Uma Heurística para a Execuc ão de Workflows com Restricões de Confidencialidade em Ambientes Conteinerizados

Rodrigo A. P. Silva, Wesley Ferreira, Esther Pacitti, Yuri Frota, Daniel de Oliveira

Uma Heurística para a Execução de *Workflows* com Restrições de Confidencialidade em Ambientes Conteinerizados*

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

[1]Instituto de Computação - Universidade Federal Fluminense – Niterói/RJ – Brasil

{rodrigo_prado, wesleyferreira}@id.uff.br {yuri, danielcmo}@ic.uff.br

[2]INRIA, University of Montpellier, CNRS, LIRMM – Montpellier – França

Esther.Pacitti@lirmm.fr

Resumo. *Ambientes conteinerizados são ideais para a execução de workflows científicos, pois oferecem um ambiente flexível e de fácil instanciação. Embora existam soluções para execução de workflows em ambientes conteinerizados, estas não foram projetadas para lidar com workflows científicos, especialmente aqueles com requisitos de confidencialidade. A não conformidade com esses requisitos permite que usuários mal-intencionados infiram resultados não publicados ou a própria estrutura dos workflows. A dispersão de dados e a criptografia podem ser adotadas nesse contexto, mas não de forma independente do escalonamento, pois isso pode aumentar o tempo total de execução ou o custo financeiro associado. Neste artigo, apresentamos a* `Okinawa`*, uma heurística para execução de workflows em ambientes conteinerizados com restrições de confidencialidade.*

Abstract. *Containerized environments are ideal for running scientific workflows because they offer a flexible and easily instantiated setting. Although there are solutions for executing workflows in containerized environments, these were not designed to handle scientific workflows, especially those with confidentiality requirements. Non-compliance with these requirements allows malicious users to infer unpublished results or the workflow structure itself. Data dispersion and encryption can be adopted in this context, but not independently of workflow scaling, as this can increase the total execution time or the associated financial cost. In this paper, we present* `Okinawa`*, a heuristic for executing workflows in containerized environments with confidentiality constraints.*

1. Introdução

Na últim década, a tecnologia de contêineres tem sido amplamente utilizada na indústria e na academia [Abraham 2023]. Ambientes conteinerizados beneficiam a implantação e execução de diversas aplicações, especialmente *workflows* que necessitam utilizar técnicas de paralelismo aliadas a ambientes de Computação de Alto Desempenho (HPC). É comum que *workflows* em larga escala utilizem diversas bibliotecas e *frameworks*, e essa miríade de soluções cria um ecossistema de *software* complexo [de Oliveira et al. 2019]. Quando não são utilizados ambientes conteinerizados, as instalações de HPC (*e.g.*, supercomputadores) podem enfrentar dificuldades em acompanhar o crescimento acelerado de *softwares* e bibliotecas.

*O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001. Os autores gostariam de agradecer também a CNPq e FAPERJ. Os experimentos foram apoiados pelo *Google Cloud Research Credits* número GCP19980904O.

Embora existam plataformas (*e.g.*, *Google Cloud Workflows*) capazes de executar *workflows* em ambientes conteinerizados, elas não foram projetadas para lidar com *workflows* científicos que possuem requisitos distintos dos *workflows* de negócios ou de ETL. Esses requisitos incluem a captura de dados de proveniência e a transferência otimizada de grandes volumes de dados. No entanto, neste artigo, o requisito tratado é o da confidencialidade dos dados produzidos pelo *workflow*, assim como da sua especificação. Embora contêineres forneçam um certo nível de isolamento, eles apresentam vulnerabilidades que podem ser exploradas por usuários maliciosos [Javed and Toor 2021]. Além disso, a maioria dos sistemas de *workflow* depende de áreas de armazenamento compartilhado, daqui em diante referidas como *volumes*, para armazenar tanto os dados brutos consumidos quanto os dados produzidos pela execução do *workflow*. Por exemplo, o Pegasus fornece mecanismos para armazenar e compartilhar dados em volumes do Amazon S3 (chamados de *buckets*). Embora o uso de armazenamento na nuvem facilite a gestão de dados durante a execução do *workflow*, o compartilhamento de recursos e a própria plataforma do provedor expõem a nuvem a várias classes de ataques [Javed and Toor 2021].

Dessa forma, a confidencialidade dos resultados permanece uma preocupação importante e ainda não resolvida na execução de *workflows* científicos em ambientes conteinerizados, uma vez que os dados produzidos e consumidos podem incluir pesquisas não publicadas ou informações sensíveis. Em uma situação onde esses arquivos são armazenados juntos no mesmo volume ou em um *storage* não confiável, e um usuário mal-intencionado ganha acesso ao volume contendo-os, ele pode inferir a especificação do *workflow* e seus resultados, um cenário indesejável. Para ilustrar essa questão de confidencialidade, consideremos o Montage [Sakellariou et al. 2009], um *workflow* da Astronomia projetado para gerar mosaicos de imagens do céu (Figura 2). A atividade `mConcatFit` do Montage combina múltiplas imagens em uma. Caso um usuário malicioso tenha acesso aos dados de entrada e saída dessa atividade, ele poderá inferir o tipo de transformação de dados aplicada, além de ter acesso a dados possivelmente não publicados do *workflow*.

Esse tipo de ameaça pode ser mitigado com a aplicação de técnicas de dispersão de dados. A definição de quais arquivos podem ou não ser armazenados juntos em um mesmo volume (*i.e.*, *Plano de Dispersão*) adiciona uma barreira adicional contra usuários maliciosos, garantindo a confidencialidade dos dados [Branco-Jr. et al. 2016]. As restrições de armazenamento necessárias para gerar um plano de dispersão podem ser representadas como um *grafo de conflito* (mais detalhes na Seção 2), onde cada aresta conectando dois arquivos representa um conflito (*i.e.*, esses dois arquivos não devem ser armazenados no mesmo volume). Portanto, desenvolver um plano de dispersão eficaz para os dados produzidos e consumidos por um *workflow* é crucial para melhorar a confidencialidade dos dados. Esse plano de dispersão deve considerar os conflitos entre arquivos e a capacidade/custo de armazenamento em volumes.

Trabalhos anteriores [Rosseti et al. 2017, Guerine et al. 2019] abordaram o desafio da dispersão de dados em *workflows* para garantir a confidencialidade, mas de forma desvinculada da execução do *workflow*, *i.e.*, sem considerar em qual contêiner cada *c-activity* (a execução de uma atividade do *workflow* consumindo um conjunto específico de dados e parâmetros em um contêiner) será realizada. Quando o plano de dispersão de dados não leva em conta o contêiner específico em que a *c-activity* será executada, a dispersão de dados pode resultar em uma sobrecarga de transferência não negligenciável, aumentando o tempo total de execução do *workflow*. Portanto, é essencial formular o plano de dispersão

de dados de maneira integrada com o escalonamento das *c-activities*. Além disso, algumas *c-activities* possuem requisitos de segurança específicos, como necessidade de criptografia dos dados ou adição de ruído, e apenas alguns contêineres atendem a essas especificações e podem ser utilizados para executar tais *c-activities*. Essas restrições também devem ser consideradas no momento da execução do *workflow* em um ambiente conteinerizado.

O objetivo deste artigo é garantir a confidencialidade dos dados e da especificação do *workflow*, ao mesmo tempo em que se reduz o tempo total de execução e os eventuais custos financeiros associados à execução de *workflows* em ambientes conteinerizados. Introduzimos uma heurística chamada `Okinawa` (do inglês *wOrKflow executIon iN contAinerized environments With confidentiAlity*), projetada para escalonar *c-activities* de *workflows* em ambientes conteinerizados, levando em consideração a heterogeneidade dos contêineres e os requisitos específicos de cada contêiner, bem como o grafo de conflito de dados do *workflow*. Assim, a `Okinawa` escalona *c-activities* e dispersa dados por múltiplos volumes, visando minimizar tanto o tempo total de execução quanto o risco de acesso malicioso aos dados e de inferência de resultados. As contribuições deste artigo são: (i) a formulação do problema de escalonamento com confidencialidade como um problema de programação inteira, (ii) o desenvolvimento de uma heurística de escalonamento de *workflows* com restrições de confidencialidade, e (iii) uma avaliação experimental da abordagem proposta utilizando o *workflow* Montage.

O restante do artigo segue a seguinte organização. Na Seção 2, serão abordados os modelos de aplicação e arquitetura considerados pela *Okinawa*. Na Seção 3, será apresentada a formulação matemática do problema de escalonamento. Na Seção 4, detalharemos a heurística *Okinawa*. Na Seção 5, discutiremos os resultados experimentais obtidos. A Seção 6 tratará dos trabalhos relacionados, e, por fim, na Seção 7, concluímos o artigo.

2. Modelo de Aplicação e Arquitetura

A heurística `Okinawa` tem como alvo *workflows* representados como Grafos Acíclicos Dirigidos (DAGs). Embora existam diversos formalismos para *workflows*, neste artigo seguimos o formalismo definido por [Teylo et al. 2017]. Um *workflow* pode ser representado como um grafo $G = (D \cup N, A)$, onde o conjunto de vértices é composto por N *c-activities* e um conjunto de arquivos de dados D, enquanto que os arcos no conjunto A definem a relação de produção/consumo entre as *c-activities* e os dados. Uma *c-activity* está associada a execução de um programa em um contêiner, além de encapsular os valores de parâmetros e arquivos de dados a serem consumidos pela mesma.

Na `Okinawa`, é necessário definir outro grafo para realizar a execução das *c-activities* no ambiente conteinerizado: o *grafo de conflito de dados*. Um grafo de conflito de dados pode ser definido como $G_c = (D, E_h \cup E_s, \delta)$, onde os vértices ($D$) em G_c correspondem a todos os dados $d \in D$ consumidos e produzidos pela execução do *workflow*. Além disso, as arestas ($E_h \cup E_s$) representam os conflitos de confidencialidade. Neste artigo, consideramos dois tipos de conflitos: *hard* (E_h) e *soft* (E_s). Conflitos *hard* definem que dados $d_i \in D$ e $d_j \in D$ não podem ser armazenados juntos (no mesmo volume ou contêiner). Por outro lado, um conflito *soft* define que dois dados $d_i \in D$ e $d_j \in D$ preferencialmente não devem ser armazenados juntos, mas

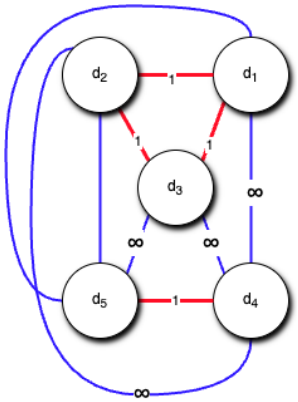

Figura 1. Exemplo de grafo de conflito.

podem ser em cenários específicos (com uma penalidade associada δ_{d_i,d_j}). A Figura 1 apresenta um exemplo de um grafo de conflito, onde arestas vermelhas representam conflitos *soft* e arestas azuis representam conflitos *hard*. Pode-se afirmar que os dados d_1, d_2 e d_3 não devem ser armazenados juntos no mesmo volume, mas se a `Okinawa` definir que devem ser armazenados juntos, uma penalidade de valor 1 é aplicada. No entanto, os dados d_1, d_2 e d_3 não podem ser armazenados juntos com os dados d_4 e d_5. Os conflitos definidos pelas arestas (d_1, d_2), (d_1, d_3), (d_2, d_3) e (d_4, d_5) são conflitos *soft*. Por outro lado, os conflitos definidos pelas arestas (d_1, d_4), (d_1, d_5), (d_2, d_5), (d_3, d_4) e (d_3, d_5) são conflitos *hard*.

Além de formalizar o *workflow* e o grafo de conflito, se faz necessário formalizar as principais características do ambiente alvo, *i.e.*, o ambiente conteinerizado. Definamos C_o como o conjunto de contêineres que podem ser instanciados para executar *c-activities* $i \in N$. Cada contêiner j possui capacidade de armazenamento local cm_j e um índice de desaceleração cs_j. Vale notar que cs_j é inversamente proporcional à capacidade de processamento do contêiner j. Essa métrica é usada para estimar o tempo de execução das *c-activities* em contêineres com capacidades de processamento heterogêneas.

Por exemplo, consideremos que uma *c-activity* $i \in N$ é executada em um contêiner $j \in C_o$. Este contêiner está associado a uma capacidade de processamento cp_j, expressa em GFlops, e um índice de desaceleração $cs_j = 1$. Se considerarmos outro contêiner $j' \in Co$, com capacidade de processamento $cp_{j'}$, o índice de desaceleração do contêiner j' é calculado como $cs_{j'} = \frac{cp_j}{cp_{j'}}$. Os dados podem ser armazenados tanto em um volume atachado ao contêiner $j \in C_o$ quanto em um volume desacoplado de contêineres $j \in B$. Vale notar que apenas os contêineres podem executar *c-activities*. Como tanto contêineres quanto volumes desacoplados podem armazenar dados, chamamos eles de *dispositivos* em nosso formalismo. Assim, o conjunto de dispositivos $\overline{C_o}$ é composto por um conjunto de contêineres C_o e um conjunto de volumes B, *i.e.*, $\overline{C_o} = (B \cup C_o)$.

Os dispositivos podem oferecer mecanismos nativos que podem ser usados pela `Okinawa` para utilizar ou não um contêiner. Tomemos como exemplo os algoritmos de criptografia de dados. Existem diversos algoritmos e cada um oferece um nível de proteção diferente. Dependendo do tipo de dado que a *c-activity* produza, um algoritmo pode ser mais adequado do que o outro. Assim, para formalizar os requisitos de privacidade, definamos o requisito de segurança $r \in R$. Cada requisito r tem um nível máximo, definido como l^r_{max}. Um dispositivo pode oferecer diferentes níveis, *e.g.*, diferentes algoritmos de criptografia. O nível de segurança do requisito $r \in R$ oferecido por um contêiner $j \in C_o$ de l^{rj}_{co}. Uma *c-activity* $i \in N$ pode também ter uma demanda específica de segurança para o requisito r, denotada l^{ri}_{task}. Além disso, definimos o nível máximo de exposição de qualquer dado como s_{max}, *i.e.*, quando os dados são armazenados em um dispositivo que não cumpre os requisitos. Podemos também definir o atraso de comunicação entre dispositivos. Definimos o índice de atraso de comunicação cd_l que representa a latência do link l. O atraso de comunicação de um link l pode ser classificado em dois tipos: atraso de leitura de dados cdr_l e atraso de escrita de dados cdw_l.

3. Formulação Matemática do Problema

A execução do *workflow* realizado pela `Okinawa` pode ser formulada como o problema de programação inteira, chamado `Okinawa-IP`, que estende o trabalho de [Silva et al. 2021], conforme apresentado a seguir. O objetivo da `Okinawa` é realizar o escalonamento das *c-activities* nos múltiplos contêineres de forma que riscos de confidencialidade sejam redu-

zidos ao mesmo tempo em que se otimiza o tempo de execução e o eventual custo financeiro do *workflow* (no caso de ser executado em ambientes onde há cobrança). Em complemento ao formalismo apresentado na Seção 2, podemos definir $D = D_s \cup D_d$ como o conjunto de todos os arquivos de dados, onde cada arquivo de dado $d \in D$ possui um tamanho $W(d)$ e pode ser estático (D_s), com um dispositivo de origem $O(d) \in \overline{C_o}$, ou dinâmico (D_d), gerado durante a execução do *workflow*. Além disso, define-se $\Delta_{in}(i)$ como o conjunto de todos os dados de entrada da *c-activity* $i \in N$, e $\Delta_{out}(i)$ como o conjunto de dados de saída gerados pela *c-activity*. Ademais, t_{max} é definido como o tempo máximo esperado para a execução do *workflow*. Dessa forma, $T = \{1...t_{max}\}$ representa o conjunto de janelas de tempo da execução onde a unidade da janela é definida pelo usuário, *e.g.*, 1 seg. Cada volume $j \in B$ é dividido em intervalos de armazenamento $\{0...|L_j|\}$, cada um com a capacidade sm_{jl}, indicando a faixa do volume utilizada. Assim, c^B_{jl} é o custo de utilização do volume $j \in B$ no intervalo $l \in L_j$, e c^{co}_j é o custo financeiro de instanciar um contêiner $j \in C_o$ por um período de tempo (R$/minuto).

A Tabela 1 resume os parâmetros e variáveis utilizados na formulação matemática proposta para o `Okinawa-IP`. O primeiro passo é a definição da função objetivo, que visa minimizar três métricas: o tempo de execução do *workflow* (1), o custo financeiro (2), e a exposição à segurança e violação de confidencialidade (3). Os pesos α_t, α_b e α_s são utilizados para determinar a importância de cada objetivo, e sua soma deve ser igual a 1. Isso possibilita aos usuários escolherem se desejam uma execução rápida, de baixo custo ou segura. Todos os três objetivos são normalizados usando t_{max}, c_{max} e s_{max}, respectivamente. As restrições em (4) asseguram que cada *c-activity* seja executada. Já as restrições em (5) e (6) estipulam que todas as operações de leitura e escrita, respectivamente, sejam realizadas.

$$\min \quad \alpha_t \cdot \left(\frac{z^T}{t_{max}}\right) + \tag{1}$$

$$\alpha_b \cdot \left(\sum_{j \in C_o} \frac{c^{co}_j z^T_j}{c_{max}} + \sum_{j \in B} \sum_{l \in L_j} \frac{c^B_{jl} q_{jl}}{c_{max}}\right) + \tag{2}$$

$$\alpha_s \cdot \left(\sum_{r \in R} \sum_{i \in N} \frac{e_{ri}}{s_{max}} + \sum_{(d_1,d_2) \in E_s} \frac{\delta_{d_1 d_2} w_{d_1 d_2}}{s_{max}}\right) \tag{3}$$

$$\sum_{j \in C_o} \sum_{t \in T} x_{ijt} = 1, \qquad \forall i \in N \tag{4}$$

$$\sum_{\substack{j \in C_o \\ p \in \overline{C_o}}} \sum_{t \in T} \overrightarrow{x}_{idjpt} = 1, \quad \forall i \in N, \forall d \in \Delta_{in}(i) \tag{5}$$

$$\sum_{\substack{j \in C_o \\ p \in \overline{C_o}}} \sum_{t \in T} \overleftarrow{x}_{idjpt} = 1, \qquad \forall i \in N, \forall d \in \Delta_{out}(i) \tag{6}$$

As desigualdades em (7) garantem que os dados $d \in \Delta_{out}(i)$ só podem ser escritos se a *c-activity* i for executada no momento correto. Adicionalmente, as restrições em (8) determinam que os dados d não podem ser escritos antes do tempo de processamento da *c-activity* i, que é responsável por sua escrita. É importante observar que ambos os conjuntos de restrições em ((7) e (8)) operam em conjunto para garantir um tempo viável para o processo de escrita. As restrições em (9) determinam que uma *c-activity* só pode ser executada se todas as leituras necessárias forem concluídas em um tempo viável.

$$\overleftarrow{x}_{idjpt} \leq \sum_{q=1}^{t-t_{ij}} x_{ijq}, \qquad \forall i \in N, \forall d \in \Delta_{out}(i), \forall j \in C_o, \forall p \in \overline{C_o}, \forall t = (t_{ij}+1) \cdots T_M \tag{7}$$

Tabela 1. Parâmetros e Variáveis da `Okinawa`

Formalismo	Descrição
D_s	Conjunto de dados estáticos (dados já existentes).
D_d	Conjunto de dados dinâmicos (dados produzidos durante a execução do *workflow*).
$O(d)$	Dispositivo que armazena dados estáticos $d \in D_s$.
$W(d)$	Tamanho dos dados $d \in D$.
N	Conjunto de *c-activities*.
B	Conjunto de volumes (desassociados de contêineres).
C_o	Conjunto de contêineres(com poder de processamento e armazenamento).
$\overline{C_o} = (B \cup C_o)$	Conjunto de dispositivos.
L_j	Conjunto de intervalos de armazenamento $\{0 \ldots \mid L_j \mid\}$ do volume $j \in B$
t_{max}	Tempo máximo de execução para o *workflow*.
T	Conjunto de janelas de tempo ($T = \{1 \ldots t_{max}\}$).
t_{ij}	Tempo de processamento da *c-activity* $i \in N$ no contêiner $j \in C_o$.
$\overrightarrow{t}_{djp}$	Tempo gasto pelo contêiner $j \in C_o$ para ler os dados $d \in D$ armazenados em $p \in \overline{C_o}$.
$\overleftarrow{t}_{djp}$	Tempo gasto pelo contêiner $j \in C_o$ para escrever $d \in D_d$ armazenados em $p \in \overline{C_o}$.
$\Delta_{in}(i) \subseteq D$	Conjunto de dados necessários para a execução da *c-activity* $i \in N$.
$\Delta_{out}(i) \subseteq D_d$	Conjunto de dados gerados pela *c-activity* $i \in N$.
cm_j	Capacidade de armazenamento do dispositivo $j \in \overline{C_o}$.
sm_{jl}	Armazenamento do volume $j \in B$ no intervalo $l \in L_j$, onde $sm_{j0} = 0$.
c_j^{co}	O custo financeiro da instanciação do contêiner $j \in C_o$ por um período de tempo.
c_{jl}^{B}	O custo financeiro da utilização do volume $j \in B$ no intervalo $l \in L_j$, onde $c_{j0}^{B} = 0$.
c_{max}	O orçamento máximo financeiro disponível.
α_t, α_b and α_s	Os pesos de tempo, custo e segurança que definem a relevância de cada objetivo na execução do *workflow* ($\alpha_t + \alpha_b + \alpha_s = 1$).
Segurança e Privacidade	**Descrição**
R	Conjunto de requisitos de segurança das *c-activities*.
l_{max}^{r}	O valor nível máximo do requisito de segurança $r \in R$.
l_{task}^{ri}	O nível mínimo do requisito de segurança $r \in R$ exigido pela *c-activity* $i \in N$.
l_{vm}^{rj}	O nível do requisito de segurança $r \in R$ oferecido pelo contêiner $j \in C_o$.
s_{max}	A exposição máxima de segurança e privacidade, definida como $s_{max} = \sum_{r \in R} \lvert N \rvert \cdot l_{max}^{r} + \sum_{(d_1,d_2) \in E_s} \delta_{d_1 d_2}$.
Variáveis	**Descrição**
x_{ijt}	Variável binária que indica se a *c-activity* $i \in N$ inicia sua execução no contêiner $j \in C_o$ no período $t \in T$ ou não.
$\overrightarrow{x}_{idjpt}$	Variável binária que indica se a *c-activity* $i \in N$ executando no contêiner $j \in C_o$ começa a ler os dados $d \in \Delta_{in}(i)$ armazenados no dispositivo $p \in \overline{C_o}$ no período $t \in T$ ou não.
$\overleftarrow{x}_{idjpt}$	Variável binária que indica se a *c-activity* $i \in N$ executando em $j \in C_o$ começa a escrever dados $d \in \Delta_{out}(i) \subset D_d$ no dispositivo $p \in \overline{C_o}$ no período $t \in T$ ou não.
y_{djt}	Variável binária que indica se os dados $d \in D$ estão armazenados no dispositivo $j \in \overline{C_o}$ no período $t \in T$ ou não.
$\overline{y}_{dj}$	Variável binária que indica se os dados $d \in D$ estão armazenados no dispositivo $j \in \overline{C_o}$ em algum período de tempo.
$w_{d_1 d_2}$	Variável binária que indica se os dados d_1 e d_2 estão armazenados no mesmo local ou não, onde $(d_1, d_2) \in E_s$.
e_{ri}	Variável contínua que indica o nível de exposição da execução da *c-activity* $i \in N$ referente ao requisito de segurança $r \in R$.
b_{jl}	Variável binária que indica se o volume $j \in B$ está sendo usado no intervalo $l \in L_j$ ou não.
q_{jl}	Variável contínua que contém o tamanho total dos dados alocados no volume $j \in B$ no intervalo $l \in L_j$.
v_{jt}	Variável binária que indica se o contêiner $j \in C_o$ é usado (instanciado) no momento $t \in T$.
z_j^T	Variável contínua que indica o tempo total de uso do contêiner $j \in C_o$.
z^T	Variável contínua que indica o tempo total para executar o *workflow* (*makespan*).

$$\overleftarrow{x}_{idjpt} = 0, \qquad \forall i \in N, \forall d \in \Delta_{out}(i), \forall j \in C_o, \forall d \in \Delta_{out}(i), 1 \leq t \leq t_{ij} \quad (8)$$

$$x_{ijt} \leq \sum_{p \in \overline{C_o}} \sum_{q=1}^{t-\overrightarrow{t}_{djp}} \overrightarrow{x}_{idjpq}, \qquad \forall i \in N, \forall d \in \Delta_{in}(i), \forall j \in C_o, \forall t \in T, \text{ such } (t - \overrightarrow{t}_{djp}) \geq 1 \quad (9)$$

As desigualdades em (10) garantem que no máximo uma ação (execução, leitura ou escrita) possa ocorrer em um período de um contêiner (*e.g.* um contêiner não pode executar uma *c-activity* e escrever dados ao mesmo tempo). Por outro lado, as desigualdades em (11) determinam que ações passivas (um dado ser lido ou escrito) podem ser realizadas em paralelo. É importante notar que qualquer ação (ativa ou passiva) realizada em um contêiner j no período t terá a variável v_{jt} definida como 1, indicando que o mesmo está em uso.

$$\sum_{i \in N} \sum_{q=\max(1, t-t_{ij}+1)}^{t} x_{ijq} + \sum_{i \in N} \sum_{d \in \Delta_{out}(i)} \sum_{p \in \overline{C_o}} \sum_{r=\max(1, t-\overleftarrow{t}_{djp}+1)}^{t} \overleftarrow{x}_{idjpr} + \sum_{i \in N} \sum_{d \in \Delta_{in}(i)} \sum_{p \in \overline{C_o}} \sum_{r=\max(1, t-\overrightarrow{t}_{djp}+1)}^{t} \overrightarrow{x}_{idjpr} \leq v_{jt}, \qquad \forall j \in C_o, \forall t \in T \quad (10)$$

$$\sum_{i \in N} \sum_{d \in \Delta_{out}(i)} \sum_{p \in C_o} \sum_{r=\max(1, t-\overleftarrow{t}_{dpj}+1)}^{t} \overleftarrow{x}_{idpjr} + \sum_{i \in N} \sum_{d \in \Delta_{in}(i)} \sum_{p \in C_o} \sum_{r=\max(1, t-\overrightarrow{t}_{dpj}+1)}^{t} \overrightarrow{x}_{idpjr} \leq |C_o|.v_{jt}, \qquad \forall j \in C_o, \forall t \in T \quad (11)$$

As restrições em (12) asseguram que não haja dados dinâmicos no início da execução do *workflow*, enquanto as restrições em (13) e (14) garantem que todos os dados estáticos estejam pré-armazenados em seus dispositivos de origem (*i.e.*, um contêiner ou um volume).

$$y_{dj1} = 0, \qquad \forall d \in D_d, \forall j \in \overline{C_o} \quad (12)$$

$$y_{dj1} = 0, \qquad \forall d \in D_s \mid j \in (\overline{C_o} \backslash O(d)) \quad (13)$$

$$y_{djt} = 1, \qquad \forall d \in D_s \mid j \in O(d), \forall t \in T \quad (14)$$

As restrições em (15) e (16) vinculam a variável de armazenamento y com as variáveis de escrita $\overleftarrow{x}$ e de leitura $\overrightarrow{x}$, garantindo um processo de escrita e leitura viável, respectivamente. Em detalhes, a restrição em (15) garante que os dados só estarão armazenados em um dispositivo se já tiverem sido produzidos (escritos) anteriormente. Por outro lado, as restrições em (16) garantem que os dados só serão lidos se estiverem previamente armazenados em um dispositivo.

$$y_{dp(t+1)} = y_{dpt} + \sum_{j \in C_o} \overleftarrow{x}_{idjp(t-\overleftarrow{t}_{djp}+1)}, \qquad \forall d \in D, \forall p \in \overline{C_o}, \forall t \in \{1 \ldots t_{max}-1\}, \text{ such } (t - \overleftarrow{t}_{djp} + 1) \geq 1, d \in \Delta_{out}(i) \quad (15)$$

$$\sum_{j \in C_o} \overrightarrow{x}_{idjpt} \leq y_{dpt}, \qquad \forall i \in N, \forall d \in \Delta_{in}(i), \forall p \in \overline{C_o}, \forall t \in T \quad (16)$$

As capacidades de armazenamento dos dispositivos (volumes e contêineres) são limitadas pelas restrições em (17). As restrições em (18) relacionam a última operação de escrita com o tempo de execução do *workflow* (*i.e.*, *makespan*). Note que, uma *c-activity* sempre produz dados. As restrições em (19) vinculam as variáveis v_{jt} e z_j^T para estabelecer o tempo de alocação (pago) do contêiner j, enquanto as restrições (20) garantem que os custos financeiros não excedam o orçamento máximo definido pelos usuários.

Além disso, a seguinte restrição operacional (21) deve ser satisfeita: uma *c-activity* i só pode iniciar qualquer processo de leitura se todos os dados $d \in \Delta_{in}(i)$ já estiverem disponíveis (*i.e.*, se todos os dados $d \in (\Delta_{in}(i) \cap D_d)$ estiverem escritos).

$$\sum_{d \in D} y_{djt} W(d) \leq cm_j, \qquad \forall j \in \overline{C_o}, \forall t \in T \quad (17)$$

$$\overleftarrow{x}_{idjpt} \cdot (t + \overleftarrow{t}_{djp}) \leq z^T, \qquad \forall i \in N, \forall d \in \Delta_{out}(i), \forall j \in C_o, \forall p \in \overline{C_o}, \forall t \in T \quad (18)$$

$$v_{jt} \cdot t \leq z_j^T, \qquad \forall j \in C_o, \forall t \in T \quad (19)$$

$$\sum_{j \in C_o} c_j^{co} z_j^T + \sum_{j \in B} \sum_{l \in L_j} c_{jl}^B q_{jl} \leq c_{max} \quad (20)$$

$$\sum_{p \in \overline{C_o}} \overrightarrow{x}_{idjpt} \cdot |\Delta_{in}(i)| \leq \sum_{g \in \Delta_{in}(i)} \sum_{p \in \overline{C_o}} y_{gpt}, \qquad \forall i \in N, \forall d \in \Delta_{in}(i), \forall j \in C_o, \forall t \in T \quad (21)$$

As restrições em (22) relacionam a variável de armazenamento y (associada ao período de tempo) com a variável de armazenamento $\overline{y}$, que é independente do tempo. A restrição em (23) garante que dois arquivos de dados conflitantes não sejam colocados no mesmo dispositivo (conflito *hard*). Da mesma forma, as desigualdades em (24) garantem que se dois dados conflitantes forem armazenados no mesmo dispositivo, a variável de penalidade w equivalente seja atribuída ao valor 1 (conflito *soft*). As restrições em (25) medem o nível de exposição, *i.e.*, o risco de se executar uma *c-activity* ao considerar os requisitos de segurança. A equação (26) calcula o volume dos dados armazenados em um mesmo volume. As restrições em (27) determinam os intervalos de armazenamentos $l \in L_j$ de cada volume $j \in B$, enquanto que as restrições em (28) restringem os tamanhos de armazenamentos utilizados no volume. Por sua vez, as restrições em (27) e (29) garantem o uso adequado dos intervalos de armazenamento (b_{jl}) de acordo com o volume total dos dados (q_{jl}). Por fim, as restrições em (30) impõem que, se um intervalo $l \in L_j \backslash \{0\}$ do volume $j \in B$ estiver ativo, deve existir pelo menos algum dado associado ao volume j.

4. A Heurística `Okinawa`

A execução de *workflows* com restrições de confidencialidade em ambientes containerizados deve seguir a formalização descrita na Seção 3. Apesar da formalização apresentada poder ser utilizada para buscar o melhor escalonamento do *workflow* no ambiente conteinerizado, ela requer que todo o espaço de busca seja avaliado, o que é custoso e não-escalável. Para *workflows* reais, como o Montage [Sakellariou et al. 2009] (que será utilizado na avaliação da `Okinawa`), se faz necessária a aplicação de uma heurísticas para que o escalonamento possa ser realizado em tempo hábil. Nesta seção, apresentamos uma heurística construtiva gulosa chamada `Okinawa`, projetada para escalonar as *c-activities* de *workflows* com restrições de confidencialidade em ambientes conteinerizados.

$$y_{djt} \leq \overline{y}_{dj} \quad ,\forall d \in D, \forall j \in \overline{C_o}, \forall t \in T \quad (22)$$

$$\overline{y}_{d_1 j} + \overline{y}_{d_2 j} \leq 1 \quad ,\forall (d_1, d_2) \in E_h, \forall j \in \overline{C_o} \quad (23)$$

$$\overline{y}_{d_1 j} + \overline{y}_{d_2 j} \leq 1 + w_{d_1 d_2} \quad ,\forall (d_1, d_2) \in E_s, \forall j \in \overline{C_o} \quad (24)$$

$$l^{ri}_{task} - \sum_{j \in C_o} \sum_{t \in T} l^{rj}_{vm} x_{ijt} \leq e_{ri} \quad ,\forall i \in N, \forall r \in R \quad (25)$$

$$\sum_{l \in L_j} q_{jl} = \sum_{d \in D} \overline{y}_{dj} W(d) \quad ,\forall j \in B \quad (26)$$

$$\sum_{l \in L_j} b_{jl} = 1 \quad ,\forall j \in B \quad (27)$$

$$\sum_{d \in D} \overline{y}_{dj} W(d) \leq \sum_{l \in L_j} sm_{jl} b_{jl} \quad ,\forall j \in B \quad (28)$$

$$sm_{j(l-1)}.b_{jl} \leq q_{jl} \leq sm_{jl}.b_{jl} \quad ,\forall j \in B, \forall l \in L_j \backslash \{0\} \quad (29)$$

$$b_{jl} \leq \sum_{d \in D} \overline{y}_{dj} \quad ,\forall j \in B, \forall l \in L_j \backslash \{0\} \quad (30)$$

O Algoritmo 1 apresenta o procedimento executado pela heurística `Okinawa`. Este algoritmo garante uma solução viável respeitando a precedência entre as *c-activities*, *i.e.*, uma *c-activity* só é executada quando os seus dados de entrada estão disponíveis. O *loop* mais externo (linha 7) itera até que todas as *c-activities* do *workflow* sejam executadas em algum contêiner. O algoritmo verifica se os dados necessários estão disponíveis para executar a *c-activity* $i \in \overline{N}$ em um contêiner $j \in C_o$, usando a função $Gen()$ (linha 11). Em seguida, calcula o custo de escalonar esta *c-activity* no contêiner, utilizando a função $Calc_{OF}$. Este custo é definido pelas Equações (1 - 3). A heurística estima também os dispositivos $W \subseteq \overline{C_o}$ que devem armazenar os dados de saída da *c-activity* i (por meio da função $EstRes()$) de forma que para todo dado $d \in \Delta_{out}(i)$, o método seleciona aleatoriamente β dispositivos de $\overline{C_o}$ e, dentre eles, seleciona aquele com o menor custo na função objetivo que não viole restrições de capacidade e confidencialidade.

Algorithm 1: `Heurística Okinawa`

Data: θ_{RCL}, β
Result: escalonamento S
1 **Record {**
2 **integer** act_{index}, c_{index};
3 **List_of_Indexes** res;
4 **float** of_{value};
5 **}** $CandList$;
6 $S \leftarrow \emptyset$; $\overline{N} \leftarrow N$;
7 **while** $\overline{N} \neq \emptyset$ **do**
8 $CandList_{temp} \leftarrow$ **new**$(CandList)$; $CandList \leftarrow$ **new**$(CandList)$; $CandList_{restricted} \leftarrow$ **new**$(CandList)$;
9 **foreach** $i \in \overline{N}$ **do**
10 **foreach** $j \in C_o$ **do**
11 **if** $Gen(\Delta_{in}(i))$ **then**
12 $OF_{value} \leftarrow Calc_{OF}(S, i, j)$; $W \leftarrow EstRes(S, i, j, \beta)$;
13 $CandList_{temp}.act_{index} \leftarrow i$; $CandList_{temp}.c_{index} \leftarrow j$; $CandList_{temp}.res \leftarrow W$; $CandList_{temp}.of_{value} \leftarrow OF_{value}$;
14 $CandList \leftarrow CandList \cup CandList_{temp}$;
15 **end**
16 **end**
17 **end**
18 **sort**$(CandList)$;
19 $CandList_{restricted} \leftarrow GenRCL(CandList, \theta_{RCL})$;
20 $(i^*, j^*, W^*) \leftarrow draw(CandList_{restricted})$;
21 $S \leftarrow S \cup (i^*, j^*, W^*)$;
22 $\overline{N} \leftarrow \overline{N} \backslash \{i^*\}$;
23 **end**

O algoritmo então gera uma lista de pares *c-activity*/contêiner candidatos ($CandList$) (linha 14), ordenando-as (de forma crescente) com base no valor of_{value} da função objetivo (linha 18) e cria uma Lista de Candidatos Restrita ($CandList_{restricted}$), usando o parâmetro θ_{RCL}, contendo os melhores candidatos (linha 19). Um candidato é escolhido aleatoriamente da $CandList_{restricted}$ e adicionado à solução corrente S (linhas 20 e 21). Este processo é repetido até que todas as *c-activities* do *workflow* estejam escalonadas para uma execução. Devido à natureza estocástica da `Okinawa`, diferentes soluções são encontradas para os mesmos dados de entrada, o que justifica a execução do algoritmo 100 vezes para selecionar o melhor resultado entre todas as execuções. O código-fonte da `Okinawa` será disponibilizado no respositório institucional do nosso grupo de pesquisa no GitHub `https://github.com/UFFeScience/okinawa`.

5. Avaliação Experimental

Nesta seção avaliamos a heurística `Okinawa` com a execução de um *workflow* da área de astronomia em um ambiente conteinerizado. O Montage [Sakellariou et al. 2009] (Figura 2) é um *workflow* que cria um mosaico a partir de múltiplas fotos do céu e é intensivo produção de dados. Ele é composto por sete atividades: (i) *mProject* (em amarelo) - projeta as imagens em uma determinada escala, (ii) *mDiffFit* (em azul) - calcula e ajusta as diferenças entre duas imagens astronômicas, (iii) *mConcatFit* (em vermelho) - combina múltiplas imagens em uma, (iv) *mBgModel* (em laranja) - modela e corrige diferenças no fundo em imagens com sobreposição, (v) *mBackground* (em verde) - remove o fundo das imagens, (vi) *mImgtbl* (em cinza claro) - extrai metadados para geracão do mosaico, e (vii) *mAdd* (em cinza escuro) - cria o mosaico com a composição das imagens. O grafo de conflito do Montage foi gerado por um *script* que determinou as restrições para os dados usando as seguintes regras: (i) arquivos de entrada e saída de uma mesma *c-activity* não podem ser armazenados juntos (restrição *hard*), e (ii) arquivos de saída de *c-activities* irmãs (mesmo nível no grafo) não devem ser armazenados juntos (restrição *soft*), e caso isso ocorra, o valor 1 de penalidade é aplicado para cada conflito entre 2 arquivos.

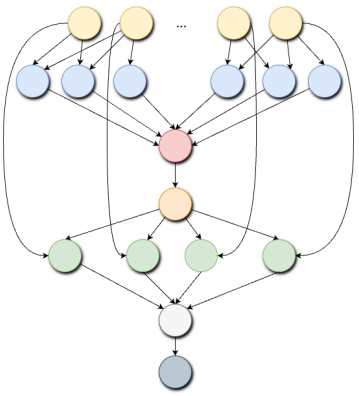

Figura 2. O Montage.

O Montage foi configurado para cobrir 0,5 graus quadrados do céu e com imagens em três canais RGB. Essa configuração do Montage gera as seguintes quantidades de *c-activities*: 12 *mProject*, 18 *mDiffFit*, 3 *mConcatFit*, 3 *mBgmodel*, 12 *mBackground*, 3 *mImgtbl* e 3 *mAdd*, totalizando 54 *c-activities*. Além disso, definimos que as *c-activities* relacionadas a atividade *mDiffFit* devem criptografar os seus dados de saída, *i.e.*, os *contêineres* que executam essas *c-activities* devem prover mecanismos de criptografia. Nos experimentos realizados, variaram-se os pesos dos objetivos, usando quatro configurações diferentes: (i) $Conf_1$ (Priorização na redução do tempo de execução) - $\alpha_t = 0.9$, $\alpha_b = 0.05$ e $\alpha_s = 0.05$, (ii) $Conf_2$ (Priorização na redução do custo financeiro) - $\alpha_t = 0.05$, $\alpha_b = 0.9$ e $\alpha_s = 0.05$, (iii) $Conf_3$ (Priorização na redução das penalidades de confidencialidade) - $\alpha_t = 0.05$, $\alpha_b = 0.05$ e $\alpha_s = 0.9$, e (iv) $Conf_4$ (Objetivos balanceados) - $\alpha_t = 0.33$, $\alpha_b = 0.33$ e $\alpha_s = 0.34$. Para os parâmetros θ_{LRC} e β, testes empíricos foram realizados e os valores $\theta_{LRC} = 0.5$ e $\beta = 4$ foram definidos.

Em relação ao ambiente, utilizamos o *Google Cloud Platform* para criar máquinas

virtuais nas quais os contêineres são instanciados durante a execução do *workflow*. A máquina virtual utilizada foi a `c3d-standard-16`, que possui 16 vCPUS AMD Genoa 2.6 GHz, 64 GB RAM, e 20 Gbps de taxa de transferência, com custo de US$ 0,908 por hora, equivalente a R$ 4,767 (R$ 0,0013 por segundo). Nessa máquina virtual foram instanciados os contêineres para execução dos experimentos. Foram definidos quatro tipos de contêineres e um tipo de volume que podem ser usados para executar as *c-activities*, conforme apresentado na Tabela 2. Além da quantidade de vCPUs, capacidade de armazenamento e total de memória RAM, a Tabela 2 informa quais tipos de contêineres oferecem (ou não) mecanismos de criptografia.

Tabela 2. Características dos contêineres e dos volumes instanciados na GCP

Tipo	# vCPU	Disco (GB)	cs_i	Rede (Gbps)	Criptografia
C_1	8	10	1.00	20	Sim
C_2	4	10	0.75	20	Não
C_3	2	10	0.50	20	Sim
C_4	1	10	0.25	20	Não
Volume	-	120	-	40	-

O Montage foi executado utilizando o *middleware* `AkôFlow` [Ferreira et al. 2024], responsável por criar os contêineres e seguir o plano de escalonamento gerado pela `Okinawa`. O *middleware* foi configurado para criar um contêiner de cada tipo e 4 volumes com as configurações apresentadas na Tabela 2. A Figura 3 apresenta os tempos de execução (média de 5 execuções) para cada configuração supracitada. Adicionalmente, comparamos os resultados com o escalonamento gerado pelo *baseline* HEFT [Topcuoglu et al. 2002], um algoritmo de escalonamento guloso que tem foco apenas na redução do tempo de execução. Na Figura 3, observamos que a execução relativa à $Conf_1$ foi de fato a mais rápida, sendo 12% mais rápida que a relativa ao HEFT. Mesmo as execuções relativas às $Conf_3$ e $Conf_4$ apresentaram bons tempos de execução, sendo 14% e 6% mais lentas que a relativa ao HEFT, respectivamente. Entretanto, como a `Okinawa` considera a dispersão de dados e a aplicação de criptografia para garantir a confidencialidade, consideramos que o *trade-off* entre confidencialidade e tempo de execução ainda se mostra vantajoso nos escalonamentos. Apenas na execução relativa à $Conf_2$ o tempo de execução 168% maior que a execução relativa à $Conf_1$. Na $Conf_2$, a `Okinawa` prioriza a utilização de contêineres menos custosos financeiramente, que possuem menos vCPUs, resultando em um aumento do tempo de execução. Entretanto, a execução relativa à $Conf_2$ se mostra útil em casos em que os contêineres são implantados em máquinas virtuais com diferentes custos financeiros, o que não era o caso.

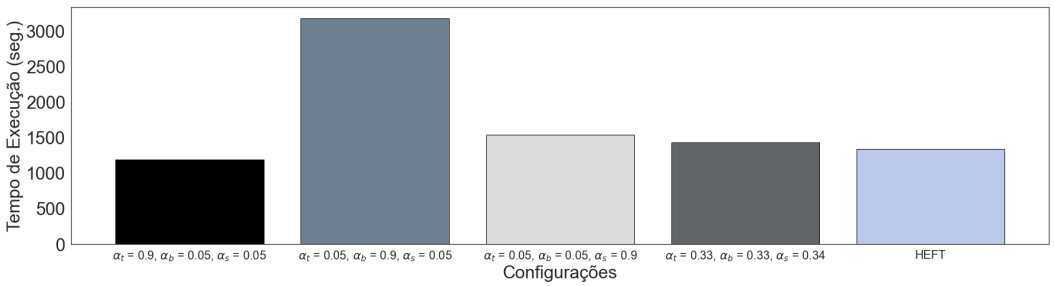

Figura 3. Tempo de execução do Montage em segundos para cada configuração

Analisamos também como a `Okinawa` realizou a dispersão dos dados nos 4 volumes e quais penalidades foram impostas a cada um dos escalonamentos. As penalidades

de confidencialidade aplicadas em cada configuração (normalizadas no intervalo [0,1]) foram: 0,0432 para a $Conf_1$, 0,0684 para a $Conf_2$, 0,0 para a $Conf_3$ e 0,0216 para a $Conf_4$. Além disso, Analisamos como o volume de arquivos foi distribuído em cada um dos 4 volumes que foram usados (Figura 4). Na $Conf_1$, $Conf_2$ e $Conf_4$ podemos perceber que há uma aleatoriedade na distribuição dos arquivos de dados nos volumes de forma a garantir a confidencialidade. Somente na $Conf_3$, que tem foco na redução da penalidade de confidencialidade, dois volumes são sempre menos utilizados, isso porque os arquivos de dados com mais restrições impostas em geral são separados dos demais nesses dois volumes (em cinza claro e cinza escuro), fazendo com que a penalidade total seja praticamente reduzida a zero.

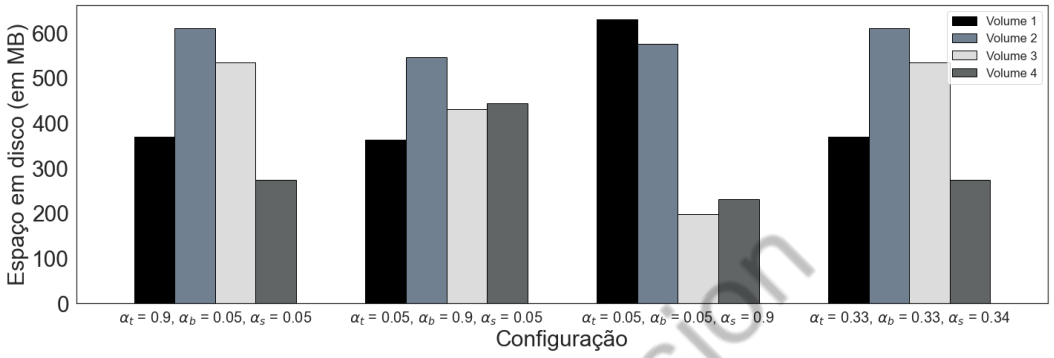

Figura 4. Espaço (em MB) dos volumes para cada configuração

6. Trabalhos Relacionados

Pelo que se tem conhecimento, não existe abordagem de escalonamento de *workflows* em ambientes conteinerizados com restrições de confidencialidade. Entretanto, diversos trabalhos têm como foco a execução de *workflows* em ambientes de nuvem minimizando o tempo de execução e o custo financeiro e considerando restrições de confidencialidade. Na categoria de dispersão de dados, [Guerine et al. 2019] propõem a dispersão dos dados produzidos por *workflows* em *buckets* na nuvem de forma a evitar acessos maliciosos. Entretanto, a dispersão de dados é desvinculada do escalonamento das *c-activities* do *workflow*, *i.e.*, os dados podem ser armazenados em um dispositivo geograficamente distante do dispositivo que irá processar os dados significando em um maior *overhead* na leitura/escrita dos dados. Trabalhos como [Shishido et al. 2018] e [Tawfeek et al. 2018] implementam soluções de escalonamento de *workflows* baseadas em algoritmos genéticos para distribuir atividades que produzem dados sensíveis em máquinas virtuais específicas na nuvem, *i.e.*, que possuem níveis ou padrões de segurança que atendam os requisitos de segurança. Similarmente, [Sujana et al. 2019] e [Abazari et al. 2019] propõem abordagens de escalonamento multicritério que consideram os requisitos de segurança do *workflow*. Apesar de considerar requisitos de segurança dos dispositivos que executam o *workflow*, esses trabalhos não consideram dispersão dos dados e assumem que todos os dados são armazenados em volumes compartilhados, o que pode significar um risco.

7. Conclusões e Trabalhos Futuros

Muitos *workflows* científicos podem se beneficiar de ambientes conteinerizados, seja por sua fácil implantação ou por sua flexibilidade. Entretanto, as plataformas disponíveis para execução de *workflows* nesses ambientes não foram projetadas para *workflows* científicos e não consideram questões de confidencialidade dos dados e da especificação do *workflow*.

Este artigo aborda a execução eficiente de *workflows* com restrições de confidencialidade em ambientes conteinerizados, propondo uma heurística que integra o escalonamento das *c-activities* com as restrições de confidencialidade e o plano de dispersão de dados. Avaliamos nossa abordagem usando o *workflow* Montage do domínio da astronomia, obtendo resultados promissores para garantir a confidencialidade e minimizar custos. Como próximo passo, planejamos evoluir a heurística `Okinawa` com a inclusão de métodos de perturbações e buscas locais, avaliando-a com vários outros *workflows* do mundo real.

Referências

Abazari, F. et al. (2019). Mows: multi-objective workflow scheduling in cloud computing based on heuristic algorithm. *Sim. Mod. Pract. and Theory.*, 93.

Abraham, S. (2023). The hpc container experience on the summit supercomputer. In *PEARC '23*, page 273–277.

Branco-Jr., E. C., Monteiro, J. M., et al. (2016). A flexible mechanism for data confidentiality in cloud database scenarios. In *ICEIS 2016*, pages 359–368.

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

Ferreira, W. et al. (2024). AkôFlow: um Middleware para execução de Workflows científicos em múltiplos ambientes conteinerizados. In *Proc. of the 39th SBBD*. SBC.

Guerine, M. A., Stockinger, M. B., et al. (2019). A provenance-based heuristic for preserving results confidentiality in cloud-based scientific workflows. *FGCS*, 97:697–713.

Javed, O. and Toor, S. (2021). An evaluation of container security vulnerability detection tools. ICCBDC '21, page 95–101.

Rosseti, I., Ocaña, K., and de Oliveira, D. (2017). Towards preserving results confidentiality in cloud-based scientific workflows. WORKS '17, New York, NY, USA. ACM.

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

Shishido, H., Estrella, J. C., et al. (2018). Multi-objective optimization for workflow scheduling under task selection policies in clouds. In *CEC*, pages 1–8.

Silva, R. et al. (2021). Análise de desempenho da distribuição de workflows científicos em nuvens com restrições de confidencialidade. In *XX WPerformance*, pages 37–48.

Sujana, J. A. J. et al. (2019). Smart pso-based secured scheduling approaches for scientific workflows in cloud computing. *Soft. Comp.*, 23(5):1745–1765.

Tawfeek, M. A. et al. (2018). Service flow management with multi-objective constraints in heterogeneous computing. In *ICCES*, pages 258–263.

Teylo, L. et al. (2017). A hybrid evolutionary algorithm for task scheduling and data assignment of data-intensive scientific workflows on clouds. *FGCS*, 76:1–17.

Topcuoglu, H. et al. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE TPDS*, 13(3):260–274.