecessário. Logo, é fundamental que o plano de dispersão dos dados esteja integrado ao escalonamento das ativações. Além disso, técnicas de segurança , como a criptografia, podem ser consideradas complementares ao plano de dispersão dos dados. Nesse artigo, apresentamos a heurística SaFER-GCH para o escalonamento de ativações de *workflows*, que leva em consideração a variedade e a heterogeneidade de VMs em um *cluster* virtual na nuvem, os mecanismos de segurança oferecidos por cada VM (*e.g.*, criptografia) e o grafo de conflito entre os arquivos, todos integrados na mesma solução. Por fim, apresentamos, na Seção 2, os trabalhos relacionados, na Seção 3, a heurística de escalonamento proposta, na Seção 4, discutimos a avaliação experimental, e, na Seção 5 concluímos o artigo.

2. Trabalhos Relacionados

Poucos trabalhos da literatura consideram questões de confidencialidade no escalonamento de *workflows* [Sujana et al. 2019, Abazari et al. 2019, Shishido et al. 2018, Guerine et al. 2019]. A abordagem OPTIC [Guerine et al. 2019] propõe uma heurística para dispersão de dados produzidos por *workflows* em *buckets*. Apesar do avanço na confidencialidade dos dados, a OPTIC não escalona o *workflow* ao mesmo tempo que planeja a dispersão, *i.e.*, dados podem ser armazenados geograficamente distantes. Isso pode ser um problema tanto em termos de *makespan* quanto em termos financeiros (maior tempo de transferência dos arquivos, maior inatividade das VMs). Por outro lado, alguns trabalhos consideram a segurança no processo de escalonamento. [Shishido et al. 2018] propõem uma abordagem de escalonamento de *workflows* em nuvem que considera restrições de níveis de segurança para as VMs envolvidas na execução. Apesar de ser um avanço, Shishido *et al.* não considera a dispersão dos dados. [Sujana et al. 2019] e [Abazari et al. 2019] propõem abordagens de escalonamento multi-critério que consideram as demandas de segurança das ativações e as interações na distribuição de ativações seguras na nuvem. Nesse artigo, apresentamos a heurística SaFER-GCH para o escalonamento de *workflow* que aumenta a confidencialidade dos dados consumidos e produzidos ao mesmo tempo que reduz o tempo e o custo de processamento.

3. Escalonamento de *Workflows* com Restrições de Confidencialidade

O problema do escalonamento de ativações com restrições de confidencialidade pode ser resolvido por métodos exatos para instâncias pequenas (com menos de 10 ativações). Para as demais instâncias, esses métodos consomem muitos recursos computacionais e muito tempo

para serem executados. Dessa forma, o uso de métodos exatos, para instâncias que não sejam pequenas, são comumente impraticáveis. Neste artigo, propomos uma heurística construtiva gulosa e aleatória denominada *SaFER-GCH*, para resolver o escalonamento de ativações com restrições de confidencialidade.

Algoritmo 1: Heurística Construtiva Gulosa-Aleatória *SaFER-GCH*

```
   Dados: α_LRC, β
   Resultado: escalonamento S
1  S ← ∅, N̄ ← N
2  enquanto N̄ ≠ ∅ faça
3      LC ← ∅
4      para cada i ∈ N̄ faça
5          para cada j ∈ M faça
6              se Δ_in(i) já foi gerado então
7                  W, FO ← CalculaFO(S, i, j, β)
8                  LC ← LC ∪ (i, j, W)
9      ordena LC por FO
10     LRC ← Obtem_LRC(LC, α_LRC)
11     (i*, j*, W*) ← Sorteia(LRC)
12     S ← S ∪ (i*, j*, W*)
13     N̄ ← N̄\{i*}
14 retorna S
```

No procedimento construtivo da *SaFER-GCH* (Algoritmo 1), o laço mais externo (linha 2) itera até que todas as ativações sejam executadas em alguma VM. Nesse laço, é criada uma Lista de Candidatos (LC) combinando cada ativação ainda não escalonada $i \in \overline{N}$ com todas as VMs disponíveis $j \in M$ (linhas 4 a 8). Porém, nem todas as ativações estão prontas para serem adicionadas à LC porque dependem se as dependências de dados foram satisfeitas, *i.e.*, se seus arquivos de entrada $\Delta_{in}(i)$ já estão disponíveis (linha 6). Para calcularmos o "custo" de cada candidato, o inserimos no escalonamento atual S e calculamos o valor da Função Objetivo (FO) que representa a soma dos custos ponderados (*makespan*, custo financeiro e confidencialidade) de execução de uma determinada ativação $i \in \overline{N}$ na máquina $j \in M$ no escalonamento S (linha 7). Nesse cálculo, definimos também os recursos computacionais $W \subseteq M$ que irão armazenar os dados de saída da ativação i da seguinte forma: para cada dado de saída d selecionamos aleatoriamente β recursos computacionais dentre o conjunto M e escolhemos o de menor acréscimo ao valor da FO que não irá gerar inviabilidade em relação às restrições de capacidade e confidencialidade (baseada no grafo de conflito dado como entrada) para realizar a gravação do dado. Em seguida, ordenamos de forma crescente a LC em função do valor do custo (FO) de cada solução (linha 9) e criamos uma Lista Restrita de Candidatos (LRC) contendo os α_{LRC} (%) melhores candidatos (linha 10). Posteriormente, um candidato é selecionado aleatoriamente de LRC e adicionado ao escalonamento atual S (linhas 11 e 12). O procedimento é repetido até que todas as ativações do *workflow* sejam escalonadas. Dadas as características aleatórias do algoritmo, o procedimento construtivo pode gerar diferentes soluções para um mesmo dado de entrada, *i.e.*, para uma mesma instância do *workflow*. Por isso, em nossos experimentos, executamos o algoritmo da *SaFER-GCH* 100 vezes, armazenando sempre o melhor resultado encontrado.

4. Avaliação Experimental

De forma a avaliar a heurística *SaFER-GCH*, implementamos uma solução exata baseado em um modelo de programação matemática. Além disso, criamos 9 instâncias *sintéticas* de

execuções de *workflows*, pequenas o suficiente para que o método exato encontre alguma solução. Em seguida, executamos a heurística para instâncias reais de *workflows* com características distintas. Tal execução com instâncias reais de execuções de *workflows* seria inviável para o método exato.

Utilizamos, para todos os escalonamentos das instâncias, 4 VMs com diferentes capacidades de armazenamento e processamento; velocidade do enlace; e o custo, por minuto, de processamento em dólares. Em se tratando da quantidade de *buckets*, definimos 2 para as instâncias *sintéticas* e variados números para as demais instâncias. Quanto a capacidade de armazenamento, velocidade do enlace e preço cobrado por GBs de dados armazenados de cada *bucket*, definimos de acordo com os preços apresentados pela sítio *Web* da Amazon AWS[1]. Inserimos a quantidade mínima de *buckets* necessárias para viabilizar cada execução das instâncias nos testes. Além dos *workflows* sintéticos, utilizamos instâncias públicas de *workflows* tradicionais (*i.e., benchmarks*), obtidas no sítio *Web* do *Workflow Generator*[2]. Para guiar a dispersão dos dados durante o escalonamento, criamos o grafo de conflitos mapeando as restrições entre os arquivos dos *workflows* de acordo com as regras a seguir: (i) arquivos de entrada de uma ativação são penalizados (com um valor padrão de 1) se armazenados junto dos arquivos de saída desta mesma ativação, (ii) arquivos gerados por ativações em um mesmo nível do *workflow* (*i.e.*, gerados por ativações "irmãs") não podem ser armazenados juntos. Quanto aos requisitos de segurança, *e.g.* criptografia, utilizamos valores que permitissem que todas as ativações pudessem ser executadas em qualquer VM, pois focamos nas restrições de confidencialidade, uma vez que, requisitos como autenticação ou criptografia são funcionalidades já estudadas em trabalhos anteriores (vide Seção 2).

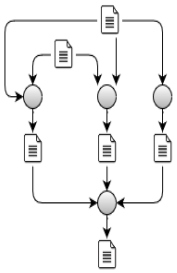

Figura 1. Instância Sintética com 4 ativações e 6 arquivos de entrada/saída

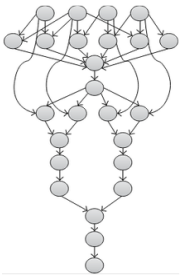

Figura 2. Instância Real Montage

Na execução de uma instância sintética representado pela Figura 1, os resultados de *makespan*, custo financeiro, confidencialidade e tempo de execução do método foram respectivamente: 48.00s, $ 22.42, 0.00, 0.01s pela heurística; e 35.00s, $ 16.81, 0.00, 1039.0s, pela resolução do modelo matemático exato. Repare que a heurística obteve valores maiores de *makespan* e custo para essa instância pequena. Uma explicação para isso é que o método exato permite que as leituras de todas as ativações iniciais ocorram antes de qualquer execução, enquanto que a heurística agrega leitura, execução e escrita atomicamente; isso faz com que o recurso que contém os arquivos de entrada tenha que ser alocado por mais tempo. Nas execuções das instâncias reais de *workflows*, mais especificamente do Montage (Figura 2), obtivemos valores, considerando os mesmos atributos, de: 857.90s, $ 522.29, 0.02 e 0.11s

[1] `https://aws.amazon.com/pt/s3/pricing/`

[2] `https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator`

para 25 ativações; 1532.50s, $ 944.01, 0.01 e 0.41s para 50 ativações; e, 2509.20s, $ 1566.27, 0.00 e 2.25s para 100 ativações. Nesses testes, o método exato não foi capaz de retornar nenhuma solução viável, pois a dimensão da formulação o torna impraticável. Já a heurística conseguiu encontrar soluções muito rapidamente, comprovando, assim, sua utilidade. Considerando todos os testes, os resultados indicam que o modelo matemático foi capaz de retornar soluções ótimas (*i.e.*, menor custo possível) para quase todas as instâncias *sintéticas*, com exceção de duas que interromperam sua execução após 1h (tempo limite configurado para sua execução). Por outro lado, a *SaFER-GCH* conseguiu, para as mesmas instâncias, produzir boas soluções com diferença de, em média, 14% em relação a solução encontrada pelo método exato, mas não conseguiu encontrar a solução ideal para 6 das instâncias sintéticas. Além disso, os tempos de execução para a *SaFER-GCH* são inferiores aos executados pelo método exato (aprox. um centésimo de segundo). Por fim, todos os escalonamentos encontrados pela heurística apresentaram baixos valores para os riscos de confidencialidade.

5. Conclusões e Trabalhos Futuros

Os *workflows* são, geralmente, compostos por programas caixa preta com dependência de dados entre eles. Muitos dados são produzidos durante a execução de *workflows* de larga-escala. Quando essa execução é feita na nuvem, comumente os SGWfs existentes armazenam os dados produzidos em um *storage* compartilhado (*i.e.*, *bucket*). Ao fazer isso, a confidencialidade do experimento pode ser comprometida. Nesse artigo, introduzimos um procedimento heurístico chamado *SaFER-GCH* para resolver o problema de otimização que utiliza um grafo de conflitos para integrar o escalonamento de ativações com o plano de dispersão dos dados determinando as áreas de armazenamento para os arquivos produzidos durante a execução. A abordagem proposta foi avaliada com uma série de instâncias de *workflows* sintéticos e reais, e se mostrou promissora quanto a garantia da confidencialidade. Como trabalho futuro, pretendemos implementar uma meta-heurística (*e.g.*, GRASP) a fim de melhorar a qualidade da solução heurística proposta, além de implementar o *SaFER-GCH* no SGWf SciCumulus.

Referências

Abazari, F., Analoui, M., Takabi, H., and Fu, S. (2019). Mows: multi-objective workflow scheduling in cloud computing based on heuristic algorithm. *SMPT*, 93:119–132.

Branco-Jr., E. C., Monteiro, J. M., Reis, R., and Machado, J. C. (2016). A new mechanism to preserving data confidentiality in cloud database scenarios. In *ICEIS*, volume 291, pages 261–283. Springer.

Ennajjar, I., Tabii, Y., and Benkaddour, A. (2017). Securing data in cloud computing by classification. BDCA'17, New York, NY, USA. ACM.

Guerine, M., Stockinger, M. B., Rosseti, I., Simonetti, L. G., Ocaña, K. A., Plastino, A., and de Oliveira, D. (2019). A provenance-based heuristic for preserving results confidentiality in cloud-based scientific workflows. *FGCS*, 97:697 – 713.

Shishido, H. Y., Estrella, J. C., and Toledo, C. F. M. (2018). Multi-objective optimization for workflow scheduling under task selection policies in clouds. In *CEC*, pages 1–8. IEEE.

Sujana, J. A. J., Revathi, T., Priya, T. S., and Muneeswaran, K. (2019). Smart pso-based secured scheduling approaches for scientific workflows in cloud computing. *Soft. Comp.*, 23(5):1745–1765.