Rumo a uma HPC Sustentável na AWS: Gestão de Clusters, Workflows Científicos Serverless e Análise de Desempenho

Márcio Castro[1], **Emilio Francesquini**[2], **Daniel Cordeiro**[3]

[1]Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

[2]Universidade Federal do ABC (UFABC)
Santo André – SP – Brasil

[3]Universidade de São Paulo (USP)
São Paulo – SP – Brasil

marcio.castro@ufsc.br, e.francesquini@ufabc.edu.br,
daniel.cordeiro@usp.br

Abstract. *This paper presents consolidated results from the research project "Sustainable High Performance Computiung on AWS", focused on the convergence of High Performance Computing (HPC) and Cloud Computing for scientific applications. The work addresses three complementary fronts: (i) fault tolerance and automated cluster management for HPC on AWS Spot Instances, (ii) serverless orchestration of scientific workflows from platform-agnostic AFCL specifications, and (iii) performance analysis of HPC application in cloud environments. Results indicate that the effectiveness of cloud HPC depends on workload-aware co-design among application, runtime and infrastructure.*

Resumo. *Este artigo apresenta resultados consolidados do projeto de pesquisa "Computação de Alto Desempenho Sustentável na AWS", com foco na convergência entre Computação de Alto Desempenho (HPC) e Computação em Nuvem para aplicações científicas. O trabalho foi desenvolvido em três frentes complementares: (i) tolerância a falhas e automação de clusters HPC em instâncias Spot da AWS, (ii) orquestração serverless de workflows científicos a partir de especificações AFCL agnósticas de provedor, e (iii) análise de desempenho de aplicações de HPC em ambientes de nuvem. Os resultados mostram que a efetividade do HPC em nuvem depende de uma abordagem de co-projeto entre aplicação, runtime e infraestrutura.*

1. Introdução

A crescente demanda computacional da pesquisa científica tem pressionado o uso de infraestruturas de Computação de Alto Desempenho (*High Performance Computing* – HPC) tradicionais, frequentemente limitadas por capacidade, elasticidade e custos de operação contínua. Nesse contexto, a Computação em Nuvem surge como alternativa para provisionamento sob demanda, permitindo alocar recursos temporários para execução de experimentos e liberar infraestrutura ao final de cada campanha.

A Computação em Nuvem pode ser entendida como um modelo de pagamento por uso, no qual recursos computacionais configuráveis são disponibilizados sob demanda,

com rápida alocação e baixo esforço operacional por parte dos usuários. Para HPC, essa característica torna a nuvem pública uma alternativa atrativa aos *clusters on-premise*, em geral mais caros e menos elásticos. Ainda assim, o ecossistema de *software* necessário para viabilizar uma plataforma de HPC em nuvem verdadeiramente sustentável permanece em evolução.

Em particular, três lacunas concretas motivaram este projeto: (i) a ausência de ferramentas abertas que automatizem o ciclo completo de gerência de clusters HPC em nuvem e tratem de forma resiliente as interrupções de instâncias preemptivas — ineficiências que elevam custos e desperdiçam recursos; (ii) a falta de mecanismos portáteis para orquestrar *workflows* científicos sobre funções *serverless* sem depender de APIs proprietárias; e (iii) a escassez de estudos sistemáticos sobre desempenho e escalabilidade de aplicações HPC em nuvem pública — conhecimento indispensável para equilibrar desempenho e consumo de recursos. Superar essas lacunas é condição necessária para que o HPC em nuvem se torne uma alternativa verdadeiramente sustentável às infraestruturas tradicionais *on-premise*.

O projeto de pesquisa intitulado *Computação de Alto Desempenho Sustentável na AWS*[1] propôs soluções para gerenciar e executar, de forma eficiente, aplicações e *workflows* científicos de HPC na AWS. Neste artigo, apresentamos três contribuições do projeto: (i) uma ferramenta para automação de clusters HPC na AWS com suporte à execução resiliente em instâncias Spot, denominada HPC@Cloud (Seção 2); (ii) uma arquitetura orientada a eventos para orquestração *serverless* de workflows científicos (Seção 3); e (iii) uma análise de desempenho de aplicações de HPC na AWS (Seção 4).

2. HPC@Cloud: Uma Ferramenta para Gerência de Clusters e Execução de Aplicações de HPC em Instâncias Spot

O HPC@Cloud é um *toolkit* de código aberto para facilitar a migração e a execução de aplicações científicas de HPC em nuvens públicas, com foco principal na AWS e arquitetura extensível para outros provedores [Munhoz and Castro 2024]. A solução foi projetada para operação via linha de comando, automatizando o ciclo completo de uso da nuvem: criação de infraestrutura, submissão de aplicações, monitoramento da execução e coleta de dados de desempenho e custo. O código-fonte está disponível publicamente[2] e combina componentes em Rust (CLI principal) e *scripts* em Python para automação e instrumentação experimental.

2.1. Visão Geral

Sua arquitetura é composta por módulos integrados e orientados a automação, conforme mostrado na Figura 1. O *Command-Line Interface* centraliza a interação do usuário; o *Resource Manager* gerencia criação, monitoramento e destruição de instâncias; o *Job Manager* executa aplicações *Message Passing Interface* (MPI) e implementa mecanismos de tolerância a falhas; e um módulo de metadados alimenta a predição de custo e tempo de execução. Essa organização desacopla a lógica de experimentação científica das particularidades de APIs de nuvem, reduzindo esforço operacional e aumentando reprodutibilidade.

[1] O projeto foi aprovado no âmbito da chamada CNPq/AWS No 64/2022 — Cloud Credits for Research e foi realizado entre os anos de 2023 e 2025.

[2] `https://github.com/lapesd/hpcac-toolkit`

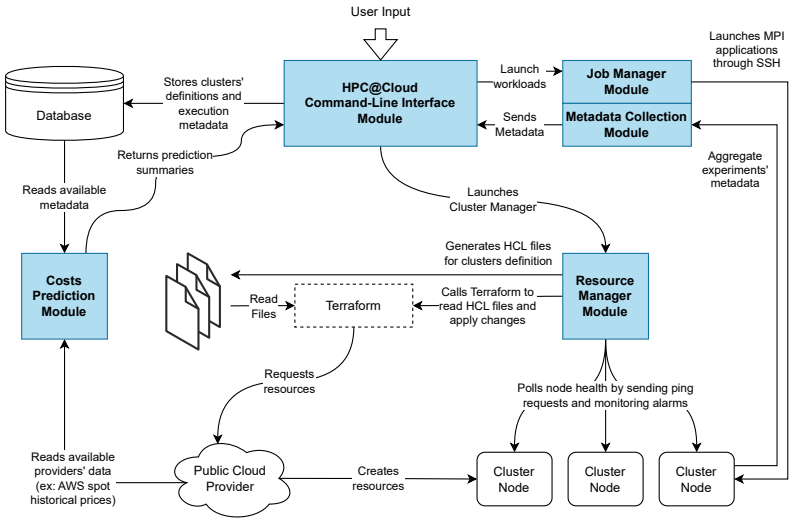


Figura 1. Visão geral da arquitetura de software do HPC@Cloud [Munhoz and Castro 2024].

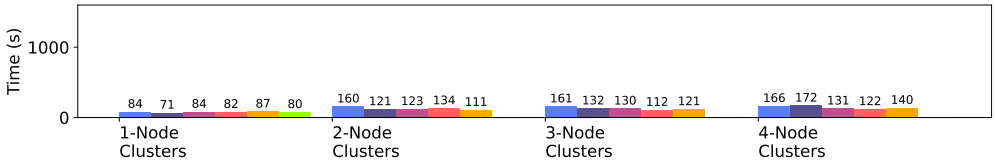
Para gerência de infraestrutura, o HPC@Cloud adota o paradigma de *Infrastructure as Code* por meio de Terraform. A partir de parâmetros de alto nível (por exemplo, número de nós, tipo de instância e armazenamento), o *toolkit* gera planos declarativos e aplica mudanças de forma idempotente, mantendo o estado do cluster consistente ao longo das execuções. Esse desenho permite reproduzir ambientes com menor intervenção manual, característica importante para campanhas experimentais de HPC com múltiplas configurações.

Um diferencial central do *toolkit* é o suporte ao ciclo de vida de instâncias Spot da AWS e à execução resiliente de workloads MPI. As *instâncias Spot* da AWS são recursos computacionais oferecidos com descontos em relação ao preço sob demanda, em troca da possibilidade de revogação do acesso pelo provedor, com aviso prévio de dois minutos, quando os recursos forem requisitados por outros clientes. Embora economicamente atrativas para workloads de HPC, essas instâncias exigem mecanismos explícitos de tolerância a falhas para garantir a continuidade das execuções.

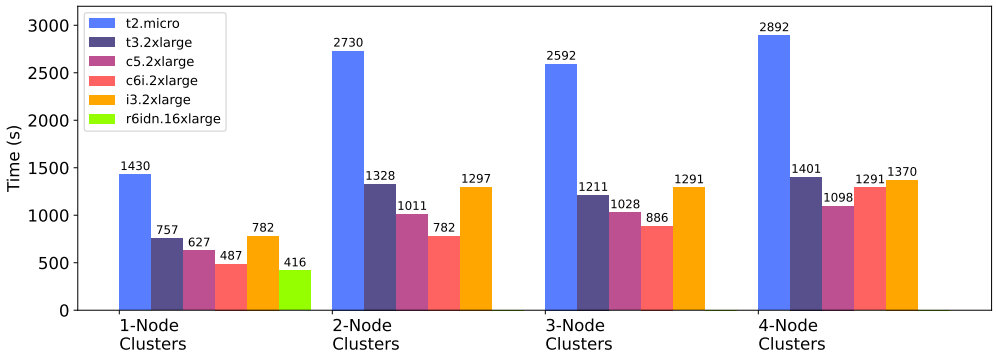
Em caso de revogação de nós, o sistema suporta estratégias de restauração bloqueante ou adaptativa. Na estratégia adaptativa, a aplicação continua em execução com redistribuição temporária de processos enquanto novos nós são solicitados, reduzindo tempo ocioso e melhorando eficiência econômica. Para tolerância a falhas, o *toolkit* integra abordagens baseadas em *checkpoint/restart* e suporte a *User-Level Failure Mitigation* (ULFM) do MPI, permitindo desde migrações sem alterar código da aplicação até estratégias adaptativas com maior desempenho em cenários de falha.

Além da execução, o HPC@Cloud coleta automaticamente metadados de infraestrutura e *workload* (configuração do *cluster*, tempo de provisionamento, tempo de execução e eventos de revogação das instâncias Spot da AWS), formando uma base para estimativas de *makespan* e custo financeiro. Com esses dados históricos, o *toolkit* se apoia em modelos de predição para orientar escolhas de configuração antes da execução completa, contribuindo para decisões mais eficientes de desempenho-custo em aplicações científicas na nuvem.

Figura 2. Resultados da eficiência de gerenciamento de recursos.



(a) Tempos de criação do *cluster*.



(b) Tempos de inicialização e configuração do sistema de arquivos compartilhado.

2.2. Resultados

Para avaliar a eficiência de gerenciamento de recursos do HPC@Cloud, medimos o tempo necessário para lançar *clusters* com diferentes tamanhos e tipos de instância na AWS. O objetivo foi isolar o custo temporal associado apenas à preparação da infraestrutura. Os experimentos consideraram *clusters* homogêneos com 1, 2, 4 e 8 nós, repetidos três vezes para cada configuração. O tempo total de preparação foi decomposto em três etapas: (i) tempo de criação das instâncias, desde a requisição até o acesso via SSH; (ii) tempo de inicialização do *cluster*, incluindo a execução dos comandos definidos no arquivo de configuração; e (iii) tempo de configuração do sistema de arquivos compartilhado. Durante a inicialização, realizamos atualização do sistema, instalação de dependências com `yum` e compilação do OpenMPI a partir do código-fonte, além de ajustes de SSH sem senha e preparação do armazenamento compartilhado.

Os testes foram executados na zona `us-east-1a`, usando Amazon Linux 2023. Foram avaliadas apenas instâncias com CPU e infraestrutura compartilhada, representativas dos tipos mais comuns em nuvens públicas e disponíveis sem contratos de longo prazo. Entre os tipos analisados, incluímos instâncias de uso geral (como `t2.micro` e `t3.2xlarge`), otimizadas para computação (`c5.2xlarge` e `c6i.2xlarge`), otimizadas para armazenamento (`i3.2xlarge`) e uma classe de maior desempenho (`r6idn.16xlarge`). Essa seleção permitiu comparar o impacto da capacidade computacional no tempo de provisionamento. A síntese dos resultados de criação e inicialização dos *clusters* é apresentada na Figura 2.

Os resultados mostram que o maior custo temporal está na execução dos comandos de inicialização, especialmente quando há compilação de dependências durante o provisionamento. Em instâncias menores, como `t2.micro`, apenas a instalação/compilação do OpenMPI pode consumir dezenas de minutos, enquanto o uso de imagem previamente

preparada reduz esse processo para poucos minutos. Instâncias mais potentes, em especial as classes otimizadas para computação e HPC, apresentaram tempos de configuração significativamente menores, embora o tempo de criação inicial da instância permaneça próximo entre diferentes categorias.

Também observamos boa escalabilidade do processo de criação: os tempos de configuração se mantiveram relativamente estáveis entre *clusters* com 2 e 8 nós, indicando que o provisionamento paralelo adotado pelo HPC@Cloud é eficiente e limitado majoritariamente pela capacidade do provedor. Em termos práticos, os experimentos reforçam que o uso de imagens pré-configuradas (AMIs/*snapshots*) é a principal estratégia para reduzir o sobrecusto de infraestrutura e custo operacional.

3. Orquestração *Serverless* de Workflows Científicos

A computação tornou-se um componente essencial da metodologia científica moderna [Cordeiro et al. 2013], e a evolução dos recursos computacionais exige a revisão contínua das técnicas empregadas nas diferentes disciplinas. Nesse contexto, a computação *serverless*, em que o provedor de nuvem gerencia dinamicamente a alocação de recursos e o usuário executa funções efêmeras cobradas por invocação, sem provisionar ou administrar servidores, surge como uma oportunidade relevante, permitindo repensar abordagens clássicas para a representação de experimentos *in silico*. Sua integração com *workflows* científicos [Atkinson et al. 2017] — modelos utilizados para orquestrar tarefas computacionais complexas — oferece ganhos potenciais de escalabilidade, eficiência de custos e melhor aproveitamento de recursos em ambientes de nuvem.

No entanto, essa integração não é trivial. A natureza *stateless* das funções *serverless* representa uma limitação importante, pois aplicações científicas costumam ser *processing-* e *data-intensive* e requer armazenamento de estados intermediários para registrar a proveniência dos dados [Davidson and Freire 2008]. Esse desafio precisa ser superado para que o paradigma *serverless* possa ser adotado de forma plena no contexto de *workflows* científicos.

3.1. Modelo Proposto

Durante o projeto, investigamos a criação de um modelo conceitual de composição e orquestração de *workflows* científicos em um ambiente *serverless*, independente da plataforma de nuvem adotada. Busca-se alcançar portabilidade na especificação dos fluxos e oferecer mecanismos de coordenação que possam ser aplicados em diferentes provedores. Os resultados preliminares foram obtidos utilizando-se a AWS a partir de experimentos especificados usando uma abstração independente de plataforma de execução denominada *Abstract Function Choreography Language* (AFCL) [Ristov et al. 2021].

O modelo proposto [Vieira and Cordeiro 2025] é tal que os *workflows* científicos podem ser mapeados para plataformas de computação em nuvem de fornecedores diferentes, utilizando-se serviços nativos para computação *serverless* oferecidos pelas plataformas. Por exemplo, a Tabela 1 mostra a equivalência de serviços entre AWS e Azure. A arquitetura proposta é composta pelos seguintes elementos principais:

- **Função Lambda coordenadora:** responsável por interpretar a especificação AFCL, acionar as funções Lambda correspondentes às tarefas e gerenciar o avanço

Tabela 1. Serviços para computação *serverless* equivalentes oferecidos por AWS e Azure.

Categoria	AWS	Azure
Function as a Service (FaaS)	AWS Lambda	Azure Functions
Banco de dados chave–valor	Amazon DynamoDB	Azure Cosmos DB
Change Data Capture (CDC)	DynamoDB Streams	Cosmos DB Change Feed
Barramento / roteamento de eventos	Amazon EventBridge	Azure Event Grid

do *workflow* com base em alterações de estado. Essa função dá suporte a diferentes construções da AFCL, como sequências, condicionais (`if`, `switch`), processamento paralelo e laços de repetição (`for`, `while`, `parallelFor`);
- **Funções Lambda de tarefas:** tratam-se de funções independentes, *stateless* e assíncronas, representando as tarefas individuais do *workflow* como caixas-pretas. Seus resultados são capturados no nível do serviço, assegurando baixo acoplamento e orquestração reativa;
- **Tabela de estado no DynamoDB:** constitui o repositório persistente utilizado para armazenar informações de execução, resultados das tarefas, contadores de processamento paralelo e dados de iterações em laços repetitivos. Essa tabela permite que a função coordenadora monitore e mantenha o progresso do *workflow* entre invocações sucessivas;
- **DynamoDB Streams:** atua como um mecanismo de disparo que aciona a função coordenadora sempre que novas atualizações de tarefa são registradas no DynamoDB. Esse arranjo promove uma orquestração dinâmica e eficiente, baseada na conclusão das tarefas do *workflow*.

Os componentes descritos operam de maneira integrada, formando uma máquina de estados eficiente, orientada a eventos e sem a necessidade de execução contínua. A orquestração é realizada por meio de invocações curtas das funções Lambda, desencadeadas em resposta a alterações no estado do sistema. Tal abordagem promove escalabilidade, permitindo que múltiplas instâncias de funções coordenadoras sejam executadas simultaneamente em cenários com eventos concorrentes. O mecanismo de dimensionamento automático do DynamoDB gerencia de forma eficaz altas taxas de leitura e escrita, caracterizando um ambiente plenamente serverless e escalável, no qual a progressão do workflow é dirigida pelas atualizações dos dados.

3.2. Desafios Principais da Arquitetura

A arquitetura proposta enfrenta desafios centrais na orquestração de *workflows* científicos em plataformas de computação *serverless*. O primeiro é a **análise e modelagem**, que envolve interpretar a linguagem AFCL e representar sua estrutura de *workflow* em uma linguagem de programação adequada. A **invocação de funções e troca de dados** demanda o correto roteamento entre funções *serverless*, tratadas como *caixas-pretas*, de modo que as saídas de cada função sejam repassadas adequadamente às próximas etapas. Em relação às **ramificações e laços**, a arquitetura deve garantir a execução apenas das funções correspondentes às condições da AFCL, além de controlar iterações até o cumprimento de uma condição de parada. O **paralelismo e a sincronização** exigem a execução simultânea de múltiplas funções e a agregação dos resultados antes da continuidade do

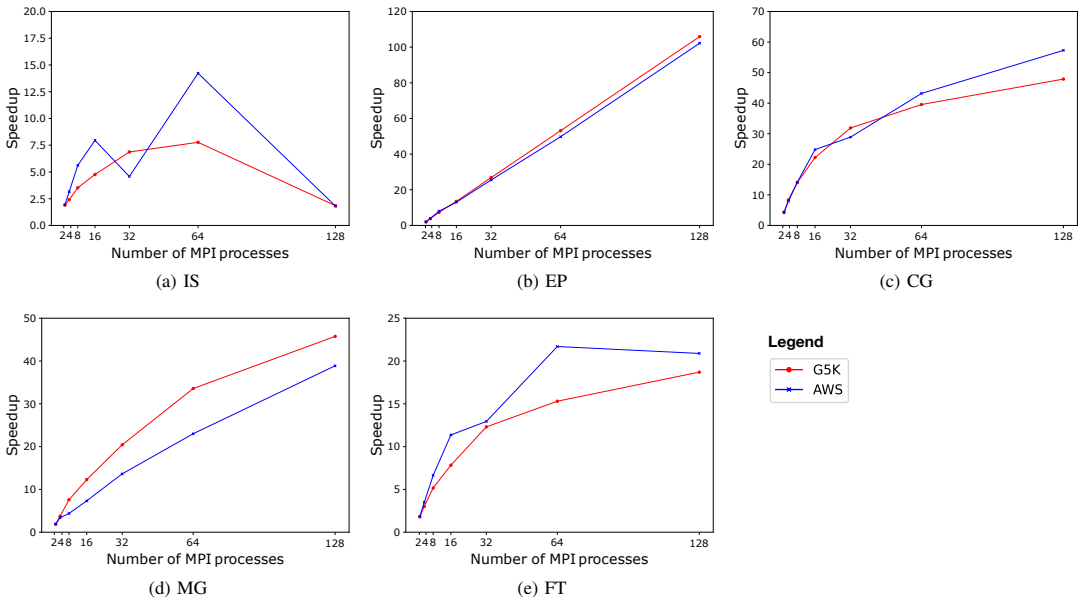


Figura 3. Comparação de desempenho e escalabilidade dos *kernels* NPB entre AWS e Grid'5000.

fluxo, assegurando consistência. Por fim, a **escalabilidade nos limites do FaaS** requer orquestração orientada a eventos e invocações *stateless*, garantindo execução gradativa, eficiente e compatível com as restrições de tempo e memória do modelo.

4. Análise de Desempenho de Aplicações de HPC na AWS

Para melhor entender o comportamento de aplicações de HPC na AWS, elaboramos um estudo experimental de escalabilidade com aplicações extraídas do conjunto *NAS Parallel Benchmarks* (NPB) [Bailey et al. 1991], comparando duas plataformas: um *cluster* virtualizado na AWS e um *cluster* tradicional *on-premise* da infraestrutura Grid'5000 (G5K). O NPB é um conjunto de *benchmarks* paralelos proposto pela divisão *NASA Advanced Supercomputing*, amplamente utilizado para avaliar desempenho computacional em sistemas paralelos. Para os experimentos, adotamos a classe C de entrada. Para garantir comparabilidade entre plataformas, selecionamos configurações de *hardware* semelhantes dentro das opções disponíveis. No G5K, utilizamos nós do *cluster Dahu*[3]; na AWS, utilizamos instâncias *Spot* `c5n.4xlarge` (16 vCPUs Intel Xeon Platinum série 8000 de 3,5 GHz, 42 GiB de RAM) do EC2. Avaliamos *clusters* com 1, 2, 4 e 8 nós, variando o número de processos MPI por nó em $\{1, 2, 4, 8, 16\}$. O número total de processos variou até 128, e para cada valor foi considerada a melhor combinação nó/processo em cada plataforma. Esse procedimento evita vieses de configuração e permite comparar o melhor resultado obtido em cada infraestrutura.

Os resultados por *kernel* mostram comportamento dependente do perfil de comunicação: no IS, a AWS obteve *speedup* superior na maior parte das configurações (com ganho próximo do dobro em 64 processos), embora ambas as plataformas tenham perdido escalabilidade em 128 processos; no EP, que possui característica *CPU-bound*,

[3]Nós com processadores Intel Xeon Gold 6130 de 2,1 GHz, 16 cores/CPU e 192 GiB de RAM. Veja a descrição dos nós em `https://www.grid5000.fr/w/Hardware`.

o desempenho foi praticamente linear e muito próximo entre plataformas, com vantagem marginal do G5K (inferior a 4%); no CG, os resultados foram semelhantes, com vantagem da AWS em cenários mais paralelos; e, entre os casos com comunicação coletiva, o MG favoreceu o G5K (maior intensidade e tamanho de mensagens), enquanto o FT manteve a AWS competitiva. Em conjunto, os dados indicam que a AWS pode ser eficiente para HPC, desde que a seleção de instâncias e a configuração de execução sejam ajustadas ao padrão de comunicação da aplicação.

De forma geral, os resultados mostram que aplicações limitadas por CPU apresentam comportamento semelhante entre as duas plataformas, enquanto aplicações com comunicação intensiva são mais sensíveis às características da rede. Em cenários com comunicação moderada e mensagens de tamanho intermediário, a AWS apresentou desempenho competitivo e, em alguns casos, até superior.

5. Conclusão

Este artigo apresentou uma visão integrada de alguns resultados atingidos durante a execução do projeto *Computação de Alto Desempenho Sustentável na AWS* financiado pelo CNPq/AWS, evidenciando que a convergência entre HPC e Computação em Nuvem depende de soluções combinadas de automação de infraestrutura, tolerância a falhas, orquestração *serverless* e análise de desempenho. Os resultados discutidos mostram que a AWS pode oferecer desempenho competitivo para aplicações científicas, especialmente quando há alinhamento entre perfil de comunicação da aplicação, configuração de instâncias e estratégia de execução. Em conjunto, as contribuições reforçam o papel de ferramentas abertas e de abordagens orientadas a dados para tornar o uso de nuvem mais eficiente, reprodutível e adequado às demandas da pesquisa científica.

Como trabalhos futuros, propomos avançar para um cenário de HPC híbrido que integre, de forma coordenada, *clusters on-premise* com recursos da AWS. Em nível de *runtime*, propor soluções para a execução híbrida (*on-premise* + AWS) de aplicações de HPC com foco em políticas de escalonamento híbrido, orquestração de dados e resiliência via *checkpoint/restart*. Em nível de escalonador de filas, planejamos evoluir o ecossistema para suportar *HPC Spot Jobs*, habilitando maleabilidade, *cloud bursting* e tratamento de preempções em instâncias Spot da AWS sem perda significativa do progresso da execução das aplicações de HPC.

Agradecimentos

Os autores agradecem ao CNPq e à parceria CNPq/AWS pelo suporte ao projeto processo 421787/2022-8, às instituições parceiras nacionais e internacionais e às pessoas estudantes e pesquisadoras envolvidas no desenvolvimento das atividades. Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP procs. 19/26702-8 e 23/00811-0) e pelo Centro de Ciência para o Desenvolvimento (CCD) 'Cidades Carbono Neutro' (FAPESP 24/01115-0).

Declaração Sobre uso de Inteligência Artificial

Ferramentas de Inteligência Artificial foram utilizadas exclusivamente para correção ortográfica do manuscrito, sem geração de conteúdo técnico-científico.

Referências

Atkinson, M., Gesing, S., Montagnat, J., and Taylor, I. (2017). Scientific workflows: Past, present and future.

Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Dagum, L., Fatoohi, R. A., Fineberg, S., Frederickson, P. O., Lasinski, T. A., Schreiber, R. S., Simon, H. D., Venkatakrishnan, V., and Weeratunga, S. K. (1991). The nas parallel benchmarks. Technical Report RNR-91-002, NASA Advanced Supercomputing Division.

Cordeiro, D., R. Braghetto, K., Goldman, A., and Kon, F. (2013). Da ciência à e-ciência: paradigmas da descoberta de conhecimento. *Revista USP*, 97:71–80.

Davidson, S. B. and Freire, J. (2008). Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1345–1350.

Munhoz, V. and Castro, M. (2024). Enabling the execution of hpc applications on public clouds with hpc@cloud toolkit. *Concurrency and Computation: Practice and Experience*.

Ristov, S., Pedratscher, S., and Fahringer, T. (2021). AFCL: An abstract function choreography language for serverless workflow specification. *Future Generation Computer Systems*, 114:368–382.

Vieira, A. and Cordeiro, D. (2025). Proposta de modelo agnóstico para composição e orquestração serverless de workflows científicos. In *Escola Regional de Alto Desempenho de São Paulo (ERAD-SP)*, pages 102–105. SBC.