Plug and Flow: Execução de *Workflows* Científicos em Contêineres com o *Middleware* `AkôFlow`[‡]

Wesley Ferreira[1], **Liliane Kunstmann**[2], **Raphael Garcia**[1], **Marcos Bedo**[1], **Aline Paes**[1], **Daniel de Oliveira**[1]

[1]Universidade Federal Fluminense (UFF) – Niterói – Brasil

[2]Universidade Federal do Rio de Janeiro (UFRJ) – Rio de Janeiro – Brasil

{wesleyferreira,raphaelgarcia}@id.uff.br, lneves@cos.ufrj.br,

{marcosbedo, alinepaes, danielcmo}@ic.uff.br

Resumo. *Neste artigo, demonstramos o `AkôFlow`, um middleware projetado para a execução paralela de workflows em ambientes conteinerizados. Construído sobre a plataforma Kubernetes, o `AkôFlow` permite o escalonamento automático das atividades dos workflows em múltiplos contêineres de acordo com as dependências de dados existentes. Cada atividade pode ser executada em uma imagem Docker distinta, e o middleware realiza a captura nativa de dados de proveniência. Neste artigo de demonstração, executamos o workflow do domínio da astronomia Montage via `AkôFlow`, avaliando diferentes configurações de alocação de recursos.*

1. Introdução

Os *workflows* são abstrações que representam simulações computacionais complexas, geralmente modeladas como Grafos Acíclicos Dirigidos (DAGs). Nesses grafos, os vértices correspondem a atividades, geralmente associadas à execução de programas, e as arestas indicam dependências de dados entre elas [de Oliveira et al. 2019]. Cada execução de uma atividade, chamada de ativação, usa um subconjunto específico de dados e parâmetros de entrada. Embora existam diferentes formas de implementar *workflows*, o uso de Sistemas de *Workflows* (SWs) é comum, pois eles oferecem ferramentas para definir, executar e monitorar *workflows* em diversas infraestruturas computacionais.

Muitos *workflows* são intensivos em processamento ou na geração de dados, o que torna essencial o uso de técnicas de paralelismo aliadas a ambientes de Computação de Alto Desempenho (HPC). Por isso, diversos SWs já incorporam mecanismos para execução eficiente em HPCs, como o Pegasus [Deelman et al. 2021], o SciCumulus [de Oliveira et al. 2010] e o Parsl [Babuji et al. 2019]. Apesar dos avanços, esses SWs ainda são limitados por dependências específicas. O Pegasus, por exemplo, exige o uso do escalonador HTCondor, que não é compatível com todos os ambientes. O SciCumulus foi projetado para nuvens públicas [de Oliveira et al. 2012]. Além disso, os *workflows* geralmente dependem de várias bibliotecas e ferramentas, tornando a pilha de *software* complexa. Isso dificulta o suporte por centros de HPC, que nem sempre conseguem atender a todos os requisitos [Kunstmann et al. 2022].

[*]Vídeo demonstrativo da ferramenta pode ser acessado em `https://youtu.be/RmrAMWkJij4`

[†]Os autores gostariam de agradecer pelo apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) – Código de Financiamento 001, do CNPq e da FAPERJ.

Uma forma de reduzir os problemas de configuração da infraestrutura na execução de *workflows* é o uso de contêineres [Struhár et al. 2020]. Contêineres encapsulam a aplicação com todas as suas dependências, *e.g.*, bibliotecas, arquivos de configuração, *etc.*, garantindo portabilidade e isolamento. Soluções como Singularity [Kurtzer et al. 2017] e Kubernetes[1] foram desenvolvidas para atender demandas específicas de HPC, automatizando tarefas como implantação, escalonamento e gerenciamento. Apesar dos avanços, essas ferramentas não foram projetadas com foco na execução de *workflows* científicos. Ainda carecem, por exemplo, de mecanismos nativos para captura automática de dados de proveniência [Freire et al. 2008] e definição de políticas de escalonamento orientadas a objetivos, como otimização de tempo ou redução de custos. Por outro lado, SWs oferecem suporte nativo à proveniência e permitem configurar políticas de escalonamento com base em diferentes metas, mas ainda enfrentam limitações em ambientes conteinerizados.

O objetivo deste artigo é demonstrar o funcionamento do `AkôFlow` [Ferreira et al. 2024], um *middleware* desenvolvido para executar *workflows* científicos de forma eficiente em ambientes conteinerizados. Construído sobre a plataforma Kubernetes, o `AkôFlow` gerencia múltiplos contêineres com base nas dependências de dados entre as atividades especificadas no *workflow*, permitindo o uso de contêineres com diferentes capacidades de processamento e armazenamento. Isso oferece maior flexibilidade no escalonamento. Além disso, o sistema realiza a captura automática de dados de proveniência durante a execução, permitindo análise e reprodutibilidade por parte do usuário.

2. O *Middleware* `AkôFlow`

O `AkôFlow` é um *middleware* voltado à execução de *workflows* científicos em ambientes conteinerizados. Ele interage diretamente com a API nativa do Kubernetes para criar e gerenciar *Pods* baseados em contêineres Docker[2]. A arquitetura do `AkôFlow` pode ser vista na Figura 1 e é composta de cinco camadas: (i) Cliente, (ii) Servidor, (iii) *Proxy*, (iv) *Worker* e (v) Metadados e Proveniência. O usuário descreve o *workflow* em um arquivo YAML, especificando a imagem Docker de cada atividade, os recursos necessários (vCPUs, memória, disco) e as dependências de dados entre ativações. Essa definição pode ser enviada via linha de comando, API ou interface *Web*.

No servidor, o `AkôFlow` registra metadados, gerencia a execução das ativações e realiza a captura automática de dados de proveniência. O orquestrador adota uma política de escalonamento gulosa, com suporte aos modelos *First-Data-First* (FDF) e *First-Activity-First* (FAF) [Ogasawara et al. 2011]. O `AkôFlow` também suporta execução em múltiplos *runtimes* (*i.e.*, ambientes HPC), permitindo o uso simultâneo de mais de um *cluster* Kubernetes, além de ser extensível para outros ambientes, como supercomputadores que utilizam contêineres Singularity.

O Cliente envia o arquivo YAML com a especificação do *workflow* ao Servidor via HTTP. Ao recebê-lo, o servidor desserializa a especificação, registra os metadados e prepara as ativações para orquestração. Cada ativação é instanciada como um *Pod* no Kubernetes, de acordo com a configuração de recursos definida. O componente Orquestrador determina a ordem de execução conforme a política de escalonamento adotada,

[1] `https://kubernetes.io/pt-br/`
[2] `https://www.docker.com/`

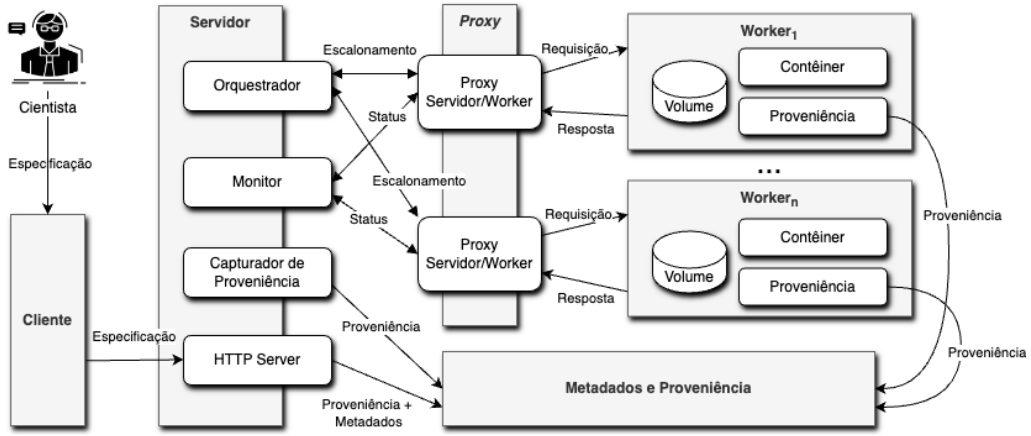

Figura 1. A Arquitetura do `AkôFlow`

atualmente do tipo gulosa, *i.e.*, sempre que um recurso é liberado, uma nova ativação é iniciada.

Uma das funcionalidades do `AkôFlow` é a captura e o armazenamento dos dados de proveniência das ativações executadas. O modelo de proveniência, ilustrado na Figura 2, é composto por cinco tabelas principais: (i) *Workflow*, (ii) *Activity*, (iii) *Metrics*, (iv) *Logs* e (v) *Storages*. A tabela *Workflow* armazena informações gerais sobre cada *workflow* executado, como o *namespace*, o arquivo de definição e o estado atual. Cada *workflow* possui diversas atividades associadas, que são registradas na tabela *Activity*. Cada atividade está vinculada a um único *workflow* e contém informações como o *namespace*, o recurso de execução e o estado da atividade. As tabelas *Logs* e *Metrics* armazenam dados coletados diretamente do ambiente de execução. A tabela *Storages* é a responsável por registrar as informações de armazenamento associadas a cada atividade, como o tamanho do disco, a lista de arquivos presentes no início da execução e os arquivos resultantes ao final. Todos os metadados e dados de proveniência são armazenados em um banco de dados SQLite. O código-fonte do `AkôFlow` se encontra disponível no GitHub na URL `https://github.com/UFFeScience/akoflow`.

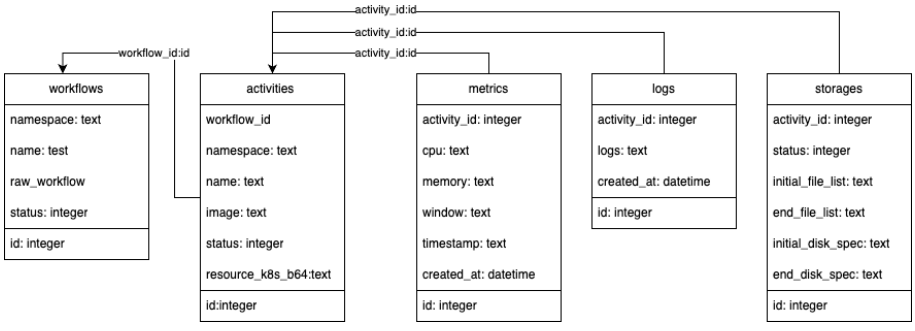

Figura 2. Modelo de Proveniência do `AkôFlow`

3. Demonstração

A demonstração do `AkôFlow` utilizará, como estudo de caso, o *workflow* científico Montage [Sakellariou et al. 2009]. O Montage é amplamente utilizado na área de astronomia para a criação de mosaicos a partir de múltiplas imagens do céu, capturadas por

Figura 3. Monitoramento do *Workflow* Montage pelo `AkôFlow Admin`

telescópios em diferentes momentos e condições. Esse *workflow* é caracterizado pela manipulação e geração de grandes volumes de dados, sendo composto por sete atividades distintas, cada uma delas correspondente à execução de um programa específico do *toolkit* Montage[3].

As etapas do *workflow* são as seguintes (Figura 4): (i) *mProject* (em amarelo), que projeta cada imagem para uma escala e sistema de coordenadas comuns; (ii) *mDiffFit* (em azul), responsável por calcular as diferenças entre imagens sobrepostas e ajustar os valores de fundo; (iii) *mConcatFit* (em vermelho), que agrega os resultados dos ajustes realizados entre pares de imagens; (iv) *mBgModel* (em laranja), encarregado de modelar o fundo com base nos ajustes e sobreposições identificados; (v) *mBackground* (em verde), que aplica as correções de fundo às imagens; (vi) *mImgtbl* (em cinza claro), utilizado para extrair os metadados necessários para a montagem final; e (vii) *mAdd* (em cinza escuro), que realiza a composição final das imagens, gerando o mosaico astronômico completo.

A demonstração consistirá na submissão do arquivo YAML contendo a especificação do *workflow* Montage ao `AkôFlow`, utilizando uma das interfaces disponíveis: (i) por meio de uma chamada a API HTTP, (ii) pela interface de linha de comando (CLI), ou (iii) pela interface *web*, que permite a criação interativa do *workflow*. A escolha da interface será feita com base em fatores como o número de atividades, o volume de dados envolvidos e a familiaridade do usuário com as opções de interação oferecidas pelo `AkôFlow`.

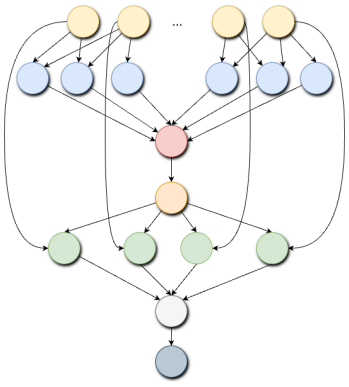

Figura 4. O *Workflow* Montage.

Uma vez submetida a especificação, o servidor do `AkôFlow` é responsável por realizar toda a orquestração e ativação das atividades nos ambientes de execução previamente configurados. O acompanhamento da execução pode ser

[3]`http://montage.ipac.caltech.edu/`

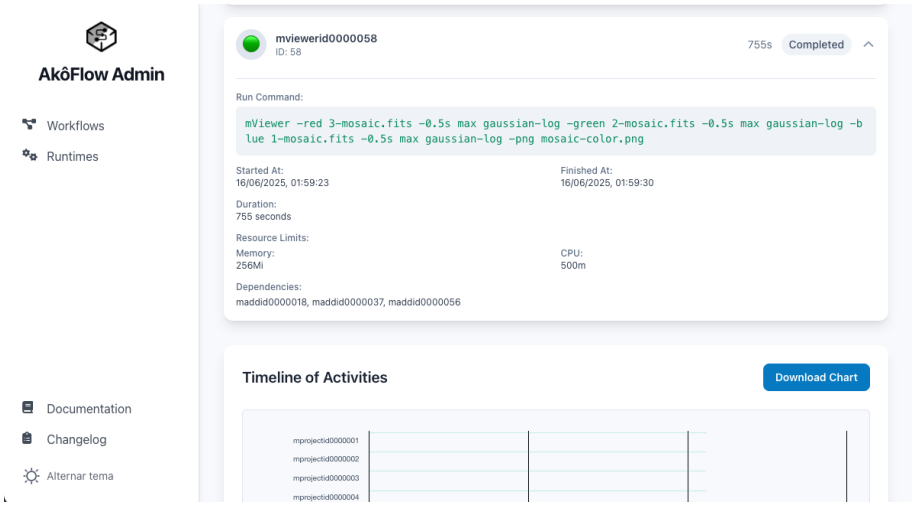

Figura 5. Listagem de Atividades no `AkôFlow Admin`

feito em tempo real por meio do módulo `AkôFlow Admin`, que oferece uma visão detalhada do progresso do *workflow*. A Figura 3 ilustra essa visualização em tempo real, incluindo informações como o status de cada atividade, o comando executado, a imagem de contêiner utilizada e o estado atual de execução. Além do monitoramento da execução, o `AkôFlow` permite também o acompanhamento dos arquivos gerados por cada atividade. Conforme mostrado na Figura 5, é possível acessar uma *timeline* com o histórico de execução de cada atividade do *workflow*.

Adicionalmente, o `AkôFlow` se encontra em processo de desenvolvimento para incorporar novas funcionalidades como a exportação de documentos e grafos de proveniência durante a execução, seguindo o padrão W3C PROV [Belhajjame et al. 2013]. Essa funcionalidade permitirá representar de forma estruturada as dependências, interações e relações entre as atividades executadas, ampliando a transparência, a rastreabilidade e a reprodutibilidade dos experimentos científicos.

4. Conclusão e Trabalhos Futuros

Desde sua proposta inicial no Simpósio Brasileiro de Banco de Dados em 2024 [Ferreira et al. 2024], o `AkôFlow` tem se mostrado como uma ferramenta promissora para a execução de *workflows* científicos em ambientes conteinerizados. Sua capacidade de permitir que um mesmo *workflow* seja definido uma única vez e executado em diferentes ambientes, por meio de contêineres, reforça tanto a portabilidade quanto a reprodutibilidade dos experimentos científicos. O projeto segue em constante evolução, com desenvolvimento ativo e código-fonte disponível no GitHub.

No momento da submissão deste trabalho, o `AkôFlow` contava com 34 estrelas, sendo o repositório mais bem avaliado na *tag eScience*, o que evidencia o interesse da comunidade na solução. Como trabalhos futuros, destacam-se a expansão do modelo de proveniência, com o objetivo de aprimorar a rastreabilidade e a reprodutibilidade das execuções, conforme o padrão W3C PROV, além da implementação de novas heurísticas de escalonamento de atividades, buscando maior eficiência na utilização dos recursos computacionais.

Referências

Babuji, Y. N. et al. (2019). Parsl: Pervasive parallel programming in python. In Weissman, J. B., Butt, A. R., and Smirni, E., editors, *HPDC'19*, pages 25–36. ACM.

Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., Miles, S., Myers, J., Sahoo, S., Tilmes, C., Moreau, L., and Missier, P. (2013). Prov-dm: The prov data model. W3C Recommendation / Technical Report REC-prov-dm-20130430, World Wide Web Consortium. Editors: Luc Moreau and Paolo Missier.

de Oliveira, D., Ocaña, K. A. C. S., Baião, F. A., and Mattoso, M. (2012). A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds. *J. Grid Comput.*, 10(3):521–552.

de Oliveira, D., Ogasawara, E. S., Baião, F. A., and Mattoso, M. (2010). Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In *CLOUD'10*, pages 378–385.

de Oliveira, D. C. M., Liu, J., and Pacitti, E. (2019). *Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Deelman, E., da Silva, R. F., Vahi, K., Rynge, M., Mayani, R., Tanaka, R., Whitcup, W. R., and Livny, M. (2021). The pegasus workflow management system: Translational computer science in practice. *J. Comput. Sci.*, 52:101200.

Ferreira, W., Kunstmann, L., Paes, A., Bedo, M., and de Oliveira, D. (2024). Akôflow: um middleware para execução de workflows científicos em múltiplos ambientes conteinerizados. In *Anais do XXXIX Simpósio Brasileiro de Bancos de Dados*, pages 27–39, Florianópolis/SC. SBC.

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for computational tasks: A survey. *Computing in science & engineering*, 10(3):11–21.

Kunstmann, L., Pina, D., Oliveira, L., Oliveira, D., and Mattoso, M. (2022). Provdeploy: Explorando alternativas de conteinerização com proveniência para aplicações científicas com pad. In *Anais do XXIII Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 49–60, Florianópolis/SC. SBC.

Kurtzer, G. M., Sochat, V., and Bauer, M. W. (2017). Singularity: Scientific containers for mobility of compute. *PloS one*, 12(5):e0177459.

Ogasawara, E. S., de Oliveira, D., Valduriez, P., Dias, J., Porto, F., and Mattoso, M. (2011). An algebraic approach for data-centric scientific workflows. *Proc. VLDB Endow.*, 4(12):1328–1339.

Sakellariou, R. et al. (2009). Mapping workflows on grid resources: Experiments with the montage workflow. In *ERCIM W. Group on Grids*, pages 119–132.

Struhár, V., Behnam, M., Ashjaei, M., and Papadopoulos, A. V. (2020). Real-time containers: A survey. In *Fog-IoT*, volume 80 of *OASIcs*, pages 7:1–7:9.