CellHeap: Instâncias maiores podem não ser melhores*

Bernardo M. Rebello[1], Lucas P. Avelar[1], Cristina Boeres[1], Vinod E.F. Rebello[1]

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

{mendes_bernardo,lucasavelar}@id.uff.br,{boeres,vinod}@ic.uff.br

Abstract. *Single-cell RNA sequencing (scRNA-seq) has gained popularity recently due to new technologies that have made it scalable and affordable, the emergence of new software that enables its analysis, and the discovery of new biological insights that have rendered it indispensable. However, the sheer amount of data involved makes scRNA-seq computationally very expensive and time-consuming. This work analyzes one phase of CellHeap, an scRNA-seq workflow, with the goal of making it more computationally efficient.*

Resumo. *O sequenciamento de RNA de célula única (scRNA-seq) ganhou popularidade recentemente devido às novas tecnologias que o tornaram escalável e acessível, ao surgimento de novos softwares que permitem sua análise e à descoberta de novos insights biológicos que o tornaram indispensável. Porém, devido à enorme quantidade de dados envolvida, o scRNA-seq é computacionalmente muito custoso e demorado. Este trabalho analisa uma fase do CellHeap, um* workflow *de scRNA-seq, com o objetivo de torná-la computacionalmente mais eficiente.*

1. Introdução

Cada órgão do nosso corpo é composto por muitos tipos diferentes de células, cada uma com sua função distinta. Com o objetivo de entender as interações que formam os sistemas biológicos, cientistas analisam genes em organismos, buscando compreender e modificar o comportamento celular por meio de abordagens moleculares direcionadas. Os métodos tradicionais de estudo celular fazem a média de toda a sua atividade, o que oculta as diferenças. O sequenciamento de RNA de célula única (scRNA-seq) permite que os cientistas observem as atividades de células individuais, o que pode, por exemplo, ajudar a identificar as células raras que causam a resistência dos tumores ao tratamento e mostrar como diferentes células imunológicas respondem, orientando o desenvolvimento de vacinas e novas terapias, além de impulsionar outras descobertas e inovações na biomedicina.

O fato de que o sequenciamento de célula única (scRNA-seq), ao permitir que pesquisadores explorem a heterogeneidade celular dentro dos tecidos, contribuiu significativamente para o recente aumento nos estudos de scRNA-seq. Esse crescimento levou à atualização de bancos de dados existentes e à criação de novos, voltados para a curadoria e para a exploração de dados unicelulares. A disponibilidade de amplos dados de scRNA-seq publicamente acessíveis facilita o desenvolvimento e a pesquisa dos biólogos. No entanto, a complexidade computacional desses estudos exige conhecimento em ferramentas de bioinformática bem como na utilização da computação de alto desempenho.

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

A adoção de fluxos de trabalho ou *workflows* científicos é a principal ferramenta usada para descrever o processo para atingir um objetivo científico, geralmente expresso em termos de tarefas e suas dependências. Normalmente, as tarefas do fluxo de trabalho envolvem etapas computacionais para simulações científicas ou análise de dados. Proposto pela Fiocruz, CellHeap [Silva et al. 2024] é um *workflow* flexível, portátil e robusto para analisar grandes conjuntos de dados scRNA-seq, com controle de qualidade em todas as fases de execução e capaz de aproveitar plataformas que suportam dados em larga escala, como supercomputadores ou nuvens.

O scRNA-seq é computacionalmente exigente devido ao grande número de células, matrizes esparsas, porém de alta dimensão, e algoritmos intensivos para agrupamentos e análises de trajetória. Com um acesso limitado a supercomputadores nos últimos anos, cada vez mais a computação em nuvem tem se tornado uma alternativa acessível. Nesse contexto, frameworks buscam implementar workflows científicos para tornar a utilização de tais recursos caros mais eficiente. A contribuição deste trabalho é uma análise do comportamento de uma versão (*standard mode* – sem workstealing) do framework distribuído, proposto por [Silva et al. 2024], que acelera a primeira fase do workflow CellHeap usando serviços de nuvem da AWS EC2. O objetivo é avaliar a eficiência desta solução proposta e se melhorias podem ser encontradas.

2. CellHeap – Fase 1

O *workflow* CellHeap é composto por 5 fases [Castro et al. 2023]. A primeira fase consiste na curadoria das amostras: existem inúmeros repositórios públicos contendo petabytes de dados de scRNA-Seq, cada um contendo centenas de estudos com milhares de amostras. Dada essa grande quantidade, ferramentas como o *SRA toolkit* auxiliam a baixar as amostras de interesse, sendo essas ferramentas também usadas para validar o conjunto de dados.

O framework do [Silva et al. 2024], que implementa a primeira fase, realiza uma etapa de pré-processamento de dados. Recebendo uma lista de amostras de sequência, fornecida pelo cientista e comumente obtidas do repositório de dados genômicos, o *Sequence Read Archive* (SRA) do *National Center for Biotechnology Information* (NCBI) [National Library of Medicine], o framework aloca cada amostra a uma *thread* que executa os 3 comandos a seguir, sendo os dois primeiros programas disponibilizados pela ferramenta *sra-toolkit* do NCBI:

- **`vdb-validate`**: realiza a validação da integridade da amostra de entrada.
- **`fastq-dump`**: utilizado para converter arquivos de formato SRA para arquivos fastq, formato para armazenar sequências biológicas e suas qualidades.
- **`gzip`**: utilizado para comprimir os arquivos fastq para serem lidos por software da Fase 2.

A execução do framework é realizada em um ambiente da nuvem AWS com o serviço ParallelCluster [Amazon Web Services 2025], que auxilia na implantação, na configuração e no gerenciamento de um *cluster* com recursos HPC. O trabalho realizado por cada thread é determinado por um nó único, nomeado *head node*; as execuções dos comandos ocorrem em múltiplos nós chamados *worker nodes*. Visto que o *head node* não realiza nenhuma execução que exija uma capacidade computacional intensa, não é necessário que ele seja do mesmo tipo de instância que os *worker nodes*.

3. Resultados e Análises

O experimento apresentado nesta seção com o *framework* do [Silva et al. 2024] foi realizado usando o mesmo tipo de instância de nuvem utilizada nesse artigo (uma *m6idn.xlarge* da AWS EC2), onde se mostrou um speedup de 1,45 usando um *worker node* de 4 vCPUs, 16 GiB de RAM e um SSD local de 237 GB, e um speedup de 2,8 com dois nós. O SSD torna-se necessário para armazenar localmente os arquivos temporários em um volume efêmero chamado *scratch*. O sistema de arquivo utilizado foi o *Amazon FSx for Lustre* para facilitar a leitura e a escrita dos arquivos do fluxo em paralelo, assim como no artigo original. Devido aos speedups abaixo do esperado, escolhemos comparar a execução usando uma instância com metade da capacidade, uma *m6idn.large* com 2 vCPUs, 8 GiB de RAM e um SSD de 118 GB.

Um objetivo é tentar alcançar a quantidade ideal de *threads* para que o tempo de execução seja minimizado. O algoritmo de escalonamento do framework previamente aloca cada sequência a um nó de forma intercalada; essa ordem não é alterada durante a execução [Silva et al. 2024]. Em virtude do espaço limitado do SSD e do tamanho dos arquivos temporários gerados durante o processo, optamos por realizar os testes com uma fração das amostras utilizadas no artigo, com a finalidade de realizar uma análise inicial.

Sequências	Tamanho	m6idn.xlarge	m6idn.large
SRR12570722	591,90 MiB	279,69s	285,51s
SRR12570159	98,40 MiB	48,08s	48,58s
SRR11783965	89,80 MiB	49,51s	49,82s
SRR11783964	131,40 MiB	74,19s	74,30s
SRR11783961	209,70 MiB	119,17s	119,15s
SRR1553580	114,00 MiB	37,05s	37,17s
SRR11409384	31,60 MiB	11,64s	11,68s
SRR19484251	43,10 MiB	16,17s	16,09s
SRR23922989	55,40 MiB	21,77s	21,87s

Tabela 1. Tamanhos das sequências e os tempos de execução individuais nos tipos da instância

Sequências	m6idn.xlarge 4 *threads*	Tempo	m6idn.xlarge 2 *threads*	Tempo	m6idn.large 4 *threads*	Tempo	m6idn.large 2 *threads*	Tempo
SRR12570722	(*Thread* 1)	373,59s	(*Thread* 1)	312,64s	(*Thread* 1)	382,66s	(*Thread* 1)	312,21s
SRR12570159	(*Thread* 1)	102,51s	(*Thread* 1)	66,80s	(*Thread* 1)	103,64s	(*Thread* 1)	66,67s
SRR11783965	(*Thread* 2)	114,37s	(*Thread* 2)	70,45s	(*Thread* 2)	110,07s	(*Thread* 2)	70,30s
SRR11783964	(*Thread* 2)	129,27s	(*Thread* 2)	102,19s	(*Thread* 2)	131,82s	(*Thread* 2)	102,27s
SRR11783961	(*Thread* 3)	207,65s	(*Thread* 1)	431,13s	(*Thread* 3)	207,20s	(*Thread* 1)	430,73s
SRR1553580	(*Thread* 3)	90,71s	(*Thread* 1)	116,23s	(*Thread* 3)	88,33s	(*Thread* 1)	116,24s
SRR11409384	(*Thread* 4)	30,91s	(*Thread* 2)	85,98s	(*Thread* 4)	31,93s	(*Thread* 2)	85,75s
SRR19484251	(*Thread* 4)	43,47s	(*Thread* 2)	121,97s	(*Thread* 4)	44,58s	(*Thread* 2)	122,06s
SRR23922989	(*Thread* 1)	395,30s	(*Thread* 1)	452,94s	(*Thread* 1)	404,58s	(*Thread* 1)	452,57s
Tempo Total		398,27s		456,17s		405,67s		453,66s

Tabela 2. Os tempos individuais totais de execução das amostras, nas *threads* que foram escalonados pelo framework, do início da execução até seu término. A linha cinza representa que a sequência foi alocada para o nó 1 e a linha branca para o nó 2.

A partir da tabela 1, é possível perceber que para uma execução paralela o resultado ideal seria aproximar-se do tempo que o SRA de maior duração leva quando executado sozinho. Entretanto, isso não aconteceu nos quatro cenários da tabela 2.

Comparando os tempos entre os dois tipos de instâncias (coluna 1 com 3 e coluna 2 com 4), percebe-se que a *m6idn.large* com um CPU físico e *hyperthreading* obtém tempos próximos aos da *m6idn.xlarge* com dois CPUs físicos e *hyperthreading*, o que indica uma melhor eficiência da instância com metade dos vCPUs para o experimento analisado. Quanto ao impacto do número de *threads*, é evidente um atraso no término de cada sequência em nós que possuem mais *threads* em razão da competição por recursos dentro de um mesmo nó. Isso sugere que o uso de muitas instâncias pequenas pode ser melhor do que poucas instâncias de maior capacidade. Neste experimento, CPUs adicionais não oferecem o nível de benefício esperado; no entanto, o uso de mais *threads* permite uma maior paralelização, resultando em uma redução do tempo total de execução.

O escalonamento utilizado prejudica o tempo total de execução, pois define estaticamente a ordem do processamento das sequências em cada nó e *thread* de forma intercalada. Um exemplo disso é a sequência SRR23922989, que leva por volta de 22 segundos, mas precisa aguardar o término de todas as outras sequências alocadas antes no mesmo nó e na mesma *thread*.

4. Conclusão e próximos passos

Com base nos resultados apresentados, podemos observar que, para o experimento analisado, a instância do tipo *m6idn.large* demonstra vantagens sobre a *m6idn.xlarge*: além de obter um resultado próximo, custa metade do preço desta última. Nesse sentido, acreditamos que tal resultado sugere que o framework para o CellHeap ainda está subótimo e que o número de CPUs não seria o único fator que influencia o tempo, podendo as operações de escrita também estar causando uma grande influência. No entanto, é ainda necessária uma análise mais aprofundada do uso de CPU e armazenamento local para arquivos temporários do fluxo de trabalho, por meio de mais testes.

Além disso, como o algoritmo de escalonamento utilizado (e a versão *worksteal-ing*) apresentou limitações em termos do makespan [Izidoro et al. 2025], trabalhos futuros pretendem utilizar diferentes algoritmos de escalonamento para lidar melhor com esse tipo de situação. Ademais, pretende-se otimizar a execução dos comandos da Fase 1 com versões paralelas e avaliar a performance com um maior número de sequências.

Referências

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

Castro, M. C. S. et al. (2023). CellHeap: A scRNA-seq workflow for large-scale bioinformatics data analysis. *bioRxiv*, pages 2023–04.

Izidoro, E., Boeres, C., and Rebello, V. (2025). Oferta ou roubo: avaliando abordagens de escalonamento dinâmico para workflow scrna-seq na nuvem aws. In *Submitido à X Escola Regional de Alto Desempenho da Região Sudeste*. SBC.

National Library of Medicine. The Sequence Read Archive (SRA). `https://www.ncbi.nlm.nih.gov/sra/docs/`. Acessado em: 05-10-2025.

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.