Oferta ou Roubo: avaliando abordagens de escalonamento dinâmico para Workflow scRNA-seq na nuvem AWS *

Erik Izidoro[1], **Cristina Boeres**[1], **Vinod E.F. Rebello**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brazil

erikizidoro@id.uff.br, {boeres,vinod}@ic.uff.br

Abstract. *Managing the execution of scientific workflows through dynamic scheduling is crucial, especially when task durations are unpredictable and achieving efficient performance while minimizing costs associated with cloud usage is a key goal. As scRNA-seq workflows typically preprocess Gigabytes of data files, this work compares two approaches to parallelize this time-consuming step, identifying advantages of adopting a Work Stealing (WS) or Master-Worker (MW) paradigm to distribute data processing tasks. Experiments conducted on cloud instances with different numbers of CPUs and threads yield results that show the MW approach is competitive in terms of performance, with execution times as much as 28.4% shorter in most of the tested configurations.*

Resumo. *Gerenciar a execução de workflows científicos através do escalonamento dinâmico é crucial para alcançar desempenho e minimizar custos associados na nuvem. Uma vez que os workflows scRNA-seq tipicamente pré-processam Gigabytes de dados, este trabalho compara duas abordagens para paralelizar este passo demorado, identificando as vantagens da adoção de um paradigma Work Stealing (WS) ou Master-Worker (MW) para distribuir as tarefas de processamento. Experimentos conduzidos em instâncias da nuvem com diferentes números de CPUs e threads resultaram em dados que mostram que a abordagem MW é competitiva em termos de desempenho, com tempos de execução até 28,4% mais curtos na maioria das configurações testadas.*

1. Introdução

Entender as diferenças entre os tipos de células em sistemas biológicos é crucial para a compreensão de seus papéis, especialmente durante o desenvolvimento e a progressão de doenças. O sequenciamento de célula única (scRNA-seq) permite que pesquisadores explorem a heterogeneidade celular dentro dos tecidos e pode ajudar a identificar populações celulares raras que passam despercebidas. O *workflow* CellHeap, proposto por pesquisadores da Fiocruz [Silva et al. 2021], é uma plataforma modular voltada para o processamento de dados de scRNA-seq em larga escala, estruturada em cinco fases principais: curadoria das amostras, pré-processamento, controle de qualidade, análise de agrupamento celular e análises avançadas em nível de célula e de gene. Cada fase integra ferramentas consolidadas da bioinformática, permitindo desde a conversão e mapeamento das sequências até a identificação de populações celulares e vias biológicas relevantes.

*Financiado pelo projeto CNPq/AWS (processo 421828/2022-6).

Uma vez que o CellHeap foi projetado para ser flexível e portátil, torna-se possível executá-lo em diferentes ambientes, incluindo plataformas de nuvem.

Apesar de sua versatilidade, a execução de *workflows* dessa natureza envolve custos expressivos de tempo de processamento, uso de memória e espaço de armazenamento. No trabalho que descreve o CellHeap [Silva et al. 2021], a execução sequencial deste workflow para apenas 11 amostras de scRNA-seq levou cerca de 294 horas (aproximadamente 12 dias), demandando centenas de gigabytes de memória e até dezenas de vezes o tamanho original dos dados em disco. Esses custos se tornam ainda mais relevantes quando se considera que bancos de dados públicos já armazenam centenas de terabytes de dados, com a previsão de alcançar petabytes nos próximos anos. Neste cenário, a infraestrutura da nuvem acaba por fornecer uma base bem estabelecida para sustentar a execução de aplicações que demandam alto poder computacional, disponibilidade de memória ou armazenamento. Assim, a importância de estudar formas de explorar os recursos oferecidos pela nuvem para atender às necessidades de tais aplicações se torna inquestionável.

Diante desse contexto, este trabalho tem como objetivo investigar estratégias de escalonamento dinâmico das tarefas da primeira e mais demorada fase do workflow CellHeap, dedicada à seleção, validação e preparação dos dados brutos de scRNA-seq, utilizando recursos da AWS Elastic Compute Cloud (EC2), em especial o serviço *Parallel Cluster*. O foco está em explorar diferentes políticas de escalonamento dinâmico que permitam otimizar o uso dos recursos disponíveis, reduzindo tempos de execução e, consequentemente, custos monetários.

Para isso, foi utilizada a ferramenta de orquestração já proposta em [Silva et al. 2024] para o CellHeap, cuja abordagem principal de escalonamento dinâmico é baseada no paradigma de *Work Stealing* (WS). Neste trabalho, é proposta e avaliada uma abordagem alternativa, fundamentada no modelo Mestre-Trabalhador (MW, do inglês *Master-Worker*), de forma a comparar o impacto de cada estratégia no desempenho e no uso eficiente da infraestrutura de nuvem.

2. Descrição das Abordagens Analisadas

[Silva et al. 2024] propôs um gerenciador de execução, baseado em OpenMP/MPI, de múltiplos fluxos lineares (aqui, simplesmente chamados tarefas) onde a realocação dinâmica de tarefas por WS foi comparada à execução de uma alocação estática de tarefas sem WS. A necessidade de escalonamento dinâmico se faz necessária devido ao fato dos tempos de execução das tarefas do CellHeap não serem conhecidos *a priori*. O mecanismo WS se baseia no roubo de tarefas esperando a serem executadas em outras CPUs. Com a especificação de um conjunto de *threads* por nó do *cluster*, cada *thread*, ao ficar ociosa, pode roubar carga (*uma tarefa*) de outra *thread* dentro de um mesmo nó ou, se necessário, de outros nós. Como proposta do presente trabalho, motivada pela percepção da possibilidade de estruturar o escalonamento dinâmico de forma a aproveitar melhor os recursos e minimizar as sobrecargas do próprio escalonador, foi desenvolvida uma nova versão do gerenciador embasada na abordagem Mestre e Trabalhador (MW), também em OpenMP/MPI.

Nesta abordagem MW, uma das *threads* é designada como *Mestre* e gerencia a distribuição das tarefas entre as demais *threads* trabalhadoras, sendo que cada trabalha-

dor pode estar tanto no mesmo nó do *Mestre* quanto nos demais nós do *cluster*. Para possibilitar esta distribuição, o nó Mestre tem seu trabalho dividido em duas fases: (i) a fase de organização dos dados de entrada e (ii) a fase de distribuição de tarefas. A fase (i) é idêntica à abordagem WS do gerenciador apresentada em [Silva et al. 2024], onde são recebidas como entrada a lista de amostras (ordenado por tamanho do arquivo) e a definição do trabalho que uma tarefa vai realizar para processar cada amostra. Na fase (ii), enquanto a abordagem WS já distribui todas as tarefas entre as *threads* de antemão, a MW segue uma abordagem de escalonamento dinâmico, se baseando na comunicação entre as *threads* trabalhadoras e a *thread Mestre*. As *threads* trabalhadoras do nó Mestre utilizam *flags* (no mesmo espaço de memória) para sinalizar à *thread Mestre* o seu estado atual. Já as *threads* trabalhadoras de outros nós utilizam MPI para enviar pedidos de tarefas ao nó Mestre. Se uma dessas *threads* estiver disponível para receber trabalho, a *thread Mestre* retira uma tarefa da fila e a envia para a thread que solicitou o trabalho. Caso não haja tarefas na fila, a *thread* trabalhadora recebe uma sinalização de término da *thread Mestre*. Esse processo de distribuição de tarefas se repete até que todas as tarefas da fila sejam enviadas e todas as *threads* trabalhadoras tenham recebido esta sinalização.

As abordagens usufruem da infraestrutura da AWS em suas execuções, dependendo do serviço *Parallel Cluster* [Amazon Web Services 2025]. Este serviço possibilita a criação de *clusters* de instâncias de Máquinas Virtuais, disponibilizadas pelo *Elastic Cloud Computing* (EC2). Para implantar um *cluster*, são especificados a quantidade de nós e o tipo de instância que comporá esses nós. Para facilitar o acesso (leitura e escrita) simultâneo a grandes quantidades de dados por parte das tarefas executando em paralelo, o serviço *FSx for Lustre* é utilizado, disponibilizando um sistema de arquivos de alto desempenho compartilhado entre os nós do *cluster*.

3. Análise Experimental

Para conduzir o experimento com análise comparativa, foram definidos ambientes de execução com configurações variadas, a partir do *Parallel Cluster*, que possibilitou o levantamento de *clusters* com 2, 4, 8 e 16 nós. Cada nó foi composto pela instância *m6idn.2xlarge*, com processador Intel Xeon 8375C com 8 vCPUs, 32 GiB de memória RAM, 437 GB de armazenamento SSD e até 40 Gbps de banda larga de rede. Em relação ao número de *threads* por nó (instância do *cluster*), foram avaliadas configurações com 4 (o número de CPUs físicos) e 5 *threads*, valores usados no [Silva et al. 2024].

O experimento utilizou a primeira fase do *workflow* CellHeap [Silva et al. 2021], que consiste, basicamente, em uma etapa de pré-processamento de dados. Os dados fornecidos para cada tarefa foram obtidos a partir do dataset PRJNA743046, disponível no *National Center for Biotechnology Information* [NCBI]. Foram selecionadas 400 amostras, cujos tamanhos se distribuem da seguinte forma: 300 amostras que possuem de 10 a 100 MB, 95 amostras que possuem de 100 a 110 MB e as 5 amostras restantes que possuem de 190 a 200 MB. A Tabela 1 reporta os tempos de execução das abordagens de escalonamento MW (T_{MW}) e WS (T_{WS}) em segundos. Ainda, foi calculado o percentual de ganho de tempo de MW sobre WS da seguinte forma: $Ganho\,\% = \frac{T_{WS} - T_{MW}}{T_{WS}} \times 100\%$.

Nas duas abordagens, é observado que, quanto mais *threads* e mais nós são definidos, melhor o tempo de execução. Praticamente em todas as configurações, MW produz melhores resultados do que WS, a não ser quando o número de nós é muito pequeno. A

razão para esses resultados se dá devido à simplicidade da abordagem MW, que apresenta um mecanismo centralizado para coordenar a distribuição das tarefas, levando a menores *sobrecargas* de escalonamento. Apesar de descentralizada, a abordagem WS depende de mecanismos de controle mais complexos que podem impactar nos tempos de execução.

Nº de Nós	Nº de Threads por Nó	T_{MW}	T_{WS}	Ganho %
2	4	2714s	2404s	-12,89%
2	5	2652s	2408s	-10,13%
4	4	477s	599s	20,36%
4	5	425s	456s	6,79%
8	4	235s	308s	23,70%
8	5	210s	246s	14,63%
16	4	121s	169s	28,40%
16	5	108s	137s	21,16%

Tabela 1. Tempos de execução (em segundos) das abordagens MW e WS, variando o número de *threads* e nós do *cluster*

4. Conclusão e Análises Futuras

Ambas as abordagens demonstram desempenho semelhante quando estruturadas nas mesmas configurações. Todavia, a abordagem MW proposta neste trabalho se mostra competitiva com a abordagem embasada em WS, apresentando tempos de execução menores na maioria dos casos estudados devido principalmente à menor sobrecarga de escalonamento. A partir dos resultados obtidos, é aberto espaço para a realização de análises futuras. Como exemplo, pode-se citar a exploração da configurabilidade dos serviços da AWS envolvidos nas abordagens do gerenciador; a possibilidade de incorporação e análise do serviço *Simple Queue Service* (SQS) na abordagem MW a fim de aproveitar a infraestrutura da nuvem para administrar a fila de tarefas e, por fim, a análise da eficiência das abordagens do gerenciador (ainda considerando a primeira fase do CellHeap como *workflow* de execução) frente a datasets de maior tamanho.

Referências

Amazon Web Services (2025). AWS Parallel Cluster. `https://docs.aws.amazon.com/parallelcluster/`. Acessado em: 20-05-2025.

NCBI. BioProject: PRJNA743046. `https://www.ncbi.nlm.nih.gov/Traces/study/?acc=PRJNA743046`. Acessado em: 10-09-2025.

Silva, H. S., Castro, M. C., Silva, F. A., and Melo, A. C. (2024). A framework for automated parallel execution of scientific multi-workflow applications in the cloud with work stealing. In *European Conference on Parallel Processing*, pages 298–311. Springer.

Silva, V. et al. (2021). CellHeap: A workflow for optimizing COVID-19 single-cell RNA-Seq data processing in the Santos Dumont supercomputer. In *Brazilian Symposium on Bioinformatics*, pages 41–52. Springer.