Investigando o Impacto de Containers no Desempenho de um Workflow Científico

Bruno da Silva Alves[1], **Andrea S. Charão**[1]

[1]Laboratório de Sistemas de Computação
Universidade Federal de Santa Maria (UFSM)

{bdalves,andrea}@inf.ufsm.br

Abstract. *The description of scientific workflows through a standard language provides, among other things, portability and scalability to scientific workflows. Meeting portability and reproducibility, containers represent a lightweight virtualization. Thus, we propose an investigation of the impacts of containers on the performance of scientific workflows through the execution of Hecil. The Toil workflow engine reads and executes the Hecil workflow and we present two containerized strategies for task scheduling. The results shows that Toil creation and removal of containers can represent a bottleneck in the execution of workflows.*

Resumo. *A descrição de workflows científicos através de uma linguagem padrão garante, entre outras coisas, portabilidade e escalabilidade aos fluxos de trabalhos científicos. Ao encontro da portabilidade e reprodutibilidade, os contêineres representam uma forma de virtualização leve. Dessa forma, o presente trabalho propõe-se a investigar os impactos gerados pelos contêineres no desempenho de workflows científicos através da execução do workflow Hecil. A engine de workflow Toil foi utilizada para a leitura e execução do fluxo e duas estratégias conteinerizadas são utilizadas para o escalonamento das tarefas. Os resultados mostram que o controle de criação e remoção de contêineres feito pelo Toil pode representar um gargalo na execução de workflows.*

1. Introdução

Cientistas de vários campos utilizam ferramentas de *software* no desenvolvimento de trabalhos e no processamento de dados coletados durante os experimentos. Diversas áreas da ciência destacam-se por utilizar tal estratégia, dentre as mais populares está a área de bioinformática com o estabelecimento de fluxos para sequenciamento de DNA. Também pode-se citar áreas como astronomia e engenharia. O estabelecimento de um fluxo de trabalho garante portabilidade, escalabilidade e reprodutibilidade aos experimentos. A utilização de uma linguagem para a descrição de um *workflow* permite que cientistas compartilhem seus experimentos, dessa forma possibilitando que outros possam avaliar, replicar e sugerir avanços.

As tarefas de um *workflow* científico são caracterizadas por consumirem e produzirem uma grande quantidade de dados e por isso necessitam de ambientes capazes de tal poder de processamento, como os *clusters*, *grids* e soluções de computação em nuvem presentes em ambientes de HPC (High-Performance Computing). Nos ambientes HPC que executam essas tarefas complexas, é comum que vários usuários possuam acesso ao

mesmo recurso computacional e por tais motivos são necessárias soluções de *software* que gerenciem as máquinas e controlem o escalonamento das diversas tarefas. Existem alguns *software* consolidados como o Slurm[1], HTcondor[2], Apache Mesos[3].

Apesar da definição de um fluxo de trabalho permitir caracterizar as entradas, tarefas e saídas, ainda existem questões sobre a definição de um ambiente capaz de executar tais computações. Um *workflow* científico possui diversas tarefas, onde cada uma possui dependências de bibliotecas, binários, arquivos, variáveis de ambiente, etc. Dessa forma, os contêineres podem provisionar um ambiente estável através de uma virtualização leve e se apresentam como uma solução a ser acrescentada aos fluxos de trabalho científico. Apesar de apresentar algumas vulnerabilidades em relação à segurança [Combe et al. 2016], os contêineres possuem desempenho de CPU, memória e disco satisfatórios dado a flexibilidade e portabilidade que os mesmos acrescentam [Xavier et al. 2013], [Felter et al. 2015], [Dua et al. 2014], [Preeth E N et al. 2015].

Assim, neste trabalho é proposta uma investigação dos impactos dos contêineres no desempenho de *workflows* científicos. Para tal, experimentos com a *engine* de *workflow* Toil são feitos através da execução do *workflow* Hecil. Uma tradução do Hecil para o padrão aberto de descrição de *workflows* CWL é apresentada e dois cenários são propostos para o posicionamento dos contêineres, sendo que em um deles o Slurm é utilizado como escalonador das tarefas.

2. Fundamentação e Trabalhos Relacionados

2.1. Integração de Contêineres à *workflows* Ciêntificos

Diversas ferramentas e estratégias já foram propostas e analisadas por outros autores na integração de contêineres à *workflows* científicos. [Sweeney and Thain 2018] avaliou diferentes aspectos para uma eficiente integração dos contêineres aos *workflows* científicos, como a composição da imagem do contêiner, tradução de diferentes imagens e a adoção de diferentes estratégias para o posicionamento dos contêineres. Sobre o posicionamento de contêineres, eles concluem que a utilização de um contêiner capaz de encapsular os recursos de toda a máquina reduz os impactos da criação e remoção de contêineres. [Zheng and Thain 2015] realizaram tal integração utilizando os contêineres do Docker juntamente com a *engine* de *workflow* Makeflow, que escalonava as tarefas por meio do *software* WorkQueue. O presente trabalho também faz o uso dos contêineres Docker e propõe a utilização do Toil como *engine* de *workflow*, pois o mesmo é compatível com o padrão aberto CWL, diferentemente do Makeflow, que utiliza uma linguagem de descrição própria.

Outros autores já avaliaram o uso da linguagem CWL como uma ferramenta na integração de *workflows* e contêineres. [Hung et al. 2018] apresentou a ferramenta BioDepot-workflow builder (Bwb) que permite que usuários criem e executem *workflows* científicos através de uma interface do estilo *drag-and-drop*. Apesar dos esforços válidos no construção dessa ferramenta, não é apresentada uma análise sobre o desempenho na execução das tarefas. Já [Jansen et al. 2020] propõe uma *framework* chamada de Curios

[1]Slurm: `https://slurm.schedmd.com/`
[2]HTcondor: `https://research.cs.wisc.edu/htcondor/`
[3]Apache Mesos: `http://mesos.apache.org/`

Container para execução de *workflows* CWL com o auxílio do formato de arquivo RED (Reproducible Experiments Description) que descreve experimentos que possam ser automaticamente replicados. [Perez-Riverol and Moreno 2019] exploram ainda outras duas *engines* de *workflows*: Galaxy e Nextflow juntamente com os BioContainers.

2.2. *Workflow* Hecil

O fluxo de trabalho descrito pelo Hecil é um exemplo de um *workflow* científico da área da bioinformática. Este fluxo atua no sequenciamento de genomas e diferencia-se por utilizar uma abordagem de aprendizagem iterativa para a correção de possíveis fragmentações que ocorrerem em sequenciamentos longos. Uma das tarefas do Hecil é realizar o mapeamento de sequências contra um genoma de referência, para tal, o mesmo utiliza o algoritmo BWA-MEM, que faz parte do pacote de *software* chamado de BWA (Burrows-Wheeler Aligner)[4]. O pacote BWA possui três diferentes tipos de algoritmos de mapeamento: BWA-backtrack, BWA-SW e BWA-MEM. O BWA-MEM caracteriza-se por ser capaz de ler longas sequências e também por possuir um melhor desempenho do que os demais algoritmos.

A Figura 1 demonstra o grafo do *workflow* Hecil, onde os quadrados azuis representam os arquivos de entrada para cada tarefa e os círculos verdes representam as tarefas em si. Tal fluxo de trabalho ser dividido em duas grandes partes: a primeira, onde o mapeamento das sequências é feito (representado pelo retângulo vermelho) e a segunda parte (representado pelo retângulo amarelo) onde a correção iterativa é feita. O presente trabalho foca na execução da primeira parte deste fluxo de trabalho, pois é nela onde o algoritmo BWA-MEM é executado e tal algoritmo está presente em diversos outros *workflows* da bioinformática como o BWA e BWA-GATK.

2.3. Padrão CWL

A definição de uma linguagem capaz de descrever fluxos de trabalho científico já foi o foco de pesquisa de vários autores como [Stefansen 2005], [Albrecht et al. 2012], [van der Aalst and ter Hofstede 2005]. Umas das características buscadas ao descrever tal linguagem é garantir uma maior portabilidade aos fluxos de trabalho, uma vez que a descrição permite que o mesmo possa ser compartilhado e executado em vários meios. No entanto, a diversidade de linguagens disponíveis acaba gerando um problema, dado que as ferramentas que lidam com *workflows* devem dobrar esforços para abranger várias especificações, assim tornando tal portabilidade questionável. Por isso, os esforços para a criação de um padrão são tão importantes. O CWL (*Common Workflow Language*) é um padrão aberto para descrição de *workflows* e ferramentas, tornando-os portáveis e escaláveis para uma variedade de *software* e *hardware*. O padrão foi desenvolvido para suprir as necessidades de áreas como bioinformática, astronomia e física [Chapman et al. 2016].

O *workflow* Hecil já foi descrito em uma linguagem chamada de Makeflow, a qual apresenta uma sintaxe parecida com os arquivos Makefile do Unix, essa descrição encontra-se disponível em um repositório público[5]. No entanto, não foi encontrada a

[4]BWA: `http://bio-bwa.sourceforge.net/`

[5]Descrição do Hecil: `https://github.com/cooperative-computing-lab/makeflow-examples/tree/master/hecil`

descrição do mesmo *workflow* utilizando o padrão CWL. Dessa maneira, o presente trabalho também propõe-se a traduzir tal fluxo. A versão da tradução do *workflow* Hecil que foi utilizada nos experimentos já se encontra disponível para acesso[6]. Conforme mencionado anteriormente, o Hecil utiliza o algoritmo BWA-MEM, este utilizado em diversos outros *workflows* da bioinformática, assim, a definição e descrição através do CWL pode ser utilizado como exemplo para a elaboração de diversos outros fluxos.

2.4. *Engine* de *workflow* Toil

Toil é uma *engine* de *workflow* capaz de executar *workflows* descritos através das linguagens CWL e WDL (*Workflow Description Language*) e dividido em três módulos: *Job Store API*, *Batch System API* e *Provisioner*. O Job Store é o módulo que representa uma abstração para o armazenamento dos arquivos necessários pelo *workflow*, nele também é armazenado informações sobre o progresso de execução, assim permitindo que a execução seja retomada com o mínimo de retrabalho caso ocorra alguma falha. O armazenamento pode ser feito localmente através de um local no sistema de arquivos ou de forma distribuída na nuvem, podendo utilizar os *buckets* do AWS ou do Google Cloud. O módulo de *Batch System* pode executar as tarefas na própria máquina local ou utilizar soluções como o Slurm, Mesos, Torque, HTcondor, entre outros. E o módulo Provisioner possui um conjunto de ferramentas para a execução de *workflow* em plataformas de processamento em nuvem.

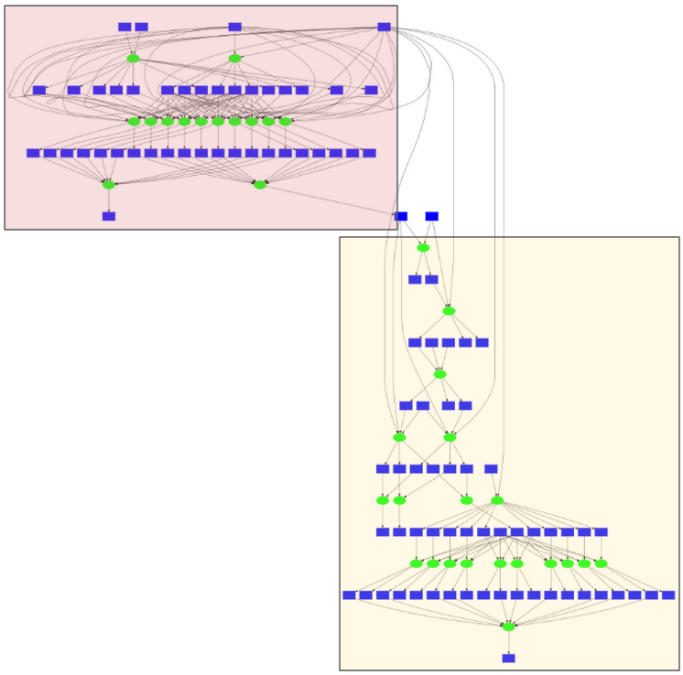

Figura 1. Representação do Wokflow Hecil.

3. Material e Métodos

Para a execução do *workflow* Hecil foi utilizado uma máquina com o sistema CentOS 7.7 que possui um processador Intel(R) Xeon(R) CPU E5-2609 0 com frequência de 2.40GHz

[6]Hecil traduzido: `https://github.com/Alves-Bruno/hecil-cwl`

e com 8 cores e 70 GB de memória RAM. Foram configurados dois cenários para tal execução. No primeiro cenário, o *workflow* descrito no arquivo CWL foi lido pelo *software* Toil, que por sua vez submeteu cada tarefa para o escalonador de tarefas Slurm. Cada tarefa processada pelo Slurm era executada dentro de contêineres Docker, que utilizavam a imagem do Ubuntu:16.04. Já no segundo cenário, o *software* Toil foi instalado e configurado dentro de um contêiner, o qual possuía acesso a todos os recursos da máquina, e as tarefas foram escalonadas pelo próprio Toil. Além disso, para cada cenário, as tarefas do *workflow* Hecil que utilizavam o algoritmo BWA-MEM de alinhamento de DNA foram configuradas para executar em paralelo em 1, 2, 4, e 8 *threads*. O *Workflow* foi executado 10 vezes e os resultados mostrados na Figura 2 representam a média aritmética das execuções. A Tabela 1 apresenta os desvios padrão em segundos das 10 execuções de cada cenário testado.

4. Resultados

A Figura 2 apresenta uma comparação entre os tempos de execução dos dois cenários propostos. Percebe-se que a utilização do Toil conteinerizado obteve menores tempos de execução em todas as configurações de *threads* se comparado à utilização do Toil com o Slurm. O *workflow* executou 100 tarefas do tipo BWA-MEM, que representam a maior parte do tempo de execução. Ainda na Figura 2, a utilização do Toil conteinerizado obteve o menor tempo quando o BWA-MEM foi executado em paralelo com 4 *threads*, por outro lado, quando o Slurm foi utilizado, os tempos de execução se mantiveram próximos em todas as configurações de *threads*. Quando comparadas, as Figuras 3 e 4 auxiliam a responder o porquê da diferença significativa de tempos de execução entre os dois cenários.

A Figura 2 apresenta os tempos de execução e de espera, onde as tarefas permanecerem na fila até serem escalonadas e executadas. Nota-se que o tempo em espera (com o uso do Slurm) das tarefas representam uma parcela significativa do tempo total de execução. Além do acréscimo de tempo sofrido pela própria utilização do Slurm, este cenário também apresenta maiores tempos devido ao custo de inicialização e remoção dos contêineres, os quais devem ser iniciados a cada nova submissão de tarefa.

Durante a elaboração dos experimentos, notou-se também que o *software* Toil não possui a opção de reutilizar os contêineres. Tal técnica representaria uma diminuição do tempo de criação e remoção dos contêineres, uma vez que um contêiner que foi criado e já executou uma tarefa, poderia ser reutilizado para novas tarefas, ao invés de ser simplesmente removido. Com a utilização do Slurm, 104 (número de tarefas) novos contêineres foram criados e tiveram que ser removidos ao final do fluxo. Em contraste, a Figura 4 apresenta os tempos com a utilização do Toil conteinerizado e nota-se que os tempos de espera representam uma parcela muito pequena do tempo total de execução. Neste cenário, apenas 1 contêiner é criado e as tarefas são escalonados pelo próprio Toil e executadas sobre o próprio ambiente do contêiner. Assim os impactos de criação e remoção de contêineres são reduzidos de forma considerável.

5. Considerações Finais

Neste trabalho uma análise do impacto de contêineres no *workflow* Hecil foi avaliada através da proposta de duas estratégias para o posicionamento de contêineres. Apesar da facilidade de se utilizar o Slurm como escalonador de tarefas, pois o mesmo é comumente

Tabela 1. Desvio padrão dos tempos capturados.

	Toil com Slurm		Toil Conteinerizado	
Núm. de *threads*	Tempo de Execução	Tempo em Espera	Tempo de Execução	Tempo em Espera
1	5,435	5,427	3,702	0,076
2	4,960	5,069	0,413	0,038
4	3,632	3,617	0,383	0,046
8	1,917	2,155	0,549	0,040

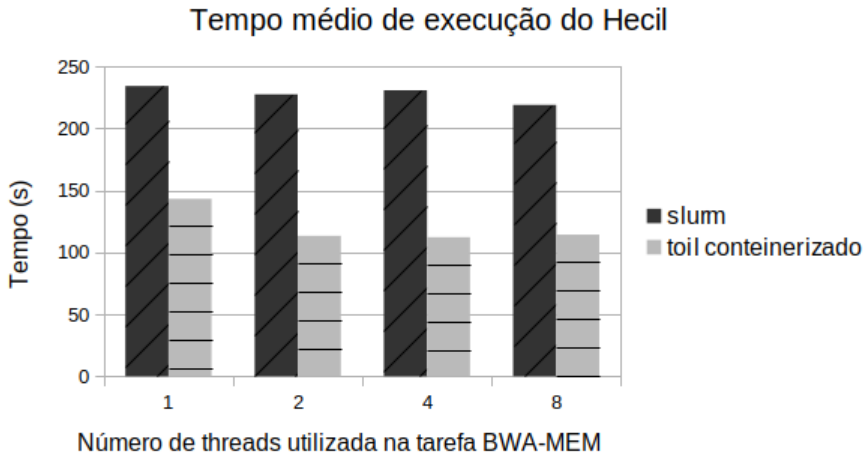

Figura 2. Representação do Wokflow Hecil.

encontrado em máquinas de HPC, percebeu-se que a maneira com que o Toil associado ao Slurm gerencia a criação e remoção de contêineres pode gerar impactos negativos significativos na execução de *workflows* científicos. O cenário que utilizou a estratégia de encapsular toda a execução através de um único contêiner obteve tempos de execução menores. Assim, a impossibilidade da reutilização de contêineres pelo Toil limita a sua aplicação quanto a utilização de contêineres. Pretende-se em trabalhos futuros, explorar o desempenho de outras *engines* de *workflows* que tenham suporte ao CWL, como o Nextflow, Galaxy, cwltool, Arvados, etc. E também avaliar outras ferramentas de contêineres como o Singularity, que pode ser instalado em espaço de usuário, assim diminuindo a dependência da instalação de *software* pelos administradores do sistema.

6. Agradecimentos

Este trabalho foi parcialmente financiado pelo projeto "GREEN-CLOUD: Computação em Cloud com Computação Sustentável" (#162551-0000 488-9), no programa FAPERGS-CNPq PRONEX 12/2014.

Referências

Albrecht, M., Donnelly, P., Bui, P., and Thain, D. (2012). Makeflow: A portable abstraction for data intensive computing on clusters, clouds, and grids. In *Proceedings of the*

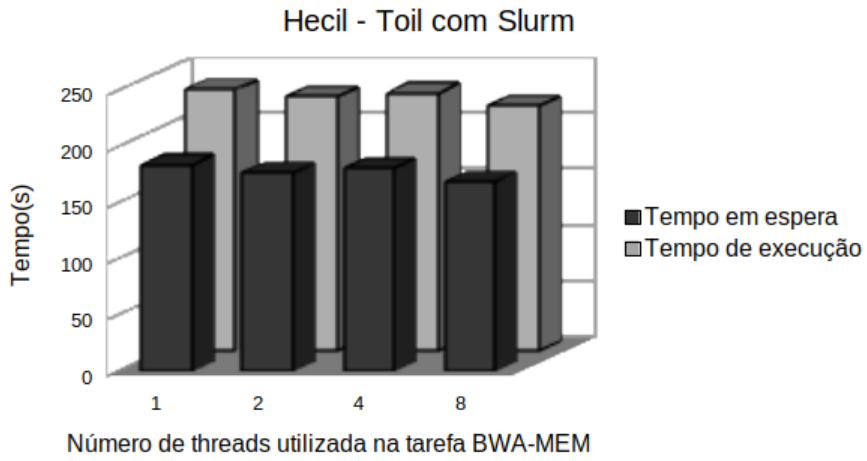

Figura 3. Tempo de execução e tempo em espera com o uso do Slurm.

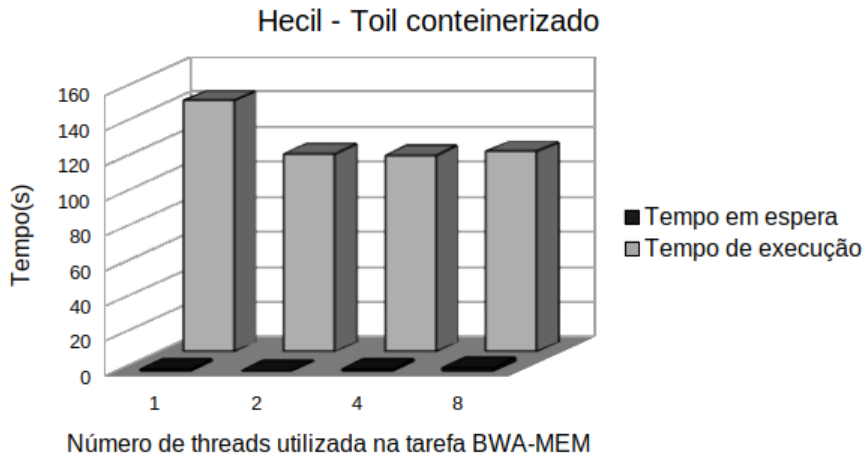

Figura 4. Tempo de execução e tempo em espera com o uso do Toil conteinerizado.

1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, SWEET '12, New York, NY, USA. Association for Computing Machinery.

Chapman, B., Chilton, J., Heuer, M., Kartashov, A., Leehr, D., Ménager, H., Nedeljkovich, M., Scales, M., Soiland-Reyes, S., and Stojanovic, L. (2016). *Common Workflow Language, v1.0*. Specification, product of the Common Workflow Language working group. http://www.commonwl.org/v1.0/.

Combe, T., Martin, A., and Di Pietro, R. (2016). To Docker or not to Docker: A security perspective. *IEEE Cloud Computing*, 3(5):54–62.

Dua, R., Raja, A. R., and Kakadia, D. (2014). Virtualization vs containerization to support PaaS. In *2014 IEEE International Conference on Cloud Engineering*, pages 610–614.

Felter, W., Ferreira, A., Rajamony, R., and Rubio, J. (2015). An updated performance comparison of virtual machines and Linux containers. In *2015 IEEE International*

Symposium on Performance Analysis of Systems and Software (ISPASS), pages 171–172.

Hung, L.-H., Hu, J., Meiss, T., Ingersoll, A., Lloyd, W., Kristiyanto, D., Xiong, Y., Sobie, E., and Yeung, K. Y. (2018). Building containerized workflows using the biodepot-workflow-builder (bwb). *bioRxiv*.

Jansen, C., Annuscheit, J., Schilling, B., Strohmenger, K., Witt, M., Bartusch, F., Herta, C., Hufnagl, P., and Krefting, D. (2020). Curious containers: A framework for computational reproducibility in life sciences with support for deep learning applications. *Future Generation Computer Systems*, 112:209 – 227.

Perez-Riverol, Y. and Moreno, P. (2019). Scalable data analysis in proteomics and metabolomics using biocontainers and workflows engines. *bioRxiv*.

Preeth E N, Mulerickal, F. J. P., Paul, B., and Sastri, Y. (2015). Evaluation of Docker containers based on hardware utilization. In *2015 International Conference on Control Communication Computing India (ICCC)*, pages 697–700.

Stefansen, C. (2005). Smawl: A small workflow language based on ccs. Technical report, Harvard Computer Science Group Technical Report TR-06-05.

Sweeney, K. M. D. and Thain, D. (2018). Efficient integration of containers into scientific workflows. In *Proceedings of the 9th Workshop on Scientific Cloud Computing*, ScienceCloud'18, pages 7:1–7:6, New York, NY, USA. ACM.

van der Aalst, W. and ter Hofstede, A. (2005). Yawl: yet another workflow language. *Information Systems*, 30(4):245 – 275.

Xavier, M. G., Neves, M. V., Rossi, F. D., Ferreto, T. C., Lange, T., and De Rose, C. A. F. (2013). Performance evaluation of container-based virtualization for high performance computing environments. In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 233–240.

Zheng, C. and Thain, D. (2015). Integrating containers into workflows: A case study using Makeflow, Work Queue, and Docker. In *Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing*, VTDC '15, pages 31–38, New York, NY, USA. ACM.