Desempenho de *Workflows* Científicos de Transcriptômica em Arquiteturas de Memória Distribuída e Compartilhada do Santos Dumont

Albert Emidio[1 2], Reiglan Soares[1 2], Lucas Cruz[1], Micaella Coelho[1], Kary Ocaña[1], Carla Osthoff[1], Diego Carvalho[3]

[1]Laboratório Nacional de Computação Científica (LNCC), RJ – Brasil

[2]Faculdade de Educação Tecnológica do Estado do Rio de Janeiro (FAETERJ-RJ), RJ – Brasil

[3]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET), RJ – Brasil

{albert,reiglan,lucruz,micaella,karyann,osthoff}@lncc.br,d.carvalho@ieee.org

Abstract. *Large-scale scientific experiments are complex due to the modeling, execution, and analysis of large volumes of data. In bioinformatics, these experiments are structured as scientific workflows, using workflow management systems and high-performance computing. This article presents a computational performance analysis of the transcriptomics workflow ParslRNA-Seq between the latest distributed memory and shared memory architectures of the Santos Dumont supercomputer, demonstrating how the choice of machines can influence performance.*

Resumo. *Experimentos científicos em larga escala são complexos devido à modelagem, execução e análise de grandes volumes de dados. Na bioinformática, esses experimentos são estruturados como workflows científicos, usando sistemas de gerência de workflows e computação de alto desempenho. Este artigo apresenta uma análise computacional do desempenho do workflow de transcriptômica ParslRNA-Seq entre arquiteturas com memória distribuída e memória compartilhada mais recentes do supercomputador Santos Dumont, demonstrando como a escolha das máquinas pode influenciar o desempenho.*

1. Introdução

Na era da ômica e do *big data,* experimentos em bioinformática são complexos e exigem tecnologias especializadas, incluindo *workflows* científicos, *e-Science*, aprendizado de máquina e computação de alto desempenho. Gerenciar grandes volumes de dados e adaptar *workflows* a diferentes arquiteturas computacionais é desafiador. Esses *workflows* são mais eficazes com Sistemas de Gerência de *Workflows* Científicos (SGWfC) e arquiteturas de Computação de Alto Desempenho (CAD), e sua implementação requer uma compreensão profunda dos problemas biológicos e dos dados envolvidos, pois a natureza dos dados orienta a modelagem e a escolha da arquitetura computacional mais adequada [2].

Análises de expressão diferencial de genes (EDG) são essenciais na análise de dados transcriptômicos, permitindo a identificação de genes com expressão significativamente diferente entre condições experimentais. Essa abordagem é crucial para a descoberta de novos transcritos e isoformas

de genes, proporcionando *insights* importantes sobre processos biológicos complexos, como diferenciação celular, resposta a estímulos e desenvolvimento de doenças.

Com base nas produções relacionadas, observa-se que [6] apresenta o *pipeline* TICR, que é capaz de produzir uma tabela de fatores de concordância de quartetos e também uma árvore de população. Esse *pipeline* estima a árvore procurando por discordâncias nas árvores de genes devido à recombinação. Utilizando o método MDL [1], o programa MrBayes, para realizar a inferência Bayesiana usando uma variante da Cadeia de Markov Monte Carlo [3]; o programa BUCKy, analisa a concordância Bayesiana, que recebe como entrada árvores completas geradas pela análise Bayesiana de *locus* individuais e gerando como saída uma amostra de árvores de genes e suas distribuições de probabilidade conjuntas, e Quartet Maxcut [5], utilizado para gerar a árvore de população. A tabela de fatores de concordância de quartetos e a árvore de população podem ser usadas em conjunto para a inferência de redes filogenéticas, servindo como entrada para o PhyloNetworks. [2] apresenta uma versão otimizada do ParslRNA-Seq, onde são observados ganhos de desempenho de até 70% em relação à versão alfa dentro de ambientes de Computação de alto desempenho (CAD).

Este estudo avalia o desempenho do *workflow* ParslRNA-Seq [2] na análise de EDG em diferentes arquiteturas do supercomputador Santos Dumont (SDumont). O objetivo é identificar a arquitetura de memória distribuída mais eficiente e comparar seu desempenho com arquiteturas de memória compartilhada, a fim de otimizar a execução de processos computacionais intensivos em bioinformática. Este estudo analisa e compara o desempenho das arquiteturas Cascade Lake e Ivy Bridge (MESCA2) com a arquitetura originalmente descrita no artigo Parsl-RNASeq [2], que foi executada no Ivy Bridge. Utilizamos essas arquiteturas para avaliar suas eficiências relativas em comparação com a configuração original.

2. Conceitos Básicos

2.1 Parsl-RNASeq: *Workflow* Científico de Transcriptômica

Parsl-RNASeq[1] [2] é um *workflow* científico de transcriptômica para estudos de expressão diferencial de genes (EDG), modelado com a linguagem de programação Parsl para ser acoplado em ambientes de computação de alto desempenho (CAD) do supercomputador Santos Dumont (SDumont). A Figura 1 apresenta o modelo conceitual do ParslRNA-Seq.

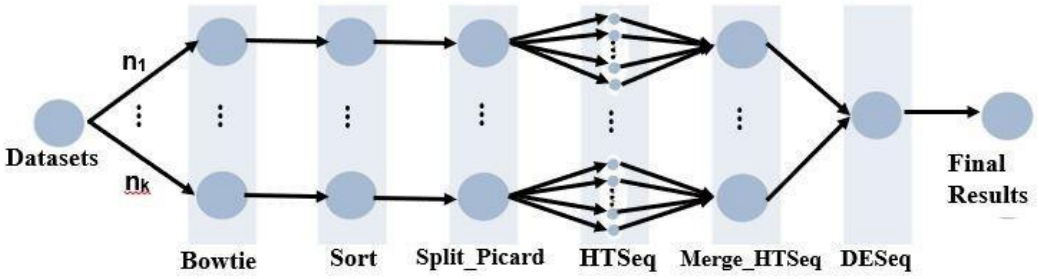

Figura 1. Modelagem Conceitual do *Workflow* Científico ParslRNA-Seq. Adaptada de Silva *et al.* 2021.

O workflow ParslRNA-Seq é composto por seis atividades e utiliza como entrada o genoma de referência do Mus musculus, arquivos GTF contendo metadados genômicos e sequenciamentos em formato FASTQ, que é utilizado para armazenar sequências biológicas, geralmente resultados de sequenciamento de DNA ou RNA. Um arquivo CSV relaciona os arquivos FASTQ às condições experimentais, sendo três de controle, que representam a condição padrão ou normal, sem intervenção

[1] https://github.com/lucruzz/RNA-seq

da via Wnt. Esses controles servem como referência para identificar mudanças na expressão gênica especificamente associadas ao tratamento com Wnt. As outras três amostras são da condição Wnt, que está relacionada à via de sinalização de transcrição Wnt. Essa via é uma importante via de sinalização celular que regula vários processos biológicos e, em estudos de transcriptômica, pode ser usada para investigar como a sinalização Wnt afeta a expressão gênica. A primeira atividade do ParslRNA-Seq emprega o Bowtie2 para mapear e comparar as leituras genômicas caractere por caractere. A segunda atividade usa o SamTools versão 1.10 para ordenar essas leituras. Na terceira atividade, o Picard versão 2.25.0 é utilizado para manipular e dividir os arquivos de leitura, melhorando o desempenho do *workflow*. A quarta atividade envolve a contagem das leituras mapeadas por gene com o htseq-count do HTSeq versão 0.13.5, distribuindo o processamento entre múltiplos núcleos para aumentar a eficiência. A quinta atividade, HTSeq-Merge, é um *script* em Python que combina os resultados da contagem de leituras em um único arquivo. Finalmente, a sexta atividade aplica o pacote DESeq2 para realizar a análise estatística de EDG nas condições experimentais.

Essas etapas formam um *workflow* integrado para análise detalhada da expressão gênica diferencial. Algumas atividades utilizam *threads*, enquanto outras utilizam *tasks*. Atividades como Bowtie e Sort fazem uso de múltiplas *threads*, aproveitando vários núcleos do processador e aumentando a eficiência e a velocidade do processamento. Em contraste, atividades como Picard, HTSeq, HTSeq-Merge e DESeq2 utilizam *tasks*, que ocupam apenas um núcleo por arquivo e não se beneficiam do multithreading, resultando em menor uso de recursos computacionais.

2.2 Parsl - Parallel Scripting Library

ParslRNA-Seq é implementado utilizando o Parsl, uma ferramenta em Python desenvolvida para executar *workflows* em ambientes de computação de alto desempenho [8]. O Parsl simplifica a criação de *workflows,* permitindo a integração de comandos externos diretamente no código Python por meio de decoradores (Apps), o que facilita a sincronização das atividades. O motor de execução do Parsl é flexível, suportando diversos ambientes computacionais e abstraindo as complexidades do *workflow*, o que torna sua implementação e integração com recursos computacionais mais simples.

O *workflow* utiliza dois modelos de execução: ThreadPoolExecutor e HTEx. O ThreadPoolExecutor é um modelo que suporta *multithreading* em recursos locais, gerenciando um *pool* de *threads* para executar atividades de forma concorrente. A eficiência desse modelo é essencial para maximizar o desempenho computacional, reduzir erros e melhorar a produtividade [9]. Por outro lado, o modelo HTEx permite a execução do *workflow* em múltiplos nós computacionais simultaneamente, facilitando o compartilhamento de dados entre as máquinas e proporcionando um controle refinado sobre a alocação de recursos e o paralelismo necessário para uma execução eficiente das tarefas.

3. Metodologia para Execução do *Workflow*

3.1 Dados de Entrada

Os dados pertencem a um experimento real de RNA-Seq[2], extraídos do repositório público *Gene Expression Omnibus*[3] (GEO) e divididos em: (1) grupo de controle: SRR5445794, SRR5445795, SRR5445796 e (2) grupo de condições das vias Wnt: SRR5445797, SRR5445798, SRR5445799. O organismo é *Mus musculus* e o GEO.ID é GSE97763 (Plataforma Illumina HiSeq 2000 - *Mus musculus*).

[2] https://sfb1002.med.uni-goettingen.de/production/literature/publications/201
[3] https://www.ncbi.nlm.nih.gov/geo/

As leituras de sequência foram alinhadas ao genoma de referência do *Mus musculus* (UCSC versão mm9) com Bowtie2. Para cada gene, o número de leituras mapeadas foi contado com Htseqcount. DESeq2 analisou a EDG a partir das matrizes das contagens do alinhamento e do mapeamento das sequências frente ao genoma de referência. Essas matrizes (arquivo GTF) contêm o número de leituras que foram alinhadas de forma única (colunas) com os exames de cada gene nas amostras (colunas). Os seis arquivos de entrada totalizam 13GB e geram um arquivo de saída de 74 GB.

3.2 O Ambiente Computacional Santos Dumont e Arquiteturas de Memória

O SDumont possui uma capacidade de processamento de 5,1 Petaflop/s, com 34.688 CPU *multicores* distribuídas em 1.132 nós computacionais que são interligados por uma rede de interconexão *Infiniband*. Todos os *softwares*, algoritmos, dependências de bioinformática (Bowtie, Samtools, Picard, HTSeq e DESeq2) e os componentes do Parsl foram alocados e instalados no ambiente do SDumont. No experimento atual, foram utilizadas três arquiteturas presentes no supercomputador Sdumont e suas configurações estão descritas a seguir na Tabela 1.

Arquiteturas	CPU	Núcleos	Memória RAM	Sistema de Memória
Ivy Bridge	2 CPUs Intel Xeon E5-2695v2	24	64 GB	Distribuída
Cascade Lake	2 CPUs Intel Xeon Cascade Lake Gold 6252	48	384 GB	Distribuída
MESCA2 (Ivy Bridge)	16 CPUs Intel Xeon Ivy Bridge	240	6 TB	Compartilhada

Tabela 1 - Configurações das arquiteturas utilizadas

Para avaliar a eficiência e o desempenho dos sistemas, comparamos dois tipos distintos de memória: compartilhada e distribuída. O MESCA2 utiliza memória compartilhada, onde todos os 240 núcleos têm acesso a uma mesma região de 6 TB de RAM. Esse modelo permite comunicação direta e rápida entre os núcleos, facilitada por uma arquitetura com cache em três níveis (L1, L2 e L3) e suporte à NUMA (Non-Uniform Memory Access), que otimiza a latência de acesso. Técnicas avançadas, como *prefetching* e *cache coherence*, asseguram que os dados necessários sejam carregados antecipadamente e que todas as cópias dos dados permaneçam consistentes, evitando conflito e mantendo a integridade durante o processamento paralelo.

Em contraste, o Ivy Bridge e o Cascade Lake empregam memória distribuída, onde cada nó possui sua própria memória local, acessível diretamente pelo processador do nó. A comunicação entre nós é realizada via rede, o que pode resultar em latências maiores devido ao tempo necessário para a transferência de dados. Para garantir a consistência entre *caches* de diferentes nós, são usados protocolos de comunicação, como a passagem de mensagens. Dados solicitados de outros nós são transferidos pela rede e temporariamente armazenados no cache local do nó solicitante. Políticas de substituição de cache, como LRU (Least Recently Used), são aplicadas localmente para gerenciar o conteúdo do cache.

No Cascade Lake, que possui processador com a arquitetura Intel Xeon Cascade Lake, inclui um cache de último nível (LLC) que apresenta um cache L2 maior em relação à geração de processadores anterior, Intel Xeon Ivy Bridge, e uma LLC não inclusiva, o que reduz a latência e melhora a taxa de acertos do cache L2, diminuindo a carga na LLC. No Ivy Bridge, que possui processador com a arquitetura Intel Xeon Ivy Bridge, todas as linhas no cache L2 de cada núcleo eram armazenadas na LLC inclusiva e compartilhada, mas no Cascade Lake, cada núcleo possui um cache L2 privado, com a LLC maior compartilhada por todos os núcleos. Essas diferenças na memória e na

arquitetura de *cache* impactam significativamente a eficiência e o desempenho do processamento de dados em cada sistema.

3.3 Configuração Experimento

Os experimentos foram implementados para 6 arquivos de entrada descritos na Seção 3.1. O *workflow* foi executado no mínimo três vezes para cada quantidade de *thread* para obter a veracidade no tempo de execução de cada análise.

3.3.1 *Workflow* executado em 1 nó (Ivy Bridge) X 1 nó (Cascade Lake)

Execução do *workflow* com configurações de 1, 2, 4, 6, 12, 18 e 24 *threads*, nas atividades *multithreading*, Bowtie e Sort, semelhantes ao experimento original, para comparar o desempenho computacional entre o nó Ivy Bridge e o nó Cascade Lake. As demais atividades executaram na forma sequencial, executando uma *task* por *core*, para cada um dos 6 arquivos. O modelo ThreadPoolExecutor do Parsl foi utilizado permitindo fazer *multithreading* em recursos locais.

Reexecução do *workflow* de [2] em um único nó computacional composto por 2 CPUs Intel Xeon Ivy Bridge de 2.4GHz, 64 GB de RAM e 24 núcleos. Execução do *workflow* em um nó computacional composto por 2 CPUs Intel Xeon Cascade Lake Gold 6252, 48 núcleos, e com 384 GB de RAM.

3.3.2 *Workflow* executado em 6 nós (Ivy Bridge) X 6 nós (Cascade Lake)

No experimento, utilizamos 6 nós Intel Xeon de 2,4 GHz com 64 GB de RAM e 24 núcleos, e 6 nós Intel Xeon Cascade Lake Gold 6252 com 384 GB de RAM e 48 núcleos. Cada nó executou um arquivo usando o modelo HTEx do Parsl, que permite a execução simultânea do *workflow*. A comparação entre as arquiteturas Ivy Bridge e Cascade Lake foi feita com diferentes quantidades de threads: 1, 2, 4, 6, 12, 18, 24 e 48, nas atividades multithreaded, Bowtie e Sort, para avaliar o desempenho e a capacidade de gerenciamento da carga de trabalho em cenários variados. As demais atividades executaram na forma sequencial, em cada um dos 6 *cores*, executando uma *task* por *core*, para cada um dos 6 arquivos.

3.3.3 *Workflow* executado em 1 nó MESCA2

A execução do workflow foi realizada com configurações de 1, 2, 4, 6, 12, 18 e 24 *threads*, semelhantes ao experimento anterior, com o objetivo de avaliar o desempenho computacional no MESCA2. As atividades Bowtie e Sort utilizaram *multithreading*, aproveitando todas as threads disponíveis no nó computacional. As demais atividades foram executadas de forma sequencial, com uma *task* sendo alocada por core, ou seja, o Parsl distribuiu cada arquivo para ser processado por um único core, resultando na execução de seis arquivos de forma independente. O modelo ThreadPoolExecutor do Parsl foi empregado nas atividades Bowtie e Sort, permitindo o uso de *multithreaded* e gerenciando essas tarefas de maneira competitiva e eficiente. Esse modelo foi utilizado apenas nas duas primeiras atividades, pois as demais ainda não possuem o suporte necessário para *multithreaded*.

Reexecução do *workflow* [2] em um único nó com 2 CPUs Intel Xeon Ivy Bridge de 2.4GHz, 64 GB de RAM e 24 núcleos.

Na presente pesquisa, o experimento foi realizado utilizando 16 CPUs Intel Ivy Bridge de 2.4GHz, totalizando 240 núcleos (15 por CPU) e 6 TB de memória RAM compartilhada.

4. Resultados e Discussão

4.1 Análise de Desempenho do *Workflow* em 1 Nó

Comparou-se o desempenho de 1 nó computacional Ivy Bridge do experimento anterior (Silva et al., 2021) com 1 nó computacional Cascade Lake processando 6 arquivos no mesmo nó. A Figura 2 mostra o Tempo Total de Execução (TTE), no Ivy Bridge, de 1 a 24 *threads,* e a Figura 3 no Cascade Lake, de 1 a 48 *threads*. O Cascade Lake teve uma melhoria de 83% com 1 *thread* e redução de 61% com 24 *threads*. O melhor desempenho foi com 18 *threads*, apresentando um TTE de 26 minutos, com aumento no tempo após esse ponto devido à falta de trabalho para *threads* adicionais.

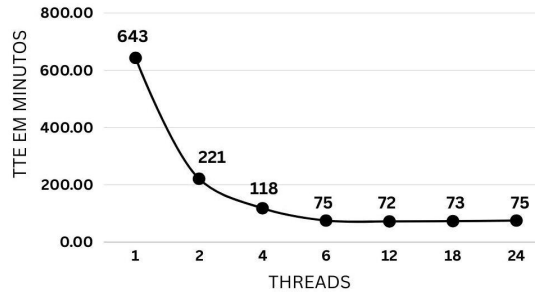

Figura 2. TTE do *workflow* em 1 nó Ivy Bridge

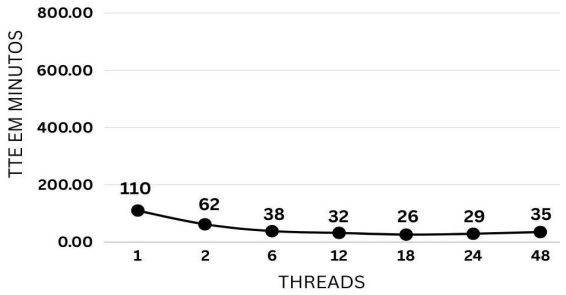

Figura 3. TTE do *workflow* em 1 nó Cascade Lake

4.2 Avaliação de Performance do *Workflow* em Múltiplos Nós

Os resultados com 6 arquivos de entrada, com um arquivo por nó, mostram a superioridade do nó computacional Cascade Lake (Figura 5) em relação ao nó computacional Ivy Bridge (Figura 4) e mostram o ganho de desempenho com o paralelismo em múltiplos nós computacionais. Para ambas as figuras a descontinuidade ocorre para 6 *cores* que corresponde ao número de *tasks* do *workflow* deixando evidente que a atuação das *threads* está nas duas atividades que suportam a tecnologia *multithreading*. Com 1 *thread*, o Cascade Lake completou o em 50 minutos, contra 60 minutos do Ivy Bridge. Com 24 *threads*, o Cascade Lake executou o em 19 minutos, enquanto o Ivy Bridge levou 21 minutos. O Cascade Lake teve seu melhor desempenho com 48 *threads*, concluindo o processo em 17 minutos. É importante observar que a curva TTE apresentada é relativa à execução de todas as atividades do *workflow* científico, onde apenas 2 atividades são executadas em *multithreading*.

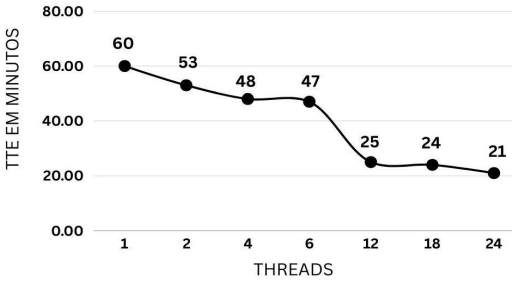

Figura 4. TTE do *workflow* em 6 nós Ivy Bridge

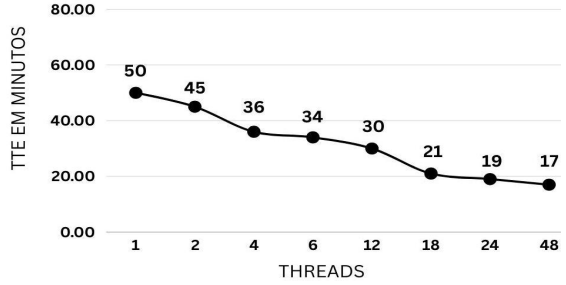

Figura 5. TTE do *workflow* em 6 nós Cascade Lake

4.3 Resultados de Desempenho do *Workflow* em 1 nó MESCA2

Comparando os resultados de [2], que executa o *workflow* com 6 arquivos (um para cada core), utilizaram 1 nó computacional Ivy Bridge em relação ao nó computacional MESCA2. A Figura 6 mostra o TTE com 24 *threads* no Ivy Bridge, enquanto a Figura 7 exibe a execução com 240 *threads* no MESCA2. Podemos observar que o MESCA2 superou o Ivy Bridge, com uma melhoria de 81%

com 1 *thread* e uma redução de 61% com 24 *threads*, alcançando o melhor desempenho com 48 *threads* e um TTE de 26 minutos**.** Essa melhoria é atribuída à memória compartilhada do MESCA2, que oferece acesso mais rápido e comunicação mais eficiente entre os núcleos.

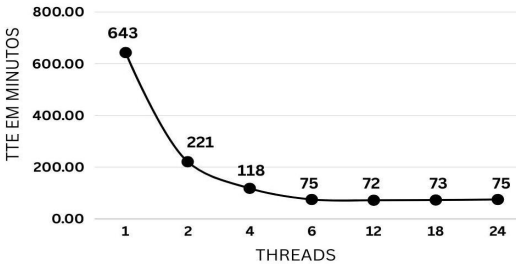

Figura 6. TTE do *workflow* em um nó Ivy Bridge

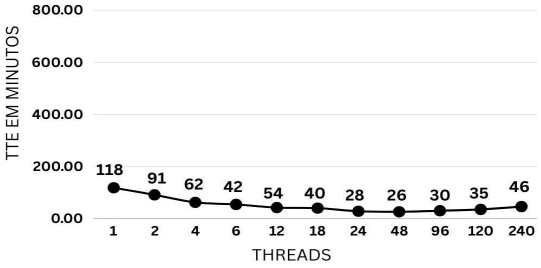

Figura 7. TTE *workflow* em um nó MESCA2

4.4 VTUNE: Análise Ivy Bridge

Esta seção tem como objetivo analisar a utilização dos *cores* em um nó computacional Ivy Bridge ao longo de toda a execução do *workflow*. Assim como no trabalho anterior [2], o VTune é utilizado para demonstrar a eficiência da paralelização do *workflow* através da implementação de tarefas, possibilitando o uso de todos os núcleos durante a execução do *workflow*, incluindo atividades que não possuem versão *multithreading*. Nesta seção, apresentamos exclusivamente o VTune do nó computacional Ivy Bridge, pois é a mesma máquina utilizada no experimento anterior. O objetivo é provar que, assim como no experimento do trabalho anterior, a implementação das tarefas permite um uso intenso de todos os núcleos ao longo de todas as atividades do *workflow*.

O Intel VTune Profiler é uma ferramenta de análise de desempenho que detalha o uso dos recursos computacionais da CPU, memória, *threading* e I/O, sendo essencial para otimizar software em sistemas Intel. A análise apresentada na Figura 8 ilustra a execução paralela das tarefas utilizando *multithreading*. Apresenta-se uma comparação com base na utilização dos processadores simultâneos (CPUs lógicas) entre seis atividades do *workflow*. O eixo horizontal representa o número de *threads*, enquanto o eixo vertical exibe o tempo máximo de execução por thread (*wall-clock*) em segundos. A utilização das CPUs lógicas pelas atividades do *workflow* é representada por cores: a cor cinza refere-se à fase de inicialização do sistema e processos das atividades; a cor vermelha indica uma utilização deficiente das CPUs; a cor amarela refere-se a uma utilização típica ou média das CPUs; e, por fim, a cor verde representa o uso ideal das CPUs, indicando que os recursos estão sendo utilizados de forma eficiente.

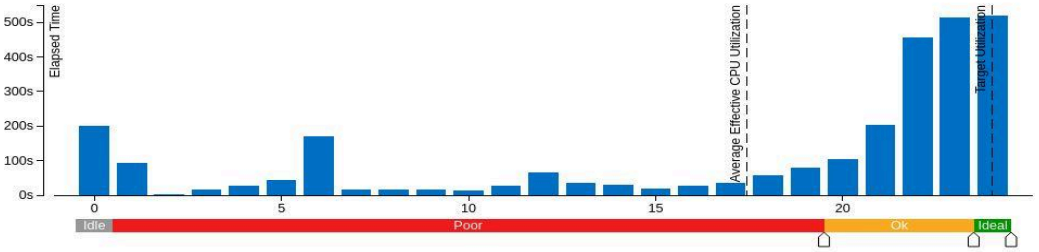

Figura 8. Uso da CPU no Ivy Bridge 24 *cores*

5. Conclusão

Os resultados evidenciam a clara superioridade do nó computacional Cascade Lake em comparação ao nó computacional Ivy Bridge para a execução de *workflows* de transcriptômica. O Cascade Lake demonstrou uma redução significativa no tempo de execução, particularmente ao utilizar 18 *threads*, evidenciando seu desempenho superior. A análise revelou que a distribuição de arquivos de entrada

entre múltiplos nós (6 nós) proporcionou uma maior eficiência. Com 48 *threads*, o Cascade Lake obteve um tempo de execução de 17 minutos, em contraste com os 21 minutos observados para o Ivy Bridge com 24 *threads*. Esta abordagem de alocação otimizada, combinada com a arquitetura avançada do Cascade Lake, resultou em um desempenho mais eficiente.

Adicionalmente, o nó MESCA2, com sua arquitetura de memória compartilhada, destacou-se em relação ao nó computacional Ivy Bridge, mostrando uma redução significativa no tempo de execução, especialmente ao usar 48 *threads*. O MESCA2 demonstrou uma utilização mais eficaz dos recursos, enquanto o Ivy Bridge enfrentou limitações mesmo com ajustes na quantidade de *threads*.

Essas descobertas ressaltam a importância de selecionar a arquitetura de hardware adequada para aplicações que exigem alto desempenho e paralelismo intensivo. O próximo passo será a continuidade da otimização do *workflow* de transcriptômica em diversos ambientes computacionais, visando aprimorar ainda mais a eficiência e a escalabilidade do processamento de dados genômicos. Esses avanços beneficiarão pesquisadores e cientistas na análise de grandes conjuntos de dados biológicos e contribuirão para o progresso da bioinformática em instituições como o LNCC e outras colaboradoras.

6. Referências bibliográficas

[1] C. Ané, "Detecting phylogenetic breakpoints and discordance from genome-wide alignments for species tree reconstruction," *Genome Biology and Evolution*, vol. 3, pp. 246–258, 2011.

[2] L. Cruz, M. Coelho, R. Terra, D. Carvalho, L. Gadelha, C. Osthoff, and K. Ocaña, "Workflows Científicos de RNA-Seq em Ambientes Distribuídos de Alto Desempenho: Otimização de Desempenho e Análises de Dados de Expressão Diferencial de Genes," in *Anais do Brazilian e-Science Workshop (BreSci)*, 2021. doi: 10.5753/bresci.2021.15789.

[3] J. P. Huelsenbeck and F. Ronquist, "Mrbayes: Bayesian inference of phylogenetic trees," *Bioinformatics*, vol. 17, no. 8, pp. 754–755, 2001.

[4] K. A. C. S. Ocaña, M. Galheigo, C. Osthoff, L. M. R. Gadelha, F. Porto, A. T. A. Gomes, D. Oliveira, and A. T. Vasconcelos, "BioinfoPortal: A scientific gateway for integrating bioinformatics applications on the Brazilian national high-performance computing network," *Future Generation Computer Systems*, vol. 107, pp. 192–214, 2020.

[5] S. Snir and S. Rao, "Quartet maxcut: a fast algorithm for amalgamating quartet trees," *Molecular phylogenetics and evolution*, vol. 62, no. 1, pp. 1–8, 2012.

[6] N. W. Stenz, B. Larget, D. A. Baum, and C. Ané, "Exploring tree-like and non-tree-like patterns using genome sequences: an example using the inbreeding plant species arabidopsis thaliana (l.) heynh," *Systematic Biology*, vol. 64, no. 5, pp. 809–823, 2015.

[7] S. H. H. Taipe et al., "An Evaluation of Direct and Indirect Memory Accesses in Fluid Flow Simulator," in *Computational Science and Its Applications – ICCSA 2023*, Lecture Notes in Computer Science, vol. 13956, O. Gervasi et al., Eds. Cham: Springer, 2023. doi: 10.1007/978-3-031-36805-9_3.

[8] Babuji, Y., Woodard, A., Li, Z., Katz, D. S., Clifford, B., Kumar, R., Lacinski, L., Chard, R., Wozniak, J., Foster, I., Wilde, M., and Chard, K. (2019). Parsl: Pervasive parallel programming in Python. In 28th ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC)

[9] Silva, R. R. and Yokoyama, R. S. (2011). Avaliação do desempenho de threads em user level utilizando o sistema operacional Linux. Revista de Informática Teórica e Aplicada´ .