UNIVERSIDADE DO ESTADO DA BAHIA

DEPARTAMENTO DE CIÊNCIAS EXATAS E DA TERRA

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

MURILO BATISTA IMPROTA BRITTO

Análise comparativa de containers para ferramentas de alinhamento de bioinformática

SALVADOR, BAHIA,

BRASIL 2023

MURILO BATISTA IMPROTA BRITTO

Análise comparativa de containers para ferramentas de alinhamento de bioinformática

Monografia apresentada ao curso de Sistemas de Informação do Departamento de Ciências Exatas e da Terra da Universidade do Estado da Bahia - UNEB, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação. Área de Concentração: Ciência da Computação

Orientador: Maria Inês Restovic

SALVADOR, BAHIA,

BRASIL 2023

Termo de Anuência do Orientador

Declaro para os devidos fins que li e revisei este trabalho e atesto sua qualidade como resultado final desta monografia. Confirmo que o referencial teórico apresentado é completo e suficiente para fundamentar os objetivos propostos e que a metodologia científica utilizada e os resultados são consistentes e com qualidade suficiente para submissão à banca examinadora final do Trabalho de Conclusão de Curso II do curso de Bacharelado em Sistemas de Informação.

Profa. Maria Inés Valderrama Restovic, (Orientadora)

MURILO BATISTA IMPROTA BRITTO

ANÁLISE COMPARATIVA DE CONTAINERS PARA FERRAMENTA DE ALINHAMENTO DE BIOINFORMATICA.

Monografia apresentada ao curso de Sistemas de Informação do Departamento de Ciências Exatas e da Terra da Universidade do Estado da Bahia - UNEB, como requisito à obtenção do grau de bacharel em Sistemas de Informação. Área de Concentração: Ciência da Computação

Aprovado em: 14/12/2023

BANCA EXAMINADORA

Profa. Maria Inés Valderrama Restovic, (Orientadora)
Universidade do Estado da Bahia – UNEB

Profa. Ana Patrícia Fontes Magalhães Mascarenhas
Universidade do Estado da Bahia – UNEB

Prof. Antônio Carlos Fontes Atta
Universidade do Estado da Bahia – UNEB

AGRADECIMENTOS

Agradeço primeiramente a minha família por toda a dedicação e investimento conferida a mim pois sem eles seria impossível chegar onde estou. Seus ensinamentos e lições ficarão eternizados em minhas memórias.

Agradeço também a meus amigos que fiz durante meu período de graduação por todo apoio moral para continuar seguindo em frente nos momentos difíceis.

Agradeço todo o suporte do grupo de pesquisa do G2BC e do setor de informática Gerência de informática (Gerinf) pela confiança e acessibilidade aos recursos de realizar o projeto e dos professores em especial da minha orientadora por me guiar durante todo o processo de pesquisa científica.

"A leitura é uma fonte inesgotável de prazer mas por incrível que pareça, a quase totalidade, não sente esta sede"

Carlos Drummond de Andrade

RESUMO

À medida que o poder computacional aumenta e se torna mais acessível, a era da bioinformática acelera a nossa capacidade de entender e enfrentar os desafios que o grande volume de dados gerados por áreas como genética e epidemiologia impõem. Várias promessas já estão surgindo, afinal a bioinformática é um campo extremamente vasto que combina diversas áreas como matemática, estatística, ciência da computação e biologia para a condução de análises de dados. No entanto nota-se uma carência muito grande de estudos comparativos entre ambientes de virtualização para ferramentas de bioinformática. Este trabalho tem como proposta um estudo analítico entre as principais ferramentas de containers da atualidade e como as ferramentas de alinhamento de alto nível de processamento impactam sobre esses ambientes. Foram comparados os containers docker e singularity com base nas métricas de tempo de execução, máximo de utilização de recursos por núcleo de cpu, média de utilização por núcleo de cpu e, gasto médio de memória do sistema durante o período de execução das ferramentas. Concluiu-se com este trabalho que ambos os containers possuem diferenças mínimas mas que, não alteram os resultados de maneira significativa, correspondendo a menos de um por cento, sendo as maiores diferenças registradas, entre os algoritmos escolhidos entre as ferramentas de alinhamento.

Palavras-chave: Containers. Ferramentas de performance. Ferramentas de Bioinformática.

ABSTRACT

As long as computing power increases and becomes more accessible, the bioinformatic age accelerates our capabilities to understand and face global challenges generated by large data volumes by fields such as genetics and epidemiology imposes. Many promises are coming, after all bioinformatic is a vast field which combines different areas like mathematics, statistics, computing science and biology to conduct data analysis. However, there is a lack of comparative studies between virtualization environments for bioinformatics tools. This work proposes an analytical study among the main container tools nowadays and how the alignment tools of high level of processing impact on these environments. The results of the comparison were satisfactory and forceful with the reality of the test scenarios submitted and the intrinsic characteristics of each environment. The Docker and Singularity containers were compared based on runtime metrics, maximum resource utilization per CPU core, average utilization per CPU core and average system memory expenditure during the tool runtime. It was concluded with this work that both containers have minimal differences but that do not change the results significantly, corresponding to less than one percent, with the largest differences recorded, algorithms chosen among the alignment tools.

Keywords: Containers. Performance tools. Bioinformatic Tools.

LISTA DE FIGURAS

SUMÁRIO

1.0 INTRODUÇÃO

A bioinformática emergiu como um domínio interdisciplinar que integra expertises em ciência da computação e genética para processar, armazenar e/ou analisar informações biológicas de acordo com as demandas da pesquisa. Devido ao considerável volume de dados gerados, um desafio crucial consiste na avaliação dessas informações provenientes de diversas fontes (RITCHIE et al, 2015). Em biologia, o maior desafio é dar sentido à grande quantidade de dados estruturais e sequências geradas pelos sistemas biológicos, por isso o desenvolvimento de ferramentas na área de bioinformática é necessário (PEVSNER, 2015).

A bioinformática tem sua origem uma década antes de um processo realizado para determinar uma sequência de nucleotídeos (adenina, guanina,citosina e timina) conhecido como sequenciamento de DNA, consistindo na análise computacional de sequências de DNA, RNA e proteínas (HAGEN, 2000).

O termo bioinformática foi utilizado pela primeira vez em 1970 no entanto, a pesquisadora pioneira Margaret Dayhoff foi quem ficou conhecida como a "mãe da bioinformática" pelo seu trabalho de desenvolvimento de computadores capazes de identificar e determinar estruturas formadas com base na ligação entre duas ou mais moléculas de aminoácidos na qual é chamada de peptídeos (GAUTHIER et al, 2018).

Após essa revolução na armazenagem de dados utilizando os computadores, ambas as ciências de computação e biologia seguiram de forma síncrona em escala de evolução com novos métodos de sequenciamentos de DNA.

Com isso, não demorou para o primeiro genoma completo ser sequenciado com cerca de 5 mil nucleotídeos em 1977. E com os novos avanços tecnológicos houve um aumento no investimento no ramo para o desenvolvimento de novos algoritmos que fossem capazes de realizar a análise de sequências de dados mais complexas, com maior volume e com o menor tempo possível (GAUTHIER et al, 2018).

Em meados de 2005, em virtude de avanços notáveis e significativos investimentos, emergiu uma tecnologia de sequenciamento de DNA inovadora conhecida como Sequenciamento de Nova Geração (NGS), promovendo uma revolução na ciência. Esta tecnologia viabiliza o sequenciamento de bilhões de fragmentos de DNA simultaneamente. Além de conferir celeridade, o NGS também propiciou uma redução substancial nos custos associados ao sequenciamento, tornando-o mais acessível financeiramente (GAUTHIER, J.et al, 2018).

1.1 NOVAS TECNOLOGIAS APLICADAS AO SEQUENCIAMENTO GENÉTICO

O desenvolvimento de tecnologias de sequenciamento da nova geração (NGS) abriram novas possibilidades, como por exemplo, para o estudo em larga escala do código genético (RITCHIE et al, 2015)

Contudo, à medida que as sequências genômicas mais recentes se tornaram mais intrincadas, acompanhadas por um significativo aumento no volume de dados a serem processados, emergiram novas ferramentas e plataformas mais sofisticadas para lidar com essas tarefas complexas. Estas demandam uma capacidade superior de processamento, maior alocação de memória, espaço de armazenamento expandido, bancos de dados robustos, além de pacotes de dependência, entre outros requisitos, Nesse sentido, surgiu a questão que norteia esse trabalho científico. Como desenvolver um ambiente ajustável para atender às demandas computacionais referentes aos projetos desenvolvidos?

Em decorrência dessa incerteza os primeiros modelos de virtualização de serviços e sistemas operacionais foram implantados. Esses modelos incluem: a) Infraestrutura como Serviço (IaaS - Infrastructure as a Service), no qual os provedores de serviços em nuvem disponibilizam infraestrutura virtualizada, abrangendo servidores, armazenamento, redes e recursos computacionais; b) Plataforma como Serviço (PaaS - Platform as a Service), no qual os provedores oferecem uma plataforma de desenvolvimento completa na nuvem, englobando infraestrutura, sistemas operacionais, serviços de banco de dados e ferramentas de desenvolvimento; c) Software como Serviço (SaaS - Software as a Service), que permite aos usuários o acesso e a utilização de aplicativos por meio de um navegador da web, dispensando a necessidade de instalação ou gestão de software em seus próprios dispositivos (PAHL et al ,2017).

A virtualização de serviços foi uma, senão a maior vantagem, da computação em nuvem devido ao fácil acesso, capacidade de múltiplas máquinas estarem ligadas em um mesmo sistema e a ausência de obrigatoriedade de um espaço físico (Daugelaite, Driscoll, Sleator, 2013).

O modelo de virtualização de sistema operacional surgiu na mesma época, mas com um intuito diferente: permitir a execução de múltiplas instâncias isoladas de sistemas operacionais em um único hardware físico. Embora não seja obrigatório, esse modelo pode utilizar o ambiente em nuvem e, hoje, é amplamente empregado em diversas áreas, desde ambientes de desenvolvimento e teste até implantações em nuvem e centros de dados.

Com base nessas demandas de diferentes tipos de serviços, surgiu a criação dos *containers* que são basicamente uma forma de virtualização de nível de sistema operacional, que permitem o empacotamento e execução de aplicativos, juntamente com todas as suas dependências e configurações, em um ambiente isolado e autocontido (R. R. YADAV, E.T.G. SOUSA, G. R. A. CALLOU, 2018).

1.2 A PESQUISA EM BIOINFORMÁTICA NA UNEB

O Grupo de Pesquisa em Bioinformática e Biologia Computacional (G2BC) e a implementação do Laboratório do G2BC, vinculados ao Colegiado de Sistemas de Informação, Departamento de Ciências Exatas e da Terra do Campus I (DCET-I) da UNEB estão desenvolvendo aplicações de bioinformática que demandam de arquiteturas e ambientes heterogêneos, Nesse sentido, já foram desenvolvidas algumas aplicações a exemplo de: Um classificador de sintomas virais que, a partir de uma análise de diversos sintomas é possível detectar a enfermidade que acomete o usuário; Um banco de dados de epidemiologia molecular.

Para desenvolver essas ferramentas é necessário que o ambiente de desenvolvimento forneça suporte para web, banco de dados, mineração de dados, inteligência artificial e outras tecnologias, além de oferecer sólido ambiente de execução para aplicações tradicionais da bioinformática, como o alinhamento de sequências e a filogenética.

Entretanto, cada uma dessas aplicações, na atualidade, opera individualmente em um servidor web ou em uma máquina local como é o caso das plataformas de alinhamento de sequências e utiliza de todos os recursos disponíveis da máquina sem ter garantias de execução em um ambiente próprio e que possa fornecer o total suporte para a realização das suas respectivas tarefas.

As principais contribuições deste trabalho para o grupo de pesquisa são: a) a criação de um ambiente sólido e conceituado, para aplicações de bioinformática, b) os containers são uma vertente de interesse do grupo, em particular, por sua capacidade de ser escalável e ajustável às necessidades do grupo, e por fim, c) a necessidade de descobrir, qual ambiente tem melhor desempenho para as ferramentas de alinhamento.

1.3 OBJETIVOS DA PESQUISA

Este projeto tem como principal objetivo realizar uma análise comparativa visando avaliar os principais *containers* disponíveis para aplicações de bioinformática, em particular, alinhamentos de sequências que são essenciais para o desenvolvimento das aplicações do grupo de pesquisa G2BC.

Os objetivos específicos traçados para a execução deste trabalho foram: a) selecionar as aplicações já realizadas de interesse do grupo de pesquisa G2BC e/ou plataformas de alinhamento sequencial para a avaliação do comportamento em ambiente de *container*, b) definir os critérios de métricas do ambiente de *container*, c) implementar o ambiente de experimentação para a testagem das ferramentas e por último, d) analisar o desempenho das ferramentas do grupo de pesquisa nos ambientes de *containers* e definir a partir dos dados coletados qual melhor se encaixa às necessidades do grupo.

1.4 ESTRUTURAÇÃO DO TRABALHO

O capítulo 2 constituído pela fundamentação teórica, traz os conceitos principais a respeito do que é a bioinformática e as suas vertentes de aplicações, e assim como as dificuldades enfrentadas na área.

O capítulo 3 discute os conceitos do que são os *containers*, como eles funcionam, como é sua arquitetura, e quais seus pontos fortes e fracos comparados com outros e como eles se aplicam para ferramentas de bioinformática.

A metodologia de pesquisa Pesquisa Experimental, que alicerça o referencial metodológico científico desta investigação, é introduzida ao longo do Capítulo 4. Nessa etapa do trabalho evidenciamos o porquê desta metodologia ser a mais adequada para conduzir a pesquisa e comparamos as etapas do processo com o que já foi realizado e o que deverá ser feito para concluirmos o projeto.

Por fim, o capítulo 5 aborda os resultados da pesquisa envolvendo e o que se pode extrair dela para aplicações futuras envolvendo as áreas de bioinformática e virtualização do sistema operacional.

2.0 BIOINFORMÁTICA: CONCEITOS E APLICAÇÕES

A bioinformática é um campo científico recente e em constante evolução. Trata-se de uma vertente que emprega soluções tecnológicas para lidar com dados biológicos, englobando atividades como aquisição e armazenamento de informações biológicas, extração de conhecimento de conjuntos de dados, busca em bancos de dados, análise e interpretação desses dados, bem como modelagem e desenvolvimento de produtos (CAN.T, 2013).

O crescimento da indústria de bioinformática é imprescindível trazendo mudanças e avanços em escala global em áreas como modelagem molecular, caracterização de doenças, descobertas farmacêuticas, cuidados clínicos de saúde, forense e agricultura (RAO et al, 2018) .

As principais aplicações referentes à área de bioinformática são: a identificação dos genes, encontrar grupos de sequências relacionadas para desenvolver modelos correspondentes, a realização de alinhamentos de sequências semelhantes e a geração de árvores filogenéticas para investigar as relações evolutivas; localizar todos os genes e proteínas de um genoma a partir de uma sequência específica de aminoácidos; e previsão de sítios ativos nas estruturas proteicas para permitir a ligação de moléculas de medicamentos. (RAO et al, 2018).

Outro enfoque muito importante é sobre a importância dos alinhamentos de sequências para a bioinformática porque ao alinhar sequências, podemos reconstruir árvores genealógicas que traçam a história evolutiva da vida na Terra. Isso nos permite explorar as relações de parentesco entre diferentes espécies, entender como os genes são herdados e compartilhados, e descobrir as bases moleculares das características e doenças humanas (AL-ABSI e KANG, 2015).

O alinhamento de sequências também desempenha um papel crucial na anotação de genomas, permitindo-nos identificar genes específicos, regiões regulatórias e elementos-chave para o funcionamento dos organismos.

No entanto, para a realização dessas atividades a integração de dados e a padronização de ferramentas na área de bioinformática desempenham um papel de suma importância no sucesso de estudos de associação e ligação.

Ter uma fonte de dados totalmente rastreável é de extrema importância, já que frequentemente os pesquisadores se deparam com anomalias nos dados em estágios avançados, o que pode demandar muito tempo para ser resolvido em uma infraestrutura que não favoreça a integração dos dados.

Outro fator que deve ser levado em consideração descrita por Kadri et al (2022) é que, a medida que os softwares que manipulam dados e informações de bioinformática se tornam mais complexos e completos, mais difícil será uma padronização de escalabilidade, reprodutibilidade, segurança das informações e gerenciamento de dependências necessárias para cumprir os seus objetivos .

Segundo Kwon, Kim & Ahn (2018) as principais razões para ter tantos problemas para utilizar as ferramentas de bioinformática são devido as ferramentas serem feitas e testadas em versões específicas e em diferentes sistemas operacionais além de necessitar de dependências adicionais e muitas vezes requer sistemas de arquivos distribuídos e configurações RAID.

Conforme essa problemática continua ocorrendo os métodos de virtualização começaram a ser cogitados e utilizados para a resolução daquele sistema em específico mas nunca houve uma padronização devido a cada método ser único e se é o mais apropriado para a ferramenta em questão.

A principal meta do grupo de pesquisa G2BC é impulsionar as pesquisas do grupo, elevando a UNEB a um nível de destaque nas áreas de pesquisa de bioinformática fúngica e viral. Por fim, a criação de aplicações web de bioinformática voltadas para fungos e vírus contribui para abordar os desafios globais contemporâneos. Dentro desse contexto, a consolidação do Laboratório de Bioinformática emerge como um elemento fundamental para o avanço das pesquisas em Bioinformática e Biologia Computacional.

Mediante a grande quantidade de dados biológicos disponíveis nos tempos atuais, torna-se necessário, o uso de ferramentas e aplicativos baseados em algoritmos sofisticados e eficientes, desenvolvidos na área de bioinformática. Além disso, é necessário o acesso a recursos computacionais de alto desempenho, para alcançar resultados em tempo razoável.

Devido às preocupações relacionadas ao armazenamento e padronização de dados das ferramentas, o G2BC propôs a adoção de um Servidor Privado. Este servidor viabiliza a execução de projetos de pesquisa aplicada, hospedando Aplicações Web de Bioinformática desenvolvidas pelo grupo. Essas aplicações serão disponibilizadas como serviços públicos e gratuitos para a comunidade científica de todo o mundo.

2.1 ALGORITMOS DE ALINHAMENTO

O grupo de pesquisa G2BC, visa algoritmos com melhores desempenho tanto por questões de menores tempos de execução decorridos, quanto por menor gasto computacional por partes desses algoritmos. Com base nessas demandas, foram selecionados algoritmos que possam discorrer de grandes sequências de dados virais e fúngicos e, que possuam uma alta taxa de eficiência comprovada.

De acordo com KATOH (2013) o algoritmo de alinhamento progressivo da ferramenta MAFFT é um método para alinhar múltiplas sequências de DNA ou RNA. O algoritmo opera na construção do alinhamento de forma incremental, começando com um par de sequências e adicionando novas sequências uma de cada vez.

O algoritmo utiliza duas técnicas fundamentais para a realização dos alinhamentos, são elas: a) Identificação de regiões homólogas na qual, o algoritmo a utiliza para identificação de regiões homólogas entre as sequências. As regiões homólogas são regiões que compartilham uma sequência similar de genes, b) Alinhamento das regiões homólogas com o qual, o algoritmo realiza alinhamento de pares de sequências para alinhar as regiões homólogas. O algoritmo de alinhamento de pares de sequências utilizado pelo MAFFT é o alinhamento local, que é mais adequado para alinhar sequências que não compartilham uma sequência similar.

Segundo KATOH (2013) o algoritmo de alinhamento progressivo do MAFFT é um algoritmo eficiente e eficaz. É um dos algoritmos de alinhamento múltiplo mais utilizados, e é frequentemente usado em estudos de filogenia e bioinformática.

Outro algoritmo que não passou despercebido é “Super 5” da Muscle é um dos algoritmos de alinhamento múltiplo mais utilizados, e é recorrentemente utilizado em estudos de filogenia e bioinformática.

Este algoritmo funciona construindo o alinhamento de forma incremental assim como o alinhamento progressivo, e inicia, com um conjunto de cinco sequências e adicionando novas sequências uma de cada vez. O algoritmo também utiliza de duas técnicas principais como: a) Alinhamento local em que, o algoritmo utiliza um algoritmo de alinhamento local para alinhar as regiões homólogas entre as sequências, b) Estratégia de refinamento na qual, o algoritmo utiliza uma estratégia de refinamento para melhorar o alinhamento. A estratégia de refinamento consiste em mover as sequências no alinhamento para melhorar a pontuação do alinhamento (EDGAR.R.C, 2021).

Ambos os algoritmos foram escolhidos por possuem pontos em comuns relevantes para o grupo G2BC como eficiência na qual, os dois podem alinhar grandes conjuntos de sequências em um tempo relativamente curto, eficácia em que, ambos produzem alinhamentos precisos, mesmo para sequências com baixa similaridade, e por fim, flexíveis já que, podem ser adaptado para diferentes tipos de sequências e aplicações conforme as necessidades do grupo de pesquisa.

2.2 PRINCÍPIOS DOS *CONTAINERS* E SUA ARQUITETURA

Os *containers* são uma forma de virtualização ao nível de sistema operacional com o intuito de fornecer isolação e gerenciamento de recursos para que os aplicativos sejam executados independente do ambiente de hospedagem. (R. R. YADAV, E. T. G. SOUSA, G. R. A. CALLOU, 2018).

Segundo Gordin (2021) Os *containers* são a solução mais utilizada para implantar e gerenciar software na nuvem. São úteis para abstrair aplicativos do ambiente físico em que estão sendo executados.

Nas palavras de Gordin (2021) muitas empresas como Youtube, Facebook e Google já começaram a usar essa solução porque eles são capazes de implantar software de maneira mais eficiente e escalar o negócio com flexibilidade e potência sem precedentes.

Um *container* nada mais é do que pacotes auto contidos sem a necessidade de um sistema operacional instanciado por completo com suas próprias bibliotecas e imagens que permitem múltiplas operações entre camadas (PAHL et al, 2017).

É disseminada uma perspectiva que os estudos a respeito dos *containers* são tão recentes que não se pode determinar estritamente o tipo de serviço em nuvem mais adequado para os mesmos já que, com determinados estudos, métodos e práticas as aplicações dos *containers* podem ser tanto usadas para infraestrutura como serviço quanto para plataforma como serviço (PAHL et al, 2017).

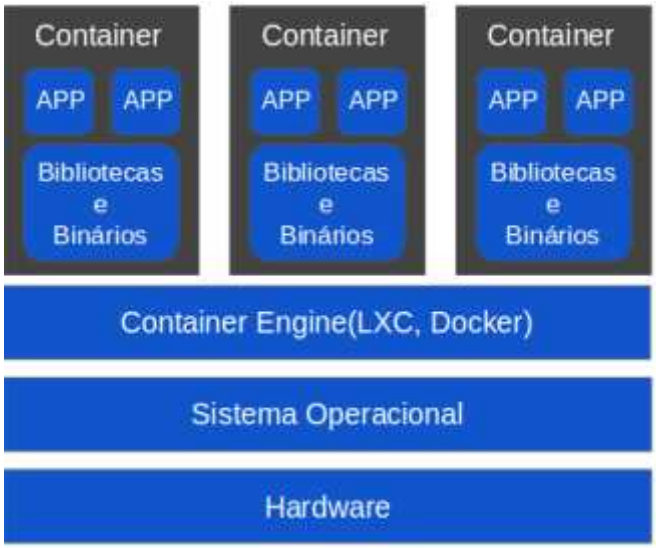
Apesar de se tratar de um estudo comparativo entre máquinas virtuais e containers, ele evidencia a perspectiva de que, em determinados cenários e com base nas métricas apresentadas, a aplicação de containers pode ser prática e funcional (RUA, KAJA e KAKADIA, 2014).

Segundo Bernstein (2014), a maioria das soluções dos *containers* são baseadas nas técnicas do Linux LXC ou containers Linux como é mais conhecido já que, o sistema fornece mecanismos para operar o kernel e isolar processos mesmo que seja em um sistema compartilhado.

Os *containers* são construídos a partir de imagens, que são pacotes leves e portáteis contendo neles a fundação do *container* , o sistema operacional e as bibliotecas necessárias. Após a escolha da imagem um arquivo de texto (writable file) é criado e dentro do mesmo, é descrito os os comandos necessários para configurar e personalizar a imagem com base em suas necessidades (PAHL et al, 2017).

Por fim, após o *container* ler o arquivo de texto, o mesmo executa cada instrução no contexto de um novo *container* temporário, criando camadas incrementais. Cada instrução é executada e seu resultado é armazenado em uma camada separada (PAHL et al, 2017). Essas camadas são empilhadas para formar a imagem final do *container* como pode ser visto na figura 1.

Figura 1 - Virtualização utilizando Containers



Fonte – Fonte: Adaptado de Morabito (2017)

2.3 TECNOLOGIAS DE *CONTAINERS*

Separamos nesta seção as principais tecnologias de *containers* utilizados em bioinformática com os resultados das bases científicas durante o processo da revisão sistemática, são eles: Docker, Singularity, BioContainers e Podman.

No decorrer das subseções, serão explicados o funcionamento de cada tipo de *container*, como sua arquitetura é definida, e suas possíveis fragilidades em comparação com outros *containers*.

2.3.1 DOCKER

Os *containers* Docker têm recebido um nível crescente de atenção por parte da comunidade científica, pois eles permitem que aplicativos sejam executados em um pacote isolado e autocontido, que pode ser distribuído e executado de maneira eficiente em uma variedade de plataformas de computação, de forma portátil (GERLACH et.al, 2014; BOETTIGER, 2015).

Uma das principais vantagens da utilização do Docker é substituição de uma instalação complicada de vários softwares, com suas dependências complexas, a partir de uma única imagem pré-configurada e pronta para ser executada. Essa imagem contém todos os softwares necessários e suas configurações correspondentes.

Outra vantagem do Docker é que ele executa cada processo em um ambiente isolado, criado a partir de uma imagem imutável. Isso evita conflitos com outros programas instalados no ambiente de hospedagem e garante que cada processo seja executado em uma configuração de sistema previsível, que não pode ser alterada devido a erros de configuração de software, atualizações do sistema ou erros de programação (PAHL et al, 2017).

Além disso, os *containers* Docker possuem uma excelente capacidade de escalonamento, o que significa que é possível executar várias instâncias do mesmo aplicativo simultaneamente e de forma isolada. Os mesmos têm um tempo de inicialização rápido e exigem um baixo consumo de recursos, o que os torna apropriados para serem implantados em grande escala.

No cenário das plataformas de *container* disponíveis, o Docker está se consolidando como o ambiente padrão para a rápida composição, criação, implantação, dimensionamento e monitoramento de aplicativos no sistema operacional Linux (NULKANI et al. 2018).

O processo de criação dos *containers* docker é muito semelhante ao descrito na seção anterior já que o mesmo utiliza a estrutura adaptada para o linux arquitetura baseada em *namespaces* e *cgroups* do Linux para fornecer isolamento de processos e recursos entre os *containers* e o sistema operacional. Isso permite que os mesmos compartilhem o mesmo kernel do sistema operacional, mas tenham ambientes isolados para executar o aplicativo. E outra diferença é o tipo de arquivo de texto utilizado pelo mesmo, chamado de "Dockerfile".

O Dockerfile deve conter uma série de instruções que descrevem os passos necessários para construir uma imagem de *container*. Essas instruções incluem comandos para a instalação de pacotes, a cópia de arquivos para dentro do *container*, e a configuração de variáveis de ambiente.

Em contrapartida aos pontos fortes de aplicabilidades, as vulnerabilidades presentes no sistema docker mostradas pelo autor, Martin et al (2018) podem ser tratadas, e mitigadas para prevenir possíveis ameaças como por exemplo, falhas de segurança em bibliotecas, sistemas operacionais desatualizados e configurações inadequadas de imagens que poderiam resultar em mal funcionamento do *container*.

2.3.2 SINGULARITY

De acordo com Kadri (2022), o singularity é uma plataforma de *container* de código aberto que oferece recursos para criar, executar e gerenciar *containers*. Sua ênfase está na portabilidade e compatibilidade, especialmente em ambientes científicos e de computação de alto desempenho .

O Singularity foi projetado para permitir que cientistas e pesquisadores executem suas aplicações e experimentos em diferentes ambientes computacionais, sem precisar modificar ou adaptar seu código.

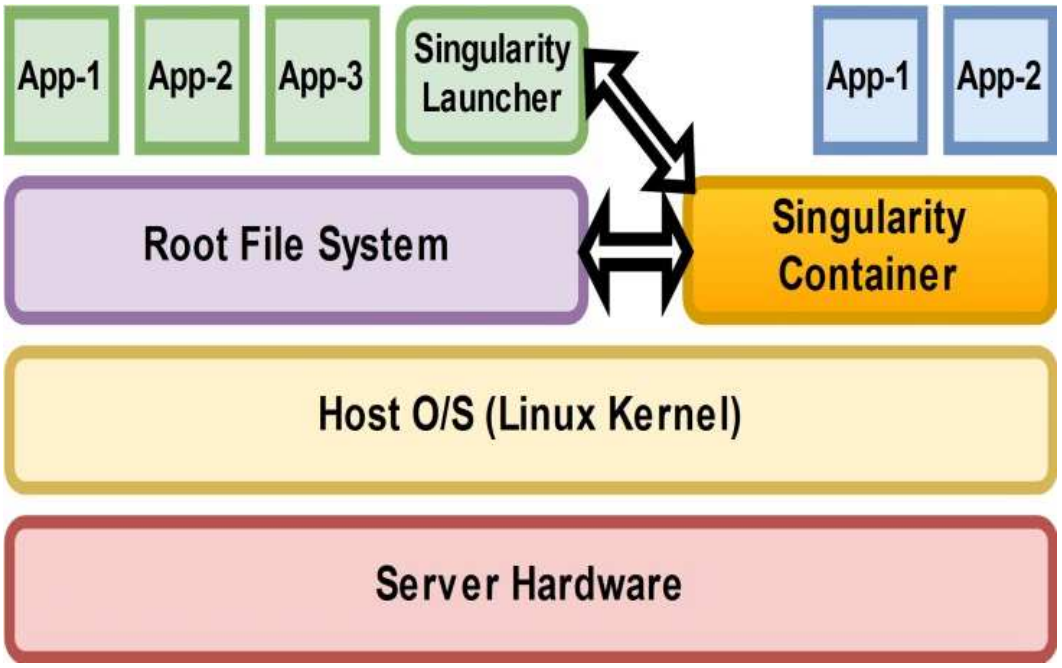
O Singularity é uma plataforma de *container* totalmente independente, projetada para garantir segurança e flexibilidade ao executar *containers* em ambientes de HPC (KADRI, 2022).

Uma característica fundamental do Singularity é sua capacidade de executar *containers* em um modo com maior acesso, comumente conhecido como modo “privilégio”, permitindo que o *container* tenha acesso a determinados dispositivos e recursos de hardware do sistema hospedeiro. Isso faz do Singularity uma escolha popular entre cientistas e pesquisadores que necessitam executar aplicações com requisitos específicos de hardware ou que dependem de bibliotecas e configurações personalizadas.

Dessa forma, a estrutura de segurança do Singularity, quando comparada a outros containers como o Docker, que utiliza o modelo cliente-servidor, difere-se ao não conceder automaticamente confiança a todos os associados ao servidor (daemon) para operar, criar e alterar um *container*.

Segundo Kadri (2022) uma das características de segurança chave do Singularity é que o usuário dentro do *container* é o mesmo usuário que o iniciou, e não é necessário ter acesso de root para executar os *containers* . Isso garante a segurança, pois os *containers* são executados com as permissões do usuário que os iniciou, evitando a necessidade de privilégios elevados que poderiam comprometer a confiabilidade e segurança do sistema conforme é mostrado na figura 2 a seguir.

Figura 2- Modelo de estrutura do container singularity



Fonte: Akalanka Bandara Mailewa (2017)

2.3.3 BIOCONTAINERS

Os *containers* de bioinformática, em especial os BioContainers, servem para facilitar a análise escalável desses dados, fornecendo um ambiente isolado e portátil para executar as ferramentas de análise (RIVEROL e MORENO, 2019) .

O mesmo ainda retrata as maiores dificuldades enfrentadas pela comunidade de bioinformática como disponibilidade do software, reprodução dos resultados e implementação do ambiente que foram sanadas pelos BioContainers.

Os BioContainers, são como uma iniciativa de padronização e compartilhamento de software para a área da bioinformática com um estudo mais detalhado destacando as vantagens do uso de *containers*, como a simplificação da instalação e configuração de software, a eliminação de conflitos de dependências e a facilitação da colaboração científica (LEPREVOST et al, 2017).

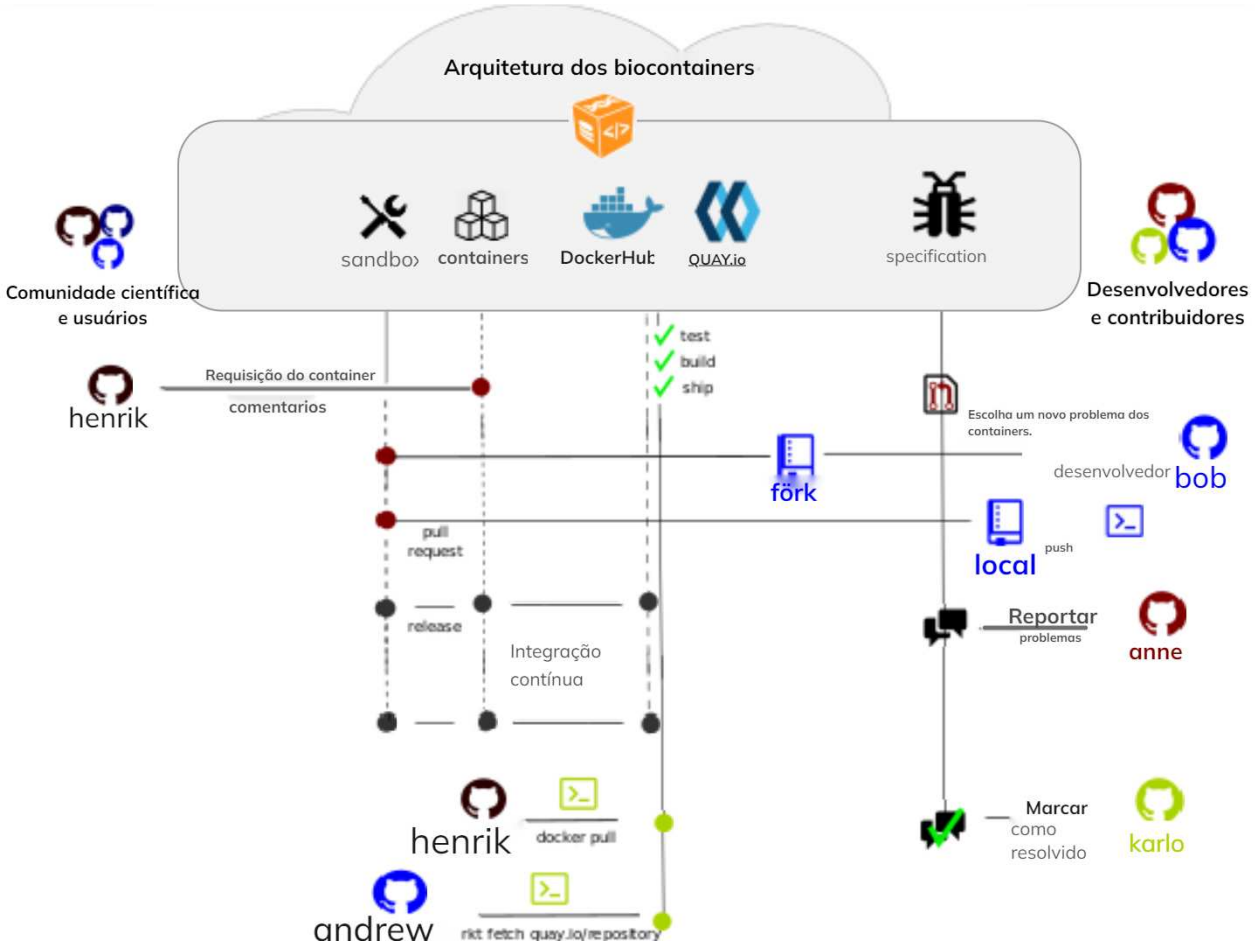
A arquitetura dos BioContainers fundamenta-se em tecnologias de containers, como o Docker e o Singularity, mencionados anteriormente. No entanto, vale destacar dois outros componentes. Primeiramente, os BioContainers recebem atualizações de pacotes e definições

de ambiente pré-programadas, voltadas especificamente para ferramentas de bioinformática, já conceituadas no mercado, diferente dos outros containers, que possuem ambientes voltados para diversos tipos, e são armazenadas como arquivos graváveis. Além disso, as ferramentas e a criação de contêineres são gerenciadas por meio do GitHub (LEPREVOST et al, 2017).

A segunda é o BioContainers Registry, uma plataforma centralizada e automatizada que simplifica a descoberta de containers e impede que os usuários precisem criá-los desde o início, visando padronizar os containers existentes na plataforma.

Na figura 3 a seguir, usuários, membros da comunidade científica, desenvolvedores e/ou contribuidores realizam a interação com o ambiente BioContainers Registry na qual realizam solicitações, *downloads*, *uploads* e criações de imagens ou ambientes pré-configurados para aplicações de bioinformática.

Figura 3: Arquitetura comunitária dos Biocontainers



Fonte: Página de documentação de BioContainers

Disponível em:< https://biocontainers-edu.readthedocs.io/what_is_biocontainers.html>

2.3.4 PODMAN

Outra alternativa de *container* a ser utilizada nessa pesquisa é o Podman, uma plataforma de *containers*, em ambientes de computação de alto desempenho (HPC) (GANTIKOW, WALTER e REICH , 2020).

Em seu trabalho os autores Gantikow, Walter e Reich (2020) investigam a funcionalidade do Podman em permitir a execução dos *containers* sem a necessidade de privilégios de root, o que oferece vantagens significativas em termos de segurança e facilidade de uso.

Além disso, Gantikow, Walter e Reich (2020) comparam a compatibilidade do Podman com as interfaces de gerenciamento de *containers* existentes, como o Docker, e destaca recursos e funcionalidades que tornam o Podman uma opção viável para ambientes de HPC.

A arquitetura do Podman se baseia em processos separados. Ao contrário do Docker, que depende de um daemon centralizado (usado para descrever o processo em segundo plano que gerencia e controla a execução de *containers*), o Podman opera de maneira distribuída. Cada *container* no Podman é executado como um processo individual no sistema operacional. Essa abordagem dispensa a necessidade do Docker Daemon, o qual seria responsável por atender solicitações para criar, iniciar, parar e gerenciar *containers*. Além disso, os containers do Podman, podem ser executados no contexto de um usuário normal, eliminando a exigência de privilégios de root.(GANTIKOW, WALTER e REICH , 2020).

O Podman utiliza recursos do kernel Linux, como *namespaces* e *cgroups*, dois recursos fundamentais do kernel Linux que desempenham papéis importantes na virtualização e isolamento de processos. Eles são essenciais para a funcionalidade dos *containers* no ambiente Linux.. Fornecendo assim uma camada de isolamento entre recursos do sistema, identificadores de arquivo, limites e políticas para recursos de processos e sistemas de arquivos.

O Podman utiliza os mesmos formatos de arquivos de imagens do Docker podendo compartilhar muitas funcionalidades das imagens do sistema de *containers* Docker.

Esses são os métodos de virtualização ao nível de sistema operacional selecionados para as nossas pesquisas devido às suas grandes aplicabilidades e variedades de implementações de diversos tipos de ferramentas assim, como uma quantidade de material disponível apresentado consistente.

Após o levantamento de todo o material científico coletado, com base no referencial teórico encontrado, e pelas necessidades do grupo, em integrar suas aplicações em ambientes mais conceituados e com maior suporte oferecido pelas plataformas, as tecnologias de *containers* adotadas para os cenários de testes são os *containers* docker, por serem consolidados e frequentemente usado por grandes instituições de renome, além de possuir um grande respaldo científico, com diversos artigos publicados a seu respeito, mostrando sua eficácia.

A outra ferramenta adotada para a realização dos testes foi o singularity, na qual tem como proposta, alta performance em ambientes científicos, o que é propício para com as intenções e interesses do grupo de pesquisa, além de possuir artigos científicos relevantes que apontam sua eficiência tanto em desempenho quanto em portabilidade.

3.0 - MÉTRICAS AVALIATIVAS

Os *containers* fornecem uma série de funcionalidades práticas relacionadas ao uso do ambiente, são essas características:

Portabilidade - Os containers permitem que o software desejado seja executado em diversos ambientes de maneira consistente (M. G. XAVIER e F. D. ROSSI, 2015).

Eficiência - Pelo fato dos containers utilizarem do mesmo Sistema operacional do usuário, as aplicações se tornam mais leves em termos de recursos gastos em comparação a outros métodos de virtualização (PAHL et al, 2017).

Gerência de dependências - Os contêineres facilitam o gerenciamento de aplicativos e executam múltiplas instâncias de aplicativos em um único sistema (PAHL et al, 2017).

Escalabilidade - Os containers permitem a replicação de aplicativos e serviços, assim, os usuários podem aumentar ou diminuir a escala de aplicativos com rapidez e facilidade, conforme necessário (M. G. XAVIER e F. D. ROSSI, 2015).

Isolamento de recursos - Uma das suas principais funções é manter o nível de isolamento entre aplicativos, garantindo que as alterações feitas em um container não afetem outros containers ou o sistema host (PAHL et al, 2017).

Segurança - Medidas de segurança implementadas, como isolamento de processos, gestão de permissões e proteção contra ameaças externas (MARTIN et al , 2018).

Os recursos mais frequentemente examinados para avaliar o desempenho de sistemas computacionais são a unidade central de processamento, a memória de acesso aleatório, a conexão de rede e o dispositivo de armazenamento em disco (R. R. YADAV, E. T. G. SOUSA, G. R. A. CALLOU, 2018).

Os parâmetros que são utilizados para avaliar o gasto computacional de recursos tanto de máquinas virtuais quanto de containers são: Taxa de utilização de CPU, tempo de execução, taxa de utilização de memória e utilização média de núcleo de CPU (JHA, D. N. et.al 2019) e (HU, G., ZHANG, Y, & CHEN, W, 2019).

A taxa de utilização de CPU está diretamente ligada ao grau de disponibilidade do recurso de processamento do sistema. O tempo de execução trata do tempo para se concluir uma determinada tarefa do sistema. A taxa de utilização da memória denota a quantidade de memória usada por um determinado sistema em uma determinada unidade de tempo. A utilização média de

núcleo de CPU está diretamente ligada com a taxa de utilização de CPU e, que denota em saber o gasto individual de um núcleo em um determinado período de tempo (R. R. YADAV, E. T. G. SOUSA, G. R. A. CALLOU, 2018) e (M. G. XAVIER e F. D. ROSSI, 2015).

Para a realização dessa análise de desempenho de sistemas geralmente são utilizados *benchmarks*, um conjunto de utilitários especializados utilizados para avaliações de desempenho. Eles oferecem ao usuário a capacidade de ajustar variáveis como, o número de threads ou o tamanho do arquivo em testes que envolvem a leitura de disco (M. G. XAVIER e F. D. ROSSI, 2015).

Existem vários *benchmarks* disponíveis no mercado, especialmente para sistemas operacionais Linux, como o Sysbench, IOZone, e o Build Apache (M. G. XAVIER e F. D. ROSSI, 2015).

A escolha do Sysbench é muito promissora devido a sua capacidade de adaptar as cargas de trabalho e sua ampla cobertura de recursos computacionais avaliados. Este utilitário é eficaz para medir os tempos totais de resposta de solicitações em relação a vários componentes do sistema.

Outras escolhas para ferramentas de monitoramento foram cogitadas e analisadas como htop, top,iostat, iotop e perf entre outros, com o intuito de coletar o máximo de informações de desempenho relevantes para a execução desse trabalho.

3.1 - TRABALHOS RELACIONADOS

Esta seção retrata trabalhos que estão relacionados com análises comparativas entre containers e máquinas virtuais. As métricas estabelecidas entre os paralelos serão levadas em consideração para compararmos diferentes tipos de containers.

O estudo de BARIK et al (2016) com seus colegas abordaram a comparação entre máquinas virtuais e containers em ambientes de nuvem. Eles descobriram que as máquinas virtuais demonstraram sobrecargas significativas, impactando consideravelmente o desempenho global da virtualização. Em contraste, os containers revelaram mínimas indicações de sobrecarga. Em resumo, os resultados indicaram que os containers superaram ligeiramente as máquinas virtuais, com a diferença sendo notável em uma escala reduzida. No entanto, o estudo careceu de uma metodologia claramente definida para a comparação de desempenho entre os dois.

No estudo de XAVIER et al (2015) e sua equipe, examinaram como o desempenho de um sistema pode ser afetado pela presença de dois containers. Eles optaram por empregar os bancos de dados Oracle e MySQL para simular diversas cargas de trabalho no sistema. Isso foi realizado utilizando o Swingbench e o Sysbench para simular transações online, como pedidos novos, pagamentos e controle de estoque. A métrica primária escolhida para a análise foi o número de transações.

As cargas de trabalho se assemelham a atividades comuns em ambientes de comércio eletrônico, incluindo a entrega de pedidos e a criação de novos pedidos. No primeiro cenário, um container se dedicava a tarefas específicas enquanto o outro permanecia ocioso. Foram monitorados recursos físicos individuais (CPU, disco e memória RAM) e os dados correspondentes foram registrados. Em um segundo estágio, ambos os containers foram submetidos a cargas de trabalho, e métricas adicionais foram coletadas.

As descobertas indicaram que, em ambientes que empregam containers, a combinação de cargas de trabalho que afetam o disco de armazenamento e a memória não é recomendada quando os containers estão hospedados na mesma máquina física. Este estudo ressalta a importância de considerar cuidadosamente a distribuição das cargas de trabalho ao implementar soluções baseadas em containers.

No artigo de Yadav, Sousa e Callou (2018) foi sugerida uma análise comparativa entre o desempenho de máquinas virtuais e containers. Com o intuito de alcançar esse objetivo, propõe-se uma abordagem metodológica que permita a comparação desses ambientes. A tecnologia de máquinas virtuais opera por meio de um hypervisor, enquanto a solução baseada em containers agrega as aplicações junto de suas bibliotecas e arquivos de sistema, dispensando a necessidade do hypervisor.

No estudo de Jha et.al (2019) foi investigado o desempenho de containers Docker executando microsserviços HPC distintos. Os autores estão particularmente interessados em como a interferência dentro do container e entre containers afetam o desempenho. Eles também examinam as variações de desempenho nos containers Docker quando grupos de controle (cgroups) são usados para limitação de recursos. Os autores usaram a Metodologia de Experimento de Avaliação de Nuvem (CEEM) para realizar seus experimentos. CEEM é uma metodologia para realizar experimentos repetíveis e confiáveis em ambientes de nuvem. Os autores usam o CEEM para definir seu design experimental, coletar dados e analisar resultados.

No geral, o artigo fornece conhecimentos valiosos sobre o desempenho dos

microsserviços HPC em ambientes de container. As descobertas dos autores podem ser usadas para orientar o desenvolvimento de microsserviços HPC na nuvem.

No artigo os autores Kumar e Thangaraju (2020) compararam o desempenho de dois *runtimes* de containers: RunC e Kata. RunC é o *runtime* padrão do Docker, enquanto Kata é um *runtime* leve que usa uma leve virtualização para melhorar a segurança.

Os autores conduziram uma série de experimentos para comparar o desempenho do RunC e do Kata. Os experimentos mediram as seguintes métricas:Uso da CPU,uso da memória,latência de rede e throughput de rede. Com os resultados, os autores concluíram que o RunC supera o Kata na maioria dos experimentos. No entanto, o Kata tem algumas vantagens sobre o RunC. Por exemplo, o Kata fornece mais segurança que o RunC e pode ser usado para executar containers que exigem isolamento de hardware.

No artigo de Hu, Zhang e Chen. (2019) Os autores realizaram uma avaliação de desempenho detalhada do Singularity comparando-o ao native usando uma série de benchmarks, incluindo HPL, STREAM e OSU Micro. Também foram realizadas outras quatro aplicações HPC típicas para validar ainda mais o impacto no desempenho do Singularity. Os resultados dos experimentos mostram que o *container* pode alcançar resultados de desempenho próximo ao native em aplicações MPI e/ou GPU paralelas.

Após a seleção, leitura e aprofundamento do conteúdo analisado nos artigos conclui-se que os *containers* possuem uma aplicação prática em qualquer área de estudo, fornecem um ambiente próprio com o máximo de performance igual ou superior a outros métodos de virtualização, e muitas métricas de desempenho desses artigos podem ser usados para diversos cenários de diversas aplicações como é o caso da proposta deste trabalho.

Foi constatado que, em nenhum dos trabalhos relacionados, os *containers* foram utilizados em conjunto com ferramentas de bioinformática. Isso gera várias dúvidas para diversos grupos de pesquisa científica, incluindo o G2BC, e para usuários que fazem uso de ferramentas de bioinformática. A pergunta que se coloca é: qual *container* oferece o melhor desempenho para atender às minhas necessidades?.

4.0 - DESCRIÇÃO DO PROJETO

O objetivo deste trabalho é realizar uma análise comparativa dos desempenhos de diferentes contêineres, utilizando métricas derivadas de artigos referenciados. A intenção é testar a eficiência e o desempenho desses contêineres em relação a ferramentas de alinhamento sequencial em bioinformática, especificamente o MAFFT e o MUSCLE.

A métrica escolhida para avaliação é a eficiência, considerando diversos fatores. Este trabalho propõe comparar o consumo computacional de ferramentas de alinhamento em ambientes de *containers*, excluindo considerações como portabilidade, isolamento de recursos, segurança, gerenciamento de dependências e escalabilidade, que não são abordadas neste experimento. Um segundo fator relevante é a limitação de tempo para aplicar uma variedade extensa de cenários de mudanças, o que inviabiliza a consideração dessas métricas avaliativas.

Várias ferramentas foram levadas em consideração, inclusive, ferramentas de benchmark e de monitoramento. No entanto, no cenário aplicado, a utilização de um benchmark não é viável devido a questões da natureza da ferramenta. Os benchmarks realizam um conjunto unitário de testes com cargas de trabalho para medir a capacidade total do sistema mas, a ferramenta não pode monitorar em tempo real e nem coletar informações de programas em tempo real, ou seja haveria dois programas dentro do container em execução e que nenhum poderia interagir com o outro.

Cada uma das ferramentas de monitoramento de desempenho disponíveis citada anteriormente, individualmente não analisam a completude do sistema, e seria necessário utilizar várias em conjunto para coletar os dados necessários. Devido a isso a escolha mais apropriada foi o “nmon” uma ferramenta de monitoramento de desempenho para sistemas Unix-like, incluindo sistemas operacionais como Linux e AIX.

A sua principal característica é a capacidade de fornecer uma visão abrangente do desempenho do sistema em tempo real. Por exemplo, mostra a carga da CPU por núcleo, detalhando a utilização do sistema, usuário, sistema e ociosa. Oferece informações sobre o uso de memória, incluindo RAM, *swap* e *buffers*. Exibe estatísticas sobre a atividade de entrada/saída do sistema de arquivos. Além disso, pode gerar arquivos de log que podem ser analisados posteriormente para revisar o desempenho do sistema ao longo do tempo.

4.1 - METODOLOGIA DE PESQUISA

A metodologia que predomina esse projeto de pesquisa fundamenta-se na abordagem conhecida como Pesquisa Experimental. A mesma se caracteriza por manipular diretamente as variáveis relacionadas com o objeto de estudo com o intuito de testar diversas hipóteses que estão de acordo com a convicção de quem está pesquisando. De acordo com o autor Gil (2008), é definido um esquema para a realização de experimentos com a proposta de definir variáveis e observar o efeito delas sobre o objeto estudado.

Resumidamente, o processo de uma Pesquisa Experimental se dá em uma sequência de passos previamente estipulados, o primeiro passo refere-se a definição do objeto de pesquisa e seus objetivos específicos que estipulamos anteriormente no qual é uma comparação entre dois sistemas de virtualização de sistemas operacionais para aplicações de bioinformática e que busca atender as métricas nos próximos passos.

Nesta segunda etapa foi definida pelo nosso referencial teórico a respeito do que são os *containers ,* como eles funcionam, e quais são suas funcionalidades.

Nesta terceira etapa são definidas as variáveis independentes e dependentes da nossa aplicação sendo as variáveis independentes as que serão manipuladas e controladas e as dependentes são as que queremos estudar a fim de ver como elas impactam as independentes.

Em paralelo a nossa pesquisa, nosso ambiente de *container* executa o papel de variável independente e queremos que ele opere a partir das seguintes métricas de desempenho referenciadas pelos artigos que são: tempo de resposta, tempo na fila, taxa de utilização de recursos e tempos de execução (todas essas são nossas variáveis dependentes).

O quarta etapa se refere a geração de carga de trabalho que as dessas variáveis independentes sobre as dependentes para a obtenção da coleta de dados.

E por último a quinta etapa de análise e interpretação dos dados. Aqui, será feito a organização e tabulação dos dados utilizados na etapa anterior assim como será feito sua classificação junto com um suporte estatístico para melhor precisão na avaliação de cada caso. Vale ressaltar que, todo esse processo ocorre de maneira encadeada ou seja não é possível seguir para uma próxima etapa sem realizar a anterior. Segue o quadro resumo para ilustrar o passo a passo da metodologia no diagrama.

ETAPA 1	Definição do Objeto de Pesquisa e Objetivos
ETAPA 2	Revisão Teórica sobre Containers
ETAPA 3	Definição de Variáveis dependentes e independentes
ETAPA 4	Geração de Carga de Trabalho
ETAPA 5	Análise e Interpretação dos Dados

fonte: o autor

A metodologia experimental apresenta uma abordagem cuidadosa e estruturada para avaliar o desempenho de diferentes containers em um contexto de bioinformática. Destacam-se a escolha da Pesquisa Experimental, a definição clara de variáveis, a ênfase em métricas relevantes e a organização lógica das etapas. Ao final deste processo espera-se obter resultados comparativos e precisos para uma tomada de decisão analítica suprindo assim o objetivo geral da pesquisa.

4.2 - INSTALAÇÃO DAS FERRAMENTAS DE ACESSO AO AMBIENTE

O ambiente para pesquisa foi disponibilizado pela Gerência de informática (Gerinf) e até o momento, configura-se em: Um servidor de memória RAM de 32 Gigabytes, 10 núcleos físicos de processadores intel Xeon, sistema operacional Debian, armazenamento entre 100 a 200 gigabytes, taxa de internet compartilhada de até 1 gigabyte.

Para acessar o servidor foi necessário um software de cliente vpn para se conectar a rede privada da Universidade do Estado da Bahia (UNEB), chamado “FortiClient VPN”, nesse software estabelecemos o tipo de conexão SSL-VPN (Secure Sockets Layer Virtual Private Network) que cria uma conexão segura e criptografada entre o dispositivo do usuário e a rede privada, protegendo os dados transmitidos de possíveis interceptações ou violações e preenchemos os campos de gateway remoto direcionando a URL relacionada a instituição e, os campos de usuário que faz o acesso e a senha.

Após configurar o software de vpn foi necessário um programa para acessar servidores remotos e dispositivos de rede que executam sistemas operacionais baseados em Linux, e esse programa recomendado pela (Gerinf) foi o “PuTTY”.

Foi usado em conjunto um software cliente de código aberto para transferência dos arquivos de uma máquina local para um servidor remoto através de protocolos como FTP (File Transfer Protocol), FTPS (FTP over SSL/TLS), e SFTP (SSH File Transfer Protocol) conhecido como Filezilla.

4.3 - INSTALAÇÃO DA FERRAMENTA DOCKER

O processo de instalação do Docker escolhido foi seguindo o passo no site oficial do Docker Scout que foi desenvolvido em 2013 por pela empresa Docker, INC. Ressaltando que todos os comandos do processo de instalação e criação dos *containers* estão disponíveis na seção de Apêndice A. O guia foi instituído da seguinte maneira:

1. Atualização do índice de pacotes. Isso garante acesso às versões mais recentes dos pacotes disponíveis.

2. Instalação dos pacotes necessários para permitir o uso de repositórios sobre HTTPS, que é uma prática de segurança recomendada.

3. Adicionar a chave GPG para o repositório oficial do Docker para garantir a autenticidade e integridade do software que está sendo baixado.

4. Adicionar o repositório do Docker ao sistema para que o sistema saiba onde encontrar os pacotes e para termos acesso ao docker em qualquer diretório e ao atualizar os repositórios.

5. Instalar a versão mais recente dos pacotes do docker e dos containers, que são os componentes essenciais.

6. Verificar se o docker está instalado com comandos básicos.

Terminado todo esse processo o método de virtualização está pronto para uso e na sua versão de programa e pacotes devidamente atualizados.

4.4 - CRIAÇÃO DOS CONTAINERS DOCKER

Foi definido que, para um melhor gerenciamento dos containers e arquivos que estão presentes neste trabalho, diretórios específicos para cada ferramenta de alinhamento foram criados e inclusive os arquivos do tipo Fasta que é um formato de arquivo de texto simples especifico para representar sequências de nucleotídeos ou aminoácidos foram importados para os diretórios onde os containers estão. Entretanto, os arquivos que contém as sequências não estão dentro dos containers. A primeira ferramenta de alinhamento instalada foi o MAFFT, entretanto o mesmo processo criativo estendeu-se para a ferramenta Muscle. Para a realização dessas tarefas alguns passos foram estabelecidos.

1. Foi criado um arquivo de texto com nome Dockerfile.Esse arquivo é utilizado para definir a imagem com uma série de instruções de procedimentos que serão realizados dentro do arquivo.

2. São definidos todos os pacotes, dependências e programas que serão instalados que serão a ferramenta MAFFT e o nmon.

3. Construção da imagem a partir do arquivo de texto.

4. É Inicializado a criação do volume do *container* no diretório atual, sem criar um volume para o container qualquer dado ou informações que deveriam ser armazenadas ou acessadas não estarão disponíveis, podendo acarretar em problemas de acesso a arquivos, perda de dados e ou mal funcionamento da ferramenta ou software que depende desse volume.

5. Dentro do Container os arquivos que estão no diretório atual podem ser chamados a partir dos comandos das ferramentas.

4.5 - INSTALAÇÃO E CRIAÇÃO DOS CONTAINERS SINGULARITY

O processo de instalação do Singularity escolhido foi semelhante ao docker, pois seguimos a documentação oficial da Sylabs.io que foi desenvolvido em 2015 pela Sylabs Inc.

Certas condições foram alteradas no processo de criação das imagens do singularity, como é um estudo comparativo as imagens do singularity foram transcritas do docker para preservar as dependências, versões e pacotes dos programas. Vale ressaltar que todos os comandos do processo de instalação e criação estão disponíveis na seção de Apêndice B. Segue abaixo o processo:

1. Diferentemente do docker que foi instalado no sistema, o singularity foi instalado em um diretório próprio para gerenciar suas pastas e dependências. Essa medida foi adotada por recomendação da documentação oficial para o máximo aproveitamento do container sem sofrer interferências externas.

2. Para compilar o Singularity a partir do código-fonte, será necessário ter o Go instalado no sistema para garantir que o processo de compilação ocorra sem problemas de dependências.

3. Após a criação do diretório, deve ser baixado o código fonte da sua página oficial do github.

4. Por fim, a partir do código fonte baixado, instalar o singularity com suas dependências e pacotes já atualizados.

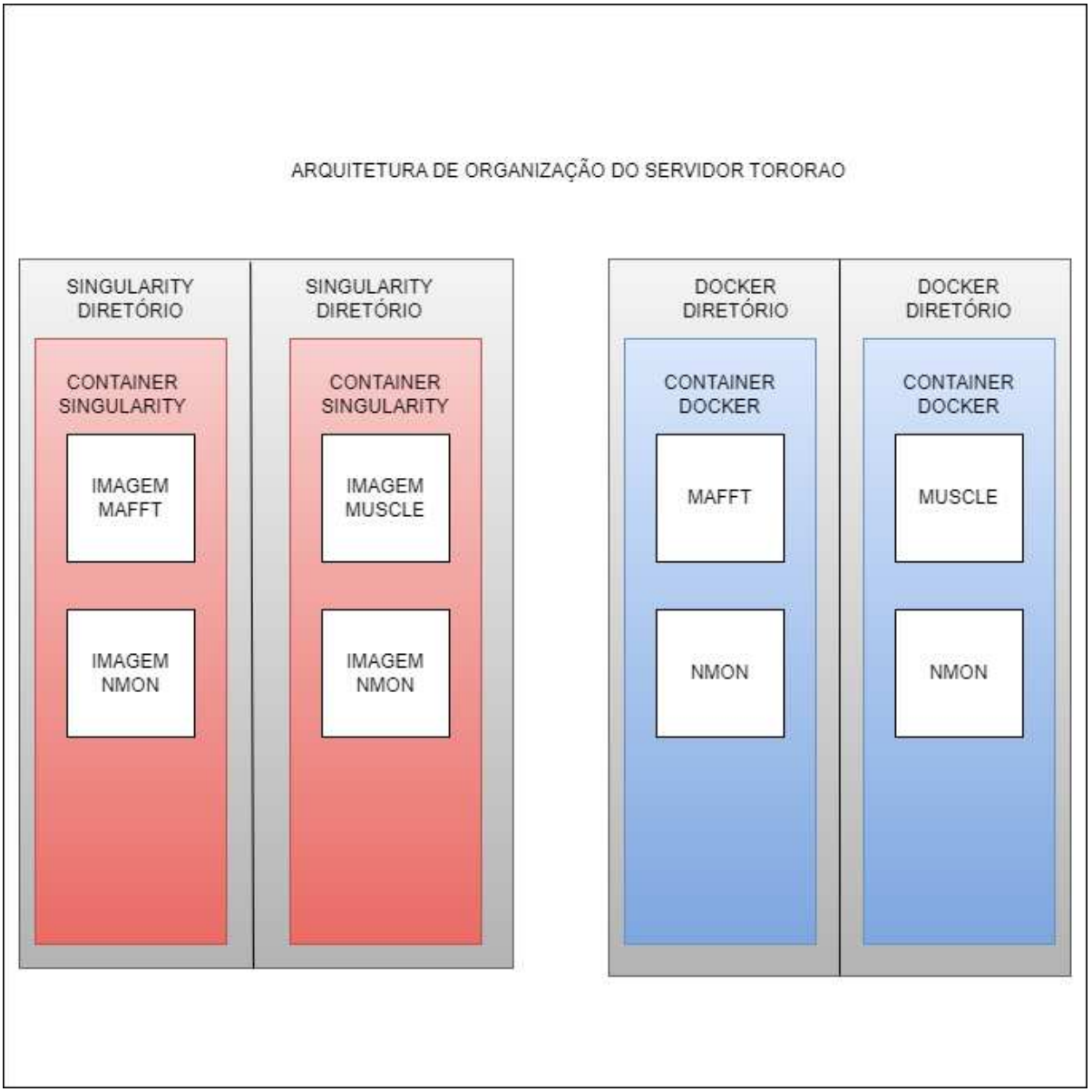
5. Exportar a imagem docker em um tipo de arquivo de formato tar (feito para arquivar dados baseados em unix).

6. Converter a imagem desejada do formato tar para o formato sif que é específico do singularity.

7. Por último criar o volume dentro do próprio container para receber as informações e parâmetros a respeito das ferramentas de alinhamento e de monitoramento.

Terminado todo o processo de configuração e criação de ambos os containers, partimos para nossa próxima etapa de cenários de testes na qual submetemos os containers a diversos testes de performance. Segue abaixo a figura 5 para elucidar a organização do servidor de testes.

Figura 4: Modelo de representação da arquitetura organizacional do servidor.



Fonte: o autor

4.6 DIFICULDADES PRESENCIADAS NO PROJETO

Durante todo o processo de instalação das ferramentas de virtualização do sistema operacional e, o próprio manuseio dos softwares instalados, houve complicações a serem levadas em conta. São eles:

- URLs de Páginas de conteúdo inválidas - Para se ter acesso ao conteúdo de um repositório necessita-se que as informações que direcionam ao mesmo estejam corretas e atualizadas. No entanto, durante as instalações do container singularity e da ferramenta Muscle tiveram problemas referentes ao endereçamento das páginas.

- Vários containers de diferentes imagens em um único diretório - Como foi informado previamente, cada ferramenta de alinhamento foi instalada em um diretório próprio para que suas respectivas imagens não sejam compartilhadas ou sobrepostas por outros containers por engano. Além de evitar um volume desnecessário de dados concentrado de dados no diretório.

- Versões Diferentes de uma ferramenta podem alterar suas linhas de comando - Nesse caso em específico se deveu no caso do Muscle que, primeiramente estava na sua versão: muscle v.3.8.31 e após uma atualização em sua imagem Dockerfile passou para: muscle v.5.1 o que trouxe novas mudanças de otimização mas também mudança de comandos mais básicos para realizar os alinhamentos.

- Tentar utilizar o container sem criar o volume - Sem um volume configurado, não é possível compartilhar dados, arquivos de entrada e saída, logs, ou qualquer outro tipo de dado que precise ser compartilhado entre o ambiente do container e o sistema host o que é fundamental para o nosso trabalho.

5.0 CONSTRUÇÃO DO CENÁRIO DE TESTES

A seção da construção do cenário de testes é fundamental para a compreensão do que foi analisado, assim como, a justificativa das suas ferramentas e métricas que são levadas em consideração. Portanto, a construção cuidadosa do cenário de testes é essencial para assegurar a validade, a relevância e a utilidade dos resultados obtidos em experimentos e avaliações de desempenho.

Primeiramente, é necessário o discernimento entre as duas ferramentas de alinhamento, muscle e o mafft, pois, apesar de realizarem as mesmas atividades, os algoritmos para atingir as etapas de cada uma são diferentes, o que pode impactar diretamente nas métricas do trabalho como tempos de execução e utilização de recursos.

O mafft é uma ferramenta que realiza múltiplos alinhamentos de sequência que operam sobre sequências de aminoácidos e nucleotídeos. A ferramenta realiza suas funções a partir de três etapas. São elas: a) construção de um guia de árvores - o primeiro passo consiste em construir um guia de árvores para as sequências. A árvore guia é uma representação da relação evolutiva entre as sequências; b) alinhamento de pares, no qual os pares de sequências são alinhados usando a FFT. A FFT é usada para identificar segmentos similares entre as sequências; c) o alinhamento múltiplo (feito com base na a árvore guia). A árvore guia é usada para refinar o alinhamento de pares e criar alinhamento múltiplo com base no refinamento.

Neste trabalho foi selecionado um dos algoritmos base para a fundação do mafft, o algoritmo de alinhamento progressivo (progressive alignment), que começa alinhando as sequências mais semelhantes entre si e, gradualmente, incorpora as sequências menos semelhantes podendo servir para alinhar grandes sequências.

A ferramenta dispõe de outras variedades de técnicas com base nessa estrutura de funcionamento operacional que podem ser encontradas na documentação da ferramenta MAFFT: Multiple sequence alignment based on the Fast Fourier Transform (KATOH, K; 2013).

O Muscle é uma é um software de alinhamento de sequências de aminoácidos desenvolvido pela empresa Muscle Software, fundamentalmente baseado nos algoritmos de múltiplos alinhamentos progressivos, que usa o método de Fourier (FFT) para alinhar sequências de aminoácidos. Assim como descrito anteriormente na ferramenta mafft, as mesmas etapas são realizadas mudando apenas o seu algoritmo.

Para a realização desse trabalho, e como foi estipulado no referencial teórico e que, atende as necessidades do grupo de pesquisa G2BC, o algoritmo selecionado do muscle foi o “Super 5” que usa uma variedade de técnicas para melhorar a precisão e a eficiência do alinhamento e principalmente um suporte a sequências de grandes comprimentos que normalmente a ferramenta não consegue comportar. A ferramenta trata de outras variedades para contar com diversos alinhamentos para mais informações a respeito da documentação da ferramenta MUSCLE v5 documentation (EDGAR.R.C, 2021).

Para realizar o cenário de testes, uma quantidade específica de arquivos do tipo fasta foi selecionada, dos mais variados tamanhos, para podermos quantificar e qualificar os nossos resultados com base na variedade e quantidade de sequências usadas. Todos os arquivos selecionados são especificamente apenas sobre sequências virais de dengue tipo 1 e tipo 2, com exceção dos arquivos Sequência Grande 1 e Sequência Grande 2 na qual contém, em sua composição, diversos tipos variados de sequências virais, com o intuito apenas de descobrir o máximo que uma ferramenta de alinhamento comporta de dados para realizar o alinhamento.

Segue abaixo, a tabela de arquivos mostrando os nomes dos arquivos e suas respectivas características de quantidade e comprimento de sequências.

Nome do arquivo	Quantidade de sequências e comprimentos
Sequence 20	20 sequências, entre 10.723 a 10.735
Sequence 50	50 sequências, entre 10.722 a 10.735
Sequence 100	100 sequências, entre 10.722 a 10.821
Sequence 200	200 sequências, entre 10.715 a 10.742
Sequência Grande 1	2541 sequências, entre 10.611 e 11.800
Sequência Grande 2	3668 sequências, entre 10516 a 15296

Fonte: elaborado pelo autor

Revelado as variáveis do cenário de teste, o experimento consistiu em importar para dentro do *container* individualmente cada um desses arquivos de sequência em conjunto com a ferramenta de monitoramento “nmon” para que pudesse ser feito a coleta de dados.

As coletas de dados pelo nmon seguiram as seguintes especificações: o intervalo entre as “amostras de dados” de utilização máxima de CPU, média de uso por núcleo de CPU, e média de uso de memória foi de um segundo por duas razões. A primeira é devido a natureza da volatilidade do processo de alinhamento, na qual não se sabe os momentos em que pode haver um pico de uso de recursos devido ao processo das ferramentas ou devido a interação do próprio kernel durante as atividades. O segundo ponto é a praticidade que um intervalo curto permite analisar os dados sem depender de preocupações de tempo que foi estabelecido caso o intervalo fosse considerável já que,o tempo estimado das aplicações é distinto e certos intervalos poderiam não ser compatíveis para a coleta.

E por fim, apesar da ferramenta nmon realizar monitoramentos em tempo real, e armazenamento dos dados, é necessário filtrar os dados que interessam para nosso cenário. Com base nessa questão foi necessária a criação de um *script* de uma linguagem nativa dos sistemas operacionais Unix chamada “AWK”, que opera em linhas de texto e divide cada linha em campos, permitindo que você defina padrões e ações associadas a esses padrões. Nesse *script*, foram realizados os cálculos médios e coleta direta dos nossos parâmetros.

6.0 ANÁLISE DE RESULTADOS

Nesta seção, são apresentadas as interpretações e implicações dos resultados. A análise é fundamentada nos dados quantitativos e qualitativos coletados durante o experimento/estudo, permitindo uma compreensão aprofundada do objeto de estudo.

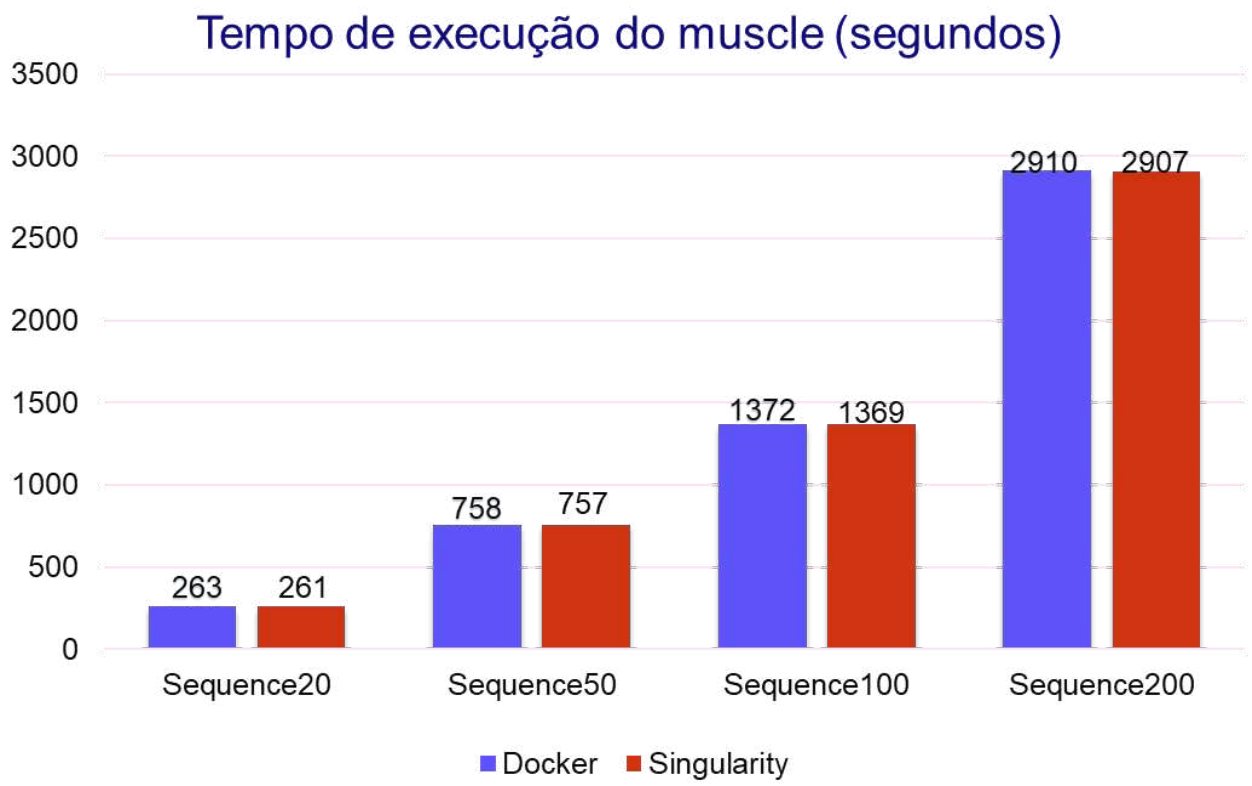
No segmento abaixo iremos fazer uma análise comparativa entre os diversos parâmetros coletados e inferir algumas observações e conclusões.

O primeiro *container* a ser analisado são os que possuem a ferramenta muscle. Vale ressaltar que os arquivos que não apareceram nos gráficos não conseguiram ser executados pelos alinhadores, devido ao comprimento da sequência ultrapassar o *buffer* de memória disponível e impossibilitar a operação, seja no começo ou no final do alinhamento.

Ao analisar o gráfico de representação do tempo de execução das sequências alinhadas percebe-se que não tiveram diferenças significativas entre o Docker e o Singularity. Apesar disso percebe-se que a principal razão que afeta o tempo de execução são as quantidades das sequências devido ao maior uso de recursos de alocação da memória e de CPU.

A conclusão a partir dessa análise é que, para a ferramenta muscle, o singularity mostrou uma quantidade de tempo menor para concluir as suas atividades, porém, não expressivamente relevante, como pode-se observar na figura a seguir:

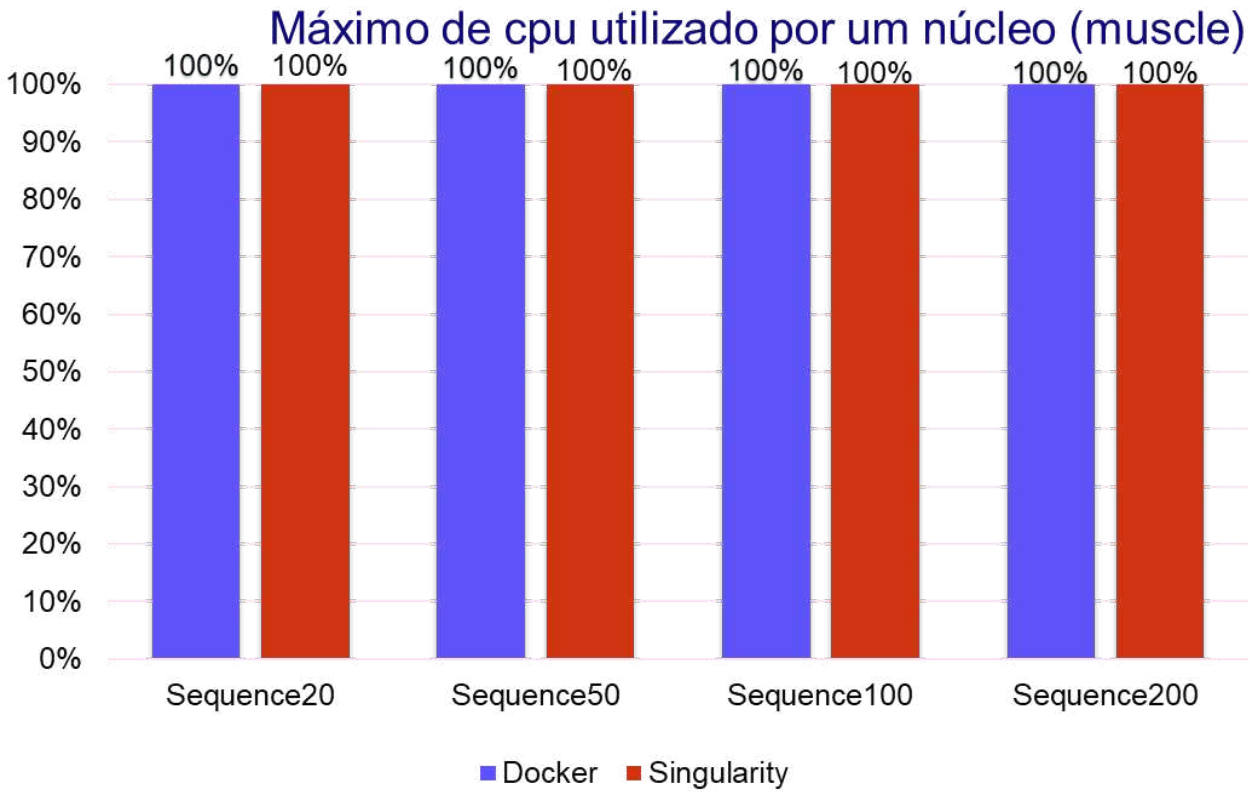
Figura 5: Comparação do tempo de execução entre os containers com o muscle.



Fonte: elaborado pelo autor

No gráfico a seguir, é analisado o máximo de CPU utilizada por algum núcleo de processador durante o período de alinhamento, consistindo em percorrer todas as linhas do arquivo salvo pela ferramenta “nmon” e, quando um novo valor de amostra é reconhecido como maior que o anterior, ele se torna o novo valor máximo. Constata-se que em ambos os *containers*, possuem uma taxa de utilização extremamente elevada igualitariamente.

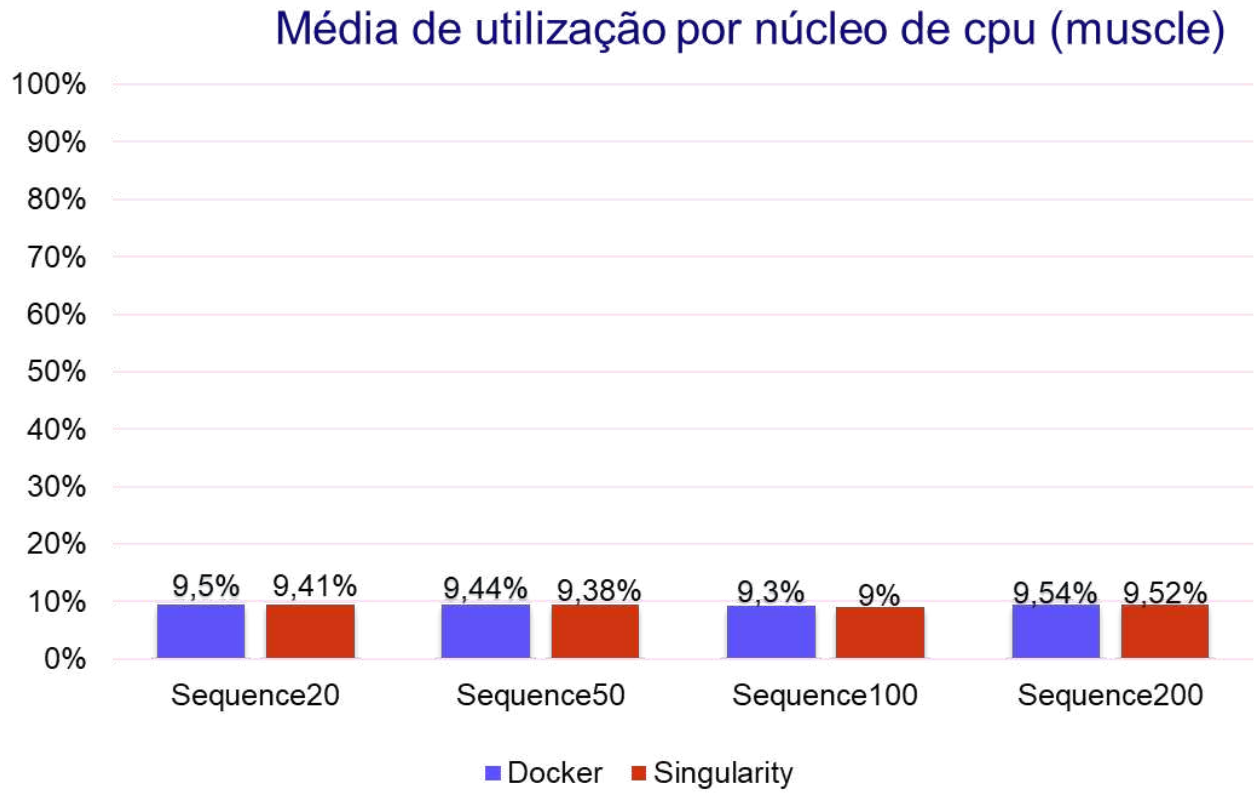
Figura 6: Comparação do máximo de cpu usado entre os containers com o muscle.



Fonte: elaborado pelo autor

No gráfico a seguir, é analisada a média por núcleo de processador utilizado durante o período de alinhamento. Esse procedimento consiste em percorrer todas as linhas do arquivo salvo pela ferramenta “nmon” e calcular a média dividindo o máximo utilizado de CPU pela quantidade de núcleos do total de todas as amostras coletadas. Neste gráfico, implica-se dizer que os núcleos não passaram sua carga de processamento adiante para um outro núcleo durante o período de alinhamento já que, nenhuma média ultrapassou 10% da média. Nesse quesito, o singularity teve um desempenho melhor, mas não significativo, com os arquivos de sequência que o docker.

Figura 7: Comparação da média por núcleo de processador entre os containers com o muscle.

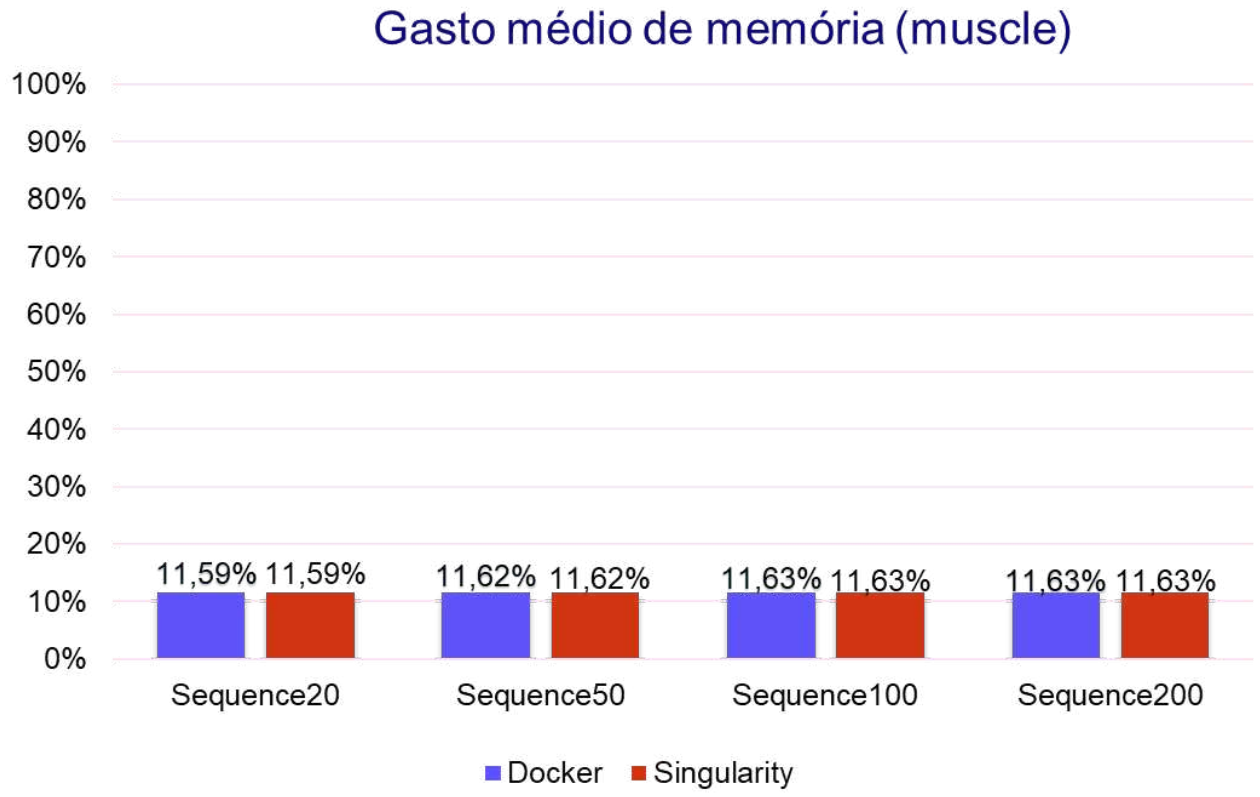


Fonte: elaborado pelo autor

Por fim, o gasto médio de memória durante esse procedimento consiste em percorrer todas as linhas do arquivo na coluna específica de memória utilizada, dividindo o total de memória pelo número de amostras realizadas, assim, encontrando o valor médio da memória. Depois, o valor médio da utilização da memória é dividido pelo total de memória instalada. O resultado é multiplicado por 100 para expressar a utilização média como uma porcentagem do total instalado.

Após a exibição dos resultados é constatado que em ambos os ambientes possuem uma taxa de utilização médio de memória idênticas e que durante o processo de alinhamento a memória entre ambos não foi relativamente muito utilizada.

Figura 8: Comparação do gasto médio de memória entre os containers com o muscle.



Fonte: elaborado pelo autor

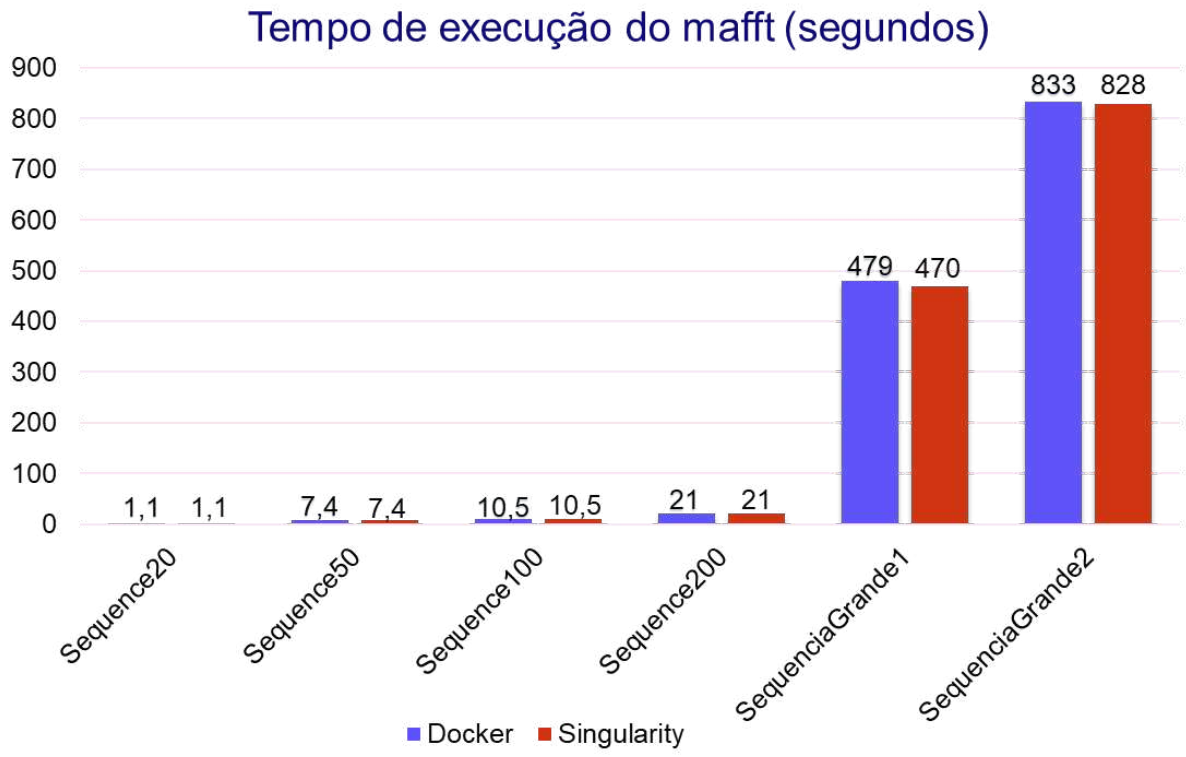
Agora iremos analisar os *containers* que contém a ferramenta mafft. Vale ressaltar que o mesmo *script* foi usado aqui para a obtenção dos resultados, então, as explicações a respeito de como está sendo realizado o procedimento, não serão repetidas.

O primeiro aspecto que vale ressaltar é a diferença da capacidade entre ambas as ferramentas de lidar com o acervo de dados, pois, no cenário dos ambientes de testes do mafft, mais arquivos conseguiram ser inicializados e implementados, uma vez que tiveram empecilhos no ambiente anterior.

O segundo aspecto é a enorme diferença no tempo de execução dos algoritmos do mafft e muscle, isso devido ao foco da proposta do alinhamento. Por exemplo, o algoritmo do mafft tem como objetivo alinhar sequências semelhantes entre si. Devido a isso, pressupõe-se que a velocidade do seu alinhamento foi devido a não encontrar semelhanças entre as amostras contidas dentro do arquivo e finalizá-lo preenchendo os espaços vazios com traços.

Comparando o desempenho entre os tempos de execução entre ambos os containers, nota-se que também não obteve-se muitas diferenciações entre os tempos encontrados, com o singularity ainda tendo uma execução minimamente superior com maiores quantidades de sequências.

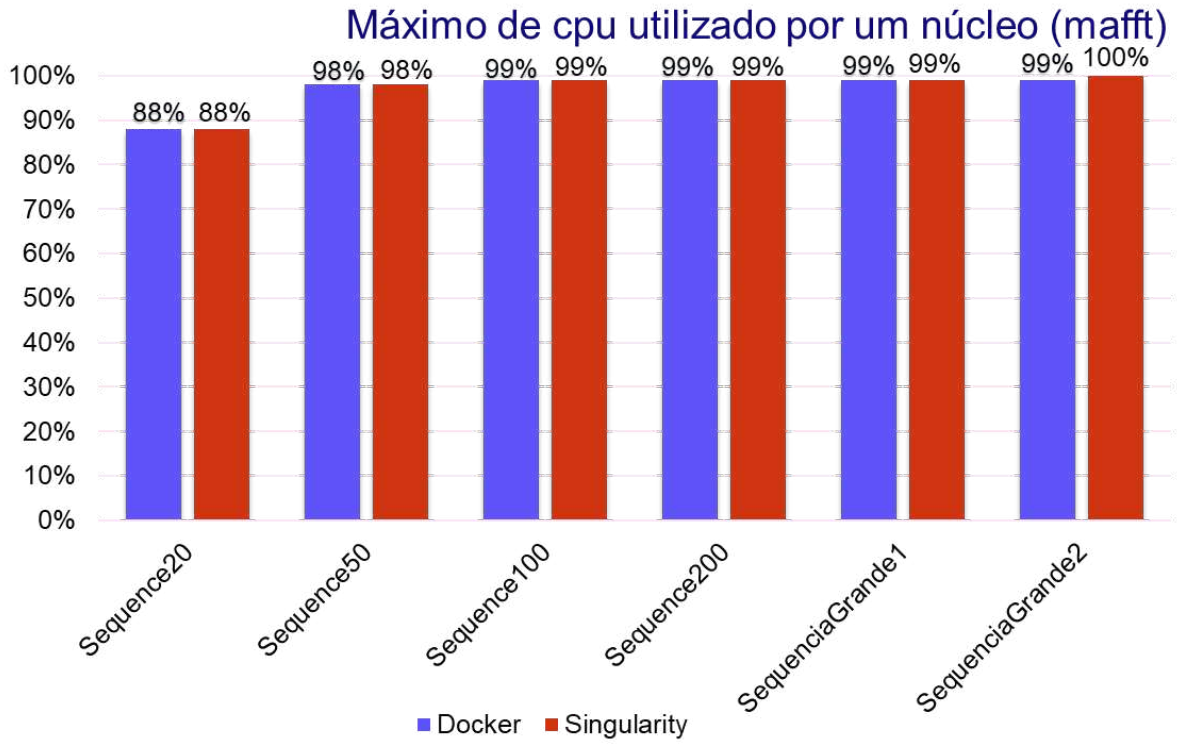
Figura 9: Comparação de tempo de execução entre os containers com o mafft.



Fonte: elaborado pelo autor

No gráfico a seguir é analisado o máximo de CPU da ferramenta mafft utilizada por ambos os containers em algum núcleo de processador durante o período de alinhamento. Como pode ser observado, ambos os *containers* possuem uma taxa de utilização extremamente elevada sem diferenciação nos ambientes observados, com exceção de um arquivo no qual o singularity atingiu o máximo de CPU.

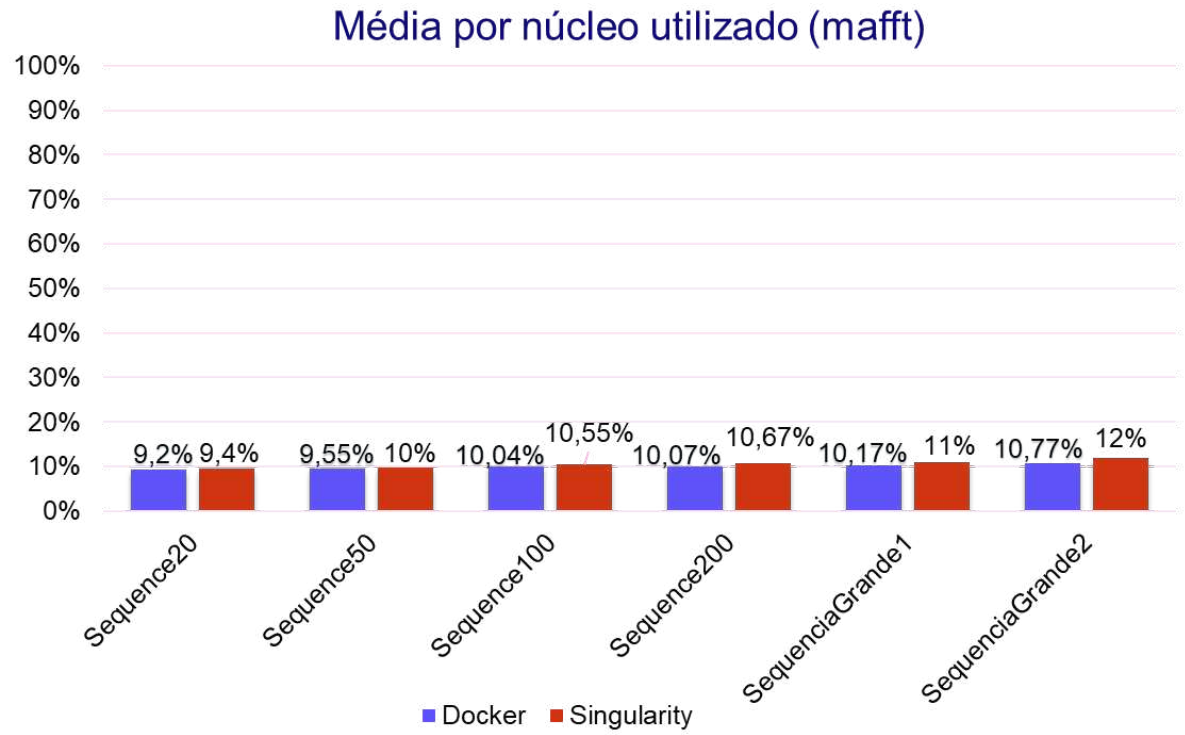
Figura 10: Comparação de máximo de cpu usado entre os containers com o mafft.



Fonte: elaborado pelo autor

Neste gráfico, entre a comparação média por núcleo, nota-se que os containers Docker tiveram um consumo menor individualmente em comparação aos containers singularity, inclusive, nesse caso, cinco arquivos passados para o singularity tiveram por um período de tempo um núcleo em sua totalidade máxima e parte da carga de trabalho de processamento foi passada para outro núcleo.

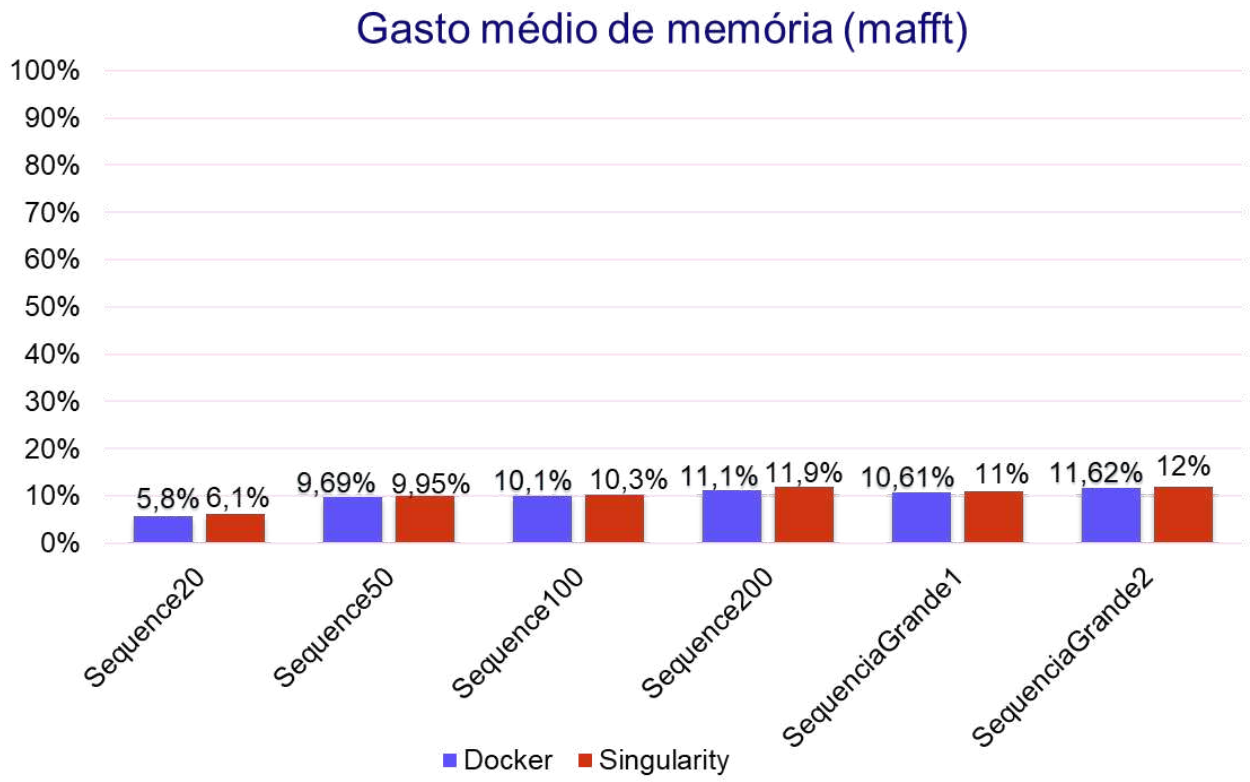
Figura 11: Comparação da média por núcleo de processador entre os containers com o mafft.



Fonte: elaborado pelo autor

Por fim, após a exibição dos resultados é constatado que, durante o processo de alinhamento, a memória entre ambos não foi muito utilizada. Além disso, os ambientes possuem uma taxa de utilização média baixa, entretanto, com uma leve vantagem do Docker, por utilizar menos a memória ao longo do período de alinhamento.

Figura 12: Comparação do gasto médio de memória entre os containers com o mafft.



Fonte: elaborado pelo autor

7.0 CONSIDERAÇÕES FINAIS

A aplicação da tecnologia dos *containers* ainda é recente na área de bioinformática, mas tem surgido muitos estudos e destaques pelo fato de ser uma área que lida com grandes volumes de dados biológicos e requer o uso de várias ferramentas e softwares especializados para análise e interpretação desses dados.

Apesar do uso de *containers* na bioinformática ter se popularizado recentemente, é relevante destacar a partir dos artigos selecionados que a área já tem se beneficiado da virtualização e do isolamento de aplicativos há algum tempo, utilizando tecnologias como as máquinas virtuais (VMs). No entanto, os *containers* apresentam uma abordagem mais eficiente, o que os torna especialmente indicados para a execução de análises bioinformáticas em larga escala e em ambientes distribuídos ou de alto desempenho.

Este trabalho se destaca por sua abordagem específica voltada para as ferramentas de alinhamento de bioinformática. Durante a revisão da literatura, identificamos uma lacuna significativa de pesquisa relacionada a qual ambiente de container oferece um desempenho superior nesse contexto. Ao contrário de trabalhos correlatos que se concentram em testes de estresse para descobrir o potencial máximo suportado pelos ambientes, nossa pesquisa direciona-se para uma análise mais refinada e contextualizada do desempenho, especialmente no que diz respeito às demandas específicas das ferramentas de alinhamento bioinformático. Ao abordar essa lacuna, buscamos contribuir para o entendimento mais aprofundado das nuances do desempenho em ambientes de container, proporcionando *insights* valiosos para aprimorar a eficiência e a eficácia das ferramentas de alinhamento na área de bioinformática.

Este trabalho de monografia realizou um estudo comparativo de desempenho entre dois container mais usados no mercado Docker e singularity, no qual uma metodologia experimental foi proposta com etapas bem definidas para a avaliação de ambientes que utilizam tecnologias de bioinformática distintas. O container Docker apresentou tempos de execução e média de utilização de núcleo de CPU minimamente maiores que o container singularity para a ferramenta de alinhamento do muscle. Em contrapartida, apresentou tempos menores com relação a média por núcleo e média de memória gasta para com a ferramenta mafft.

Esses resultados minimamente destoantes podem estar ligados a como cada ferramenta reage dentro da organização estrutural do container e/ou do próprio kernel vigente do sistema. Apesar disso, as análises foram satisfatórias e precisas pelo fato de conceitualmente os containers

serem capazes de replicar os mesmos resultados ou próximos a eles com o máximo de eficácia possível se possuírem as mesmas características de imagem. Neste trabalho as maiores diferenças constatadas não foram os dados impressos pelos containers mas sim, pelas ferramentas de alinhamentos e os algoritmos selecionados, culminando em resultados diferentes entre as ferramentas Muscle e Mafft.

Diante da equivalência identificada nas métricas de desempenho e da adaptabilidade comprovada de nossas aplicações em ambos os ambientes, concluímos que o Grupo de Pesquisa G2BC está em posição de utilizar qualquer um dos containers, Docker ou Singularity, de acordo com as necessidades específicas de cada projeto. Esta flexibilidade na escolha proporciona liberdade para considerar fatores adicionais, como preferências da equipe, ecossistema, facilidade de integração e custos associados, contribuindo para uma decisão informada e alinhada com os objetivos de nossas pesquisas.

Durante a execução deste trabalho, foi constatada uma dificuldade para encontrar artigos científicos que constasse uma análise comparativa entre diferentes tipos de *containers* para ferramentas de bioinformática. Em contrapartida, foram encontrados diversos artigos comparativos entre *containers* e máquinas virtuais para workflows de bioinformática.

Esperamos, com este trabalho, contribuir para a comunidade científica, trazendo mais análises de qual ambiente se adequa mais às necessidades de determinadas ferramentas, conforme as plataformas demandam.

8 REFERÊNCIAS

AL-ABSI, A. A.; KANG, D.-K. Long Read Alignment with Parallel MapReduce Cloud Platform. **BioMed Research International**, v. 2015, p. 1–13, 2015.

BARIK, R. K. et al. Performance Analysis of Virtual Machines and Containers in Cloud Computing. **2016 International Conference on Computing, Communication and Automation (ICCCA)**, abr. 2016.

BERNSTEIN, D. Containers and Cloud: From LXC to Docker to Kubernetes. **IEEE Cloud Computing**, v. 1, n. 3, p. 81–84, set. 2014.

DA VEIGA LEPREVOST, F. et al. BioContainers: an open-source and community-driven framework for software standardization. **Bioinformatics**, v. 33, n. 16, p. 2580–2582, 30 mar. 2017.

DAUGELAITE, J.; O' DRISCOLL, A.; SLEATOR, R. D. **An Overview of Multiple Sequence Alignments and Cloud Computing in Bioinformatics**. Disponível em: <https://www.hindawi.com/journals/isrn/2013/615630/>.

Docker. Docker Engine. Install Docker Engine on Ubuntu. 2023-11-24. Disponível em: https://docs.docker.com/engine/install/ubuntu/. Acesso em: 2023-11-24.

DUA, R.; RAJA, A. R.; KAKADIA, D. Virtualization Vs Containerization to Support PaaS. **2014 IEEE International Conference on Cloud Engineering**, mar. 2014.

GANTIKOW, H. **Rootless Containers with Podman for HPC**. [s.l: s.n.]. Disponível em: <https://vhpc.org/static/PapersPresentations2020/iscworkshops2020_paper_12.pdf>. Acesso em: 26 nov. 2023.

GAUTHIER, J. et al. A brief history of bioinformatics. **Briefings in Bioinformatics**, v. 20, n. 6, p. 1981–1996, 27 nov. 2019.

GERLACH, W. et al. Skyport: container-based execution environment management for multi-cloud scientific workflows. **In: Proceedings of the 5th international workshop on data-intensive computing in the clouds.** DataCloud'14. Piscataway: IEEE Press,25–32. 2015.

GORDIN, I; GRAUR, A; ADOMNITEI, C. Web portal development with different cloud containers: Docker vs. Kubernetes. **Journal of Applied Computer Science & Mathematics**, Issue 2/2021, vol.16, no. 32, Suceava . 2021.

HAGEN, J. B. The origins of bioinformatics. **Nature Reviews Genetics**, v. 1, n. 3, p. 231–236, 1 dez. 2000.

JHA, D. N. et al. A study on the evaluation of HPC microservices in containerized environment. **Concurrency and Computation: Practice and Experience**, p. e5323, 2 maio 2019.

KADRI, S. et al. Containers in Bioinformatics: Applications, Practical Considerations, and Best Practices in Molecular Pathology. **The Journal of Molecular Diagnostics**, 18 fev. 2022.

KATOH, K. **MAFFT ver.7 - a Multiple Sequence Alignment Program**. Disponível em: <https://mafft.cbrc.jp/alignment/software/algorithms/algorithms.html>.

KULKARNI, N. et al. Reproducible bioinformatics project: a community for reproducible bioinformatics analysis pipelines. **BMC Bioinformatics**, v. 19, n. S10, out. 2018.

KUMAR, R.; THANGARAJU, B. **Performance Analysis between RunC and Kata Container Runtime**. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9198653/>. Acesso em: 11 set. 2022.

KWON, C.-H.; KIM, J. K.; AHN, J. Web Application for Efficient Use of Bioinformatics Docker Images. **DockerBIO**, 27 nov. 2018.

MARTIN, A. et al. Docker Ecosystem – Vulnerability Analysis. **Computer Communications**, v. 122, p. 30–43, jun. 2018.

PAHL, C. et al. Cloud Container Technologies: A State-of-the-Art Review. **IEEE Transactions on Cloud Computing**, v. 7, n. 3, p. 677–692, 1 jul. 2019.

PEREZ‐RIVEROL, Y.; MORENO, P. Scalable Data Analysis in Proteomics and Metabolomics Using BioContainers and Workflows Engines. **PROTEOMICS**, v. 20, n. 9, p. 1900147, 18 dez. 2019.

PEVSNER, J. **Bioinformatics and Functional Genomics**. Chichester, West Sussex: Wiley Blackwell, 2015.

RITCHIE, M. D. et al. Methods of integrating data to uncover genotype–phenotype interactions. **Nature Reviews Genetics**, v. 16, n. 2, p. 85–97, 13 jan. 2015.

RYAN CHRISTOPHER EDGAR. High-accuracy Alignment Ensembles Enable Unbiased Assessments of Sequence Homology and Phylogeny. 21 jun. 2021.

Singularity Team. Singularity: A Lightweight Containerization Platform for High Performance

Computing. 2023-08-02. Disponível em: https://sylabs.io/singularity/. Acesso em: 2023-08-24.

VADDI SESHAGIRI RAO et al. Recent Developments in Life Sciences research: Role of Bioinformatics. **African Journal of Biotechnology**, v. 7, n. 5, p. 495–503, 4 mar. 2008.

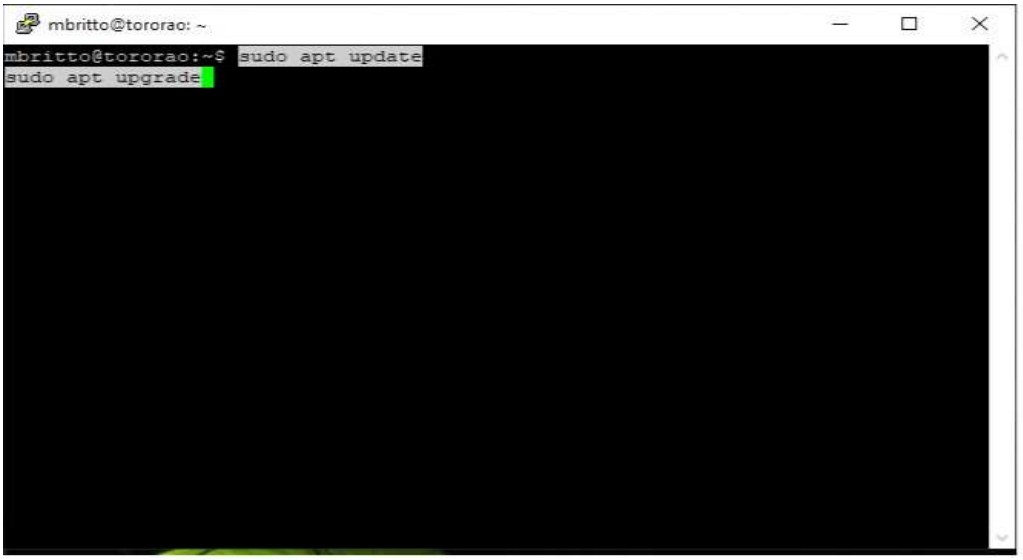
XAVIER, Miguel G.; OLIVEIRA, Israel C. De; ROSSI, Fabio D.; *et al*. A Performance Isolation Analysis of Disk-Intensive Workloads on Container-Based Clouds. 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2015. Disponível em: <https://www.academia.edu/88538682/A_Performance_Isolation_Analysis_of_Disk_Intensive_Workloads_on_Container_Based_Clouds>. Acesso em: 8 dez. 2023.

YADAV, R. R.; SOUSA, E. T. G. ; CALLOU, G. R. A. Performance Comparison between Virtual Machines and Docker Containers. IEEE Latin America Transactions, v. 16, n. 8, p. 2282–2288, 2018.

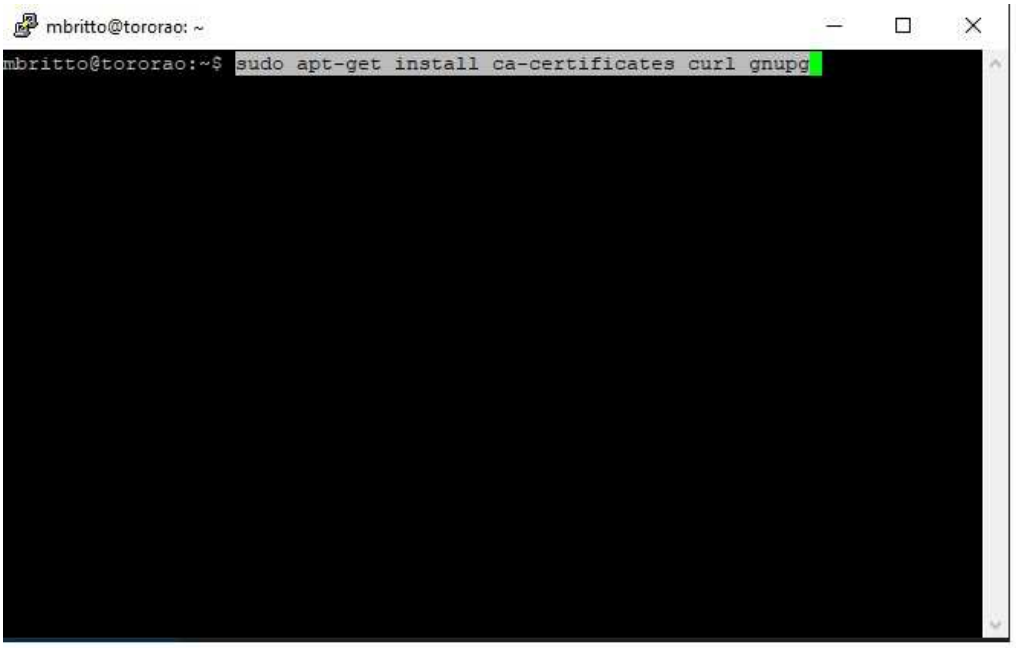
APÊNDICES

APÊNDICE A - CONFIGURAÇÃO DOS CONTAINERS DOCKER

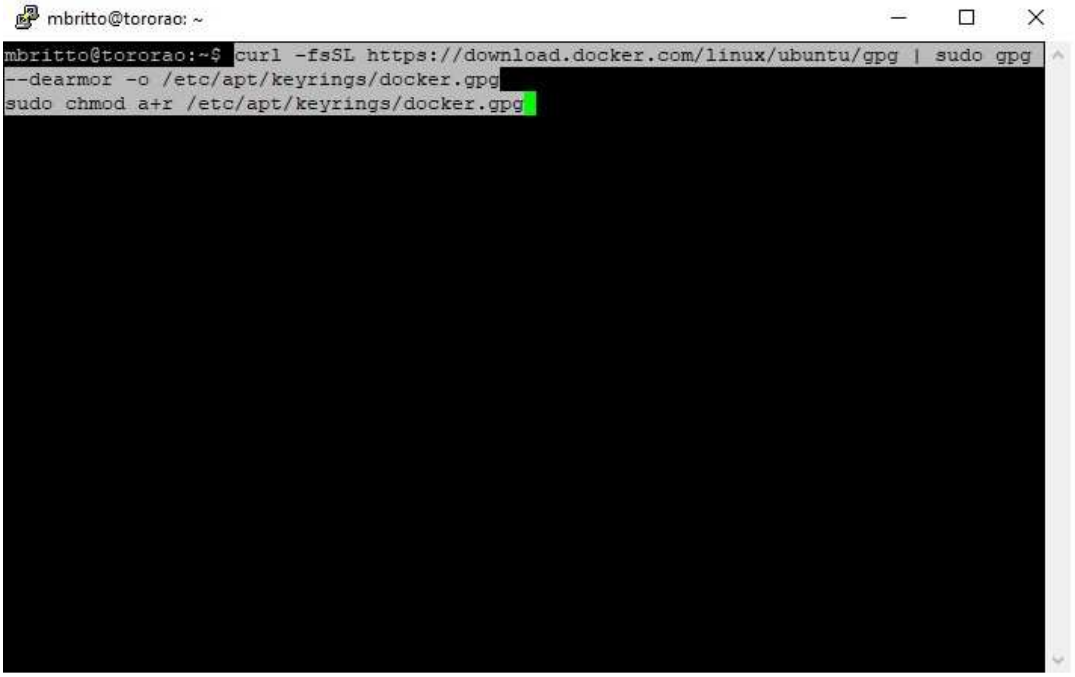
Atualização do sistema



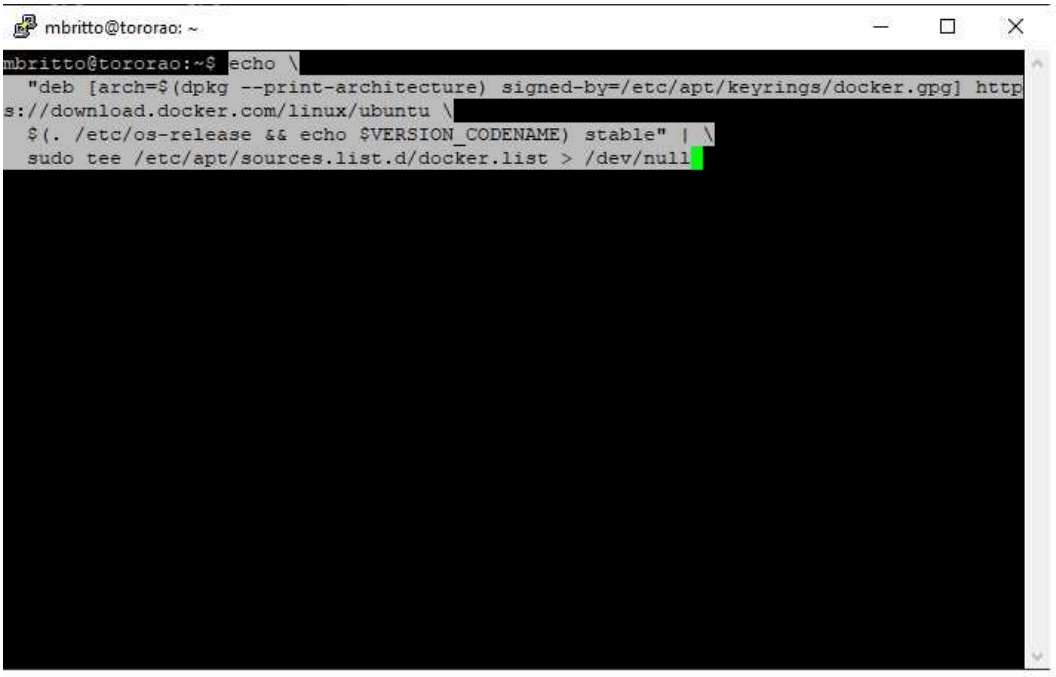
Instalação de pacotes e dependências



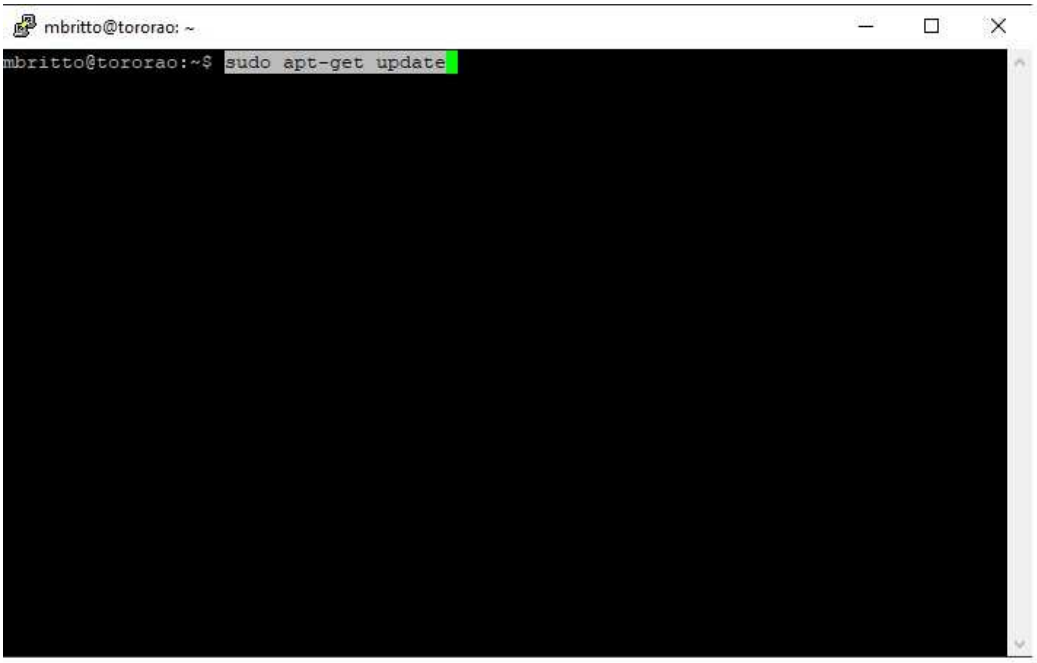
Para adicionar a chave GPG oficial do Docker



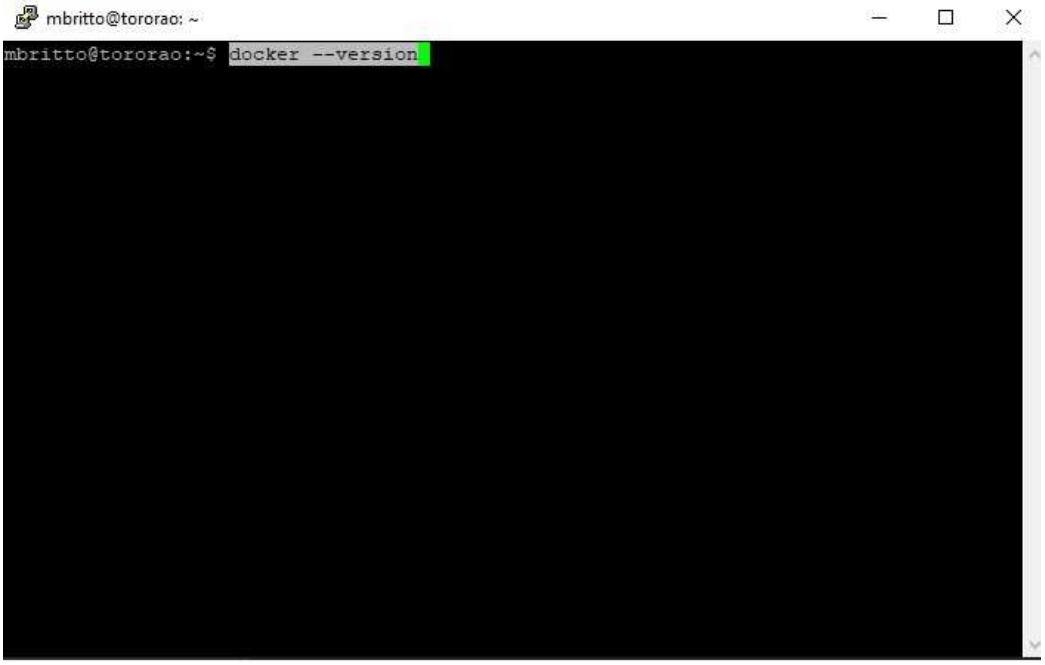
Adicionar o repositório ao sistema



Atualiza a lista de pacotes disponíveis no sistema a partir dos novos repositórios adicionados

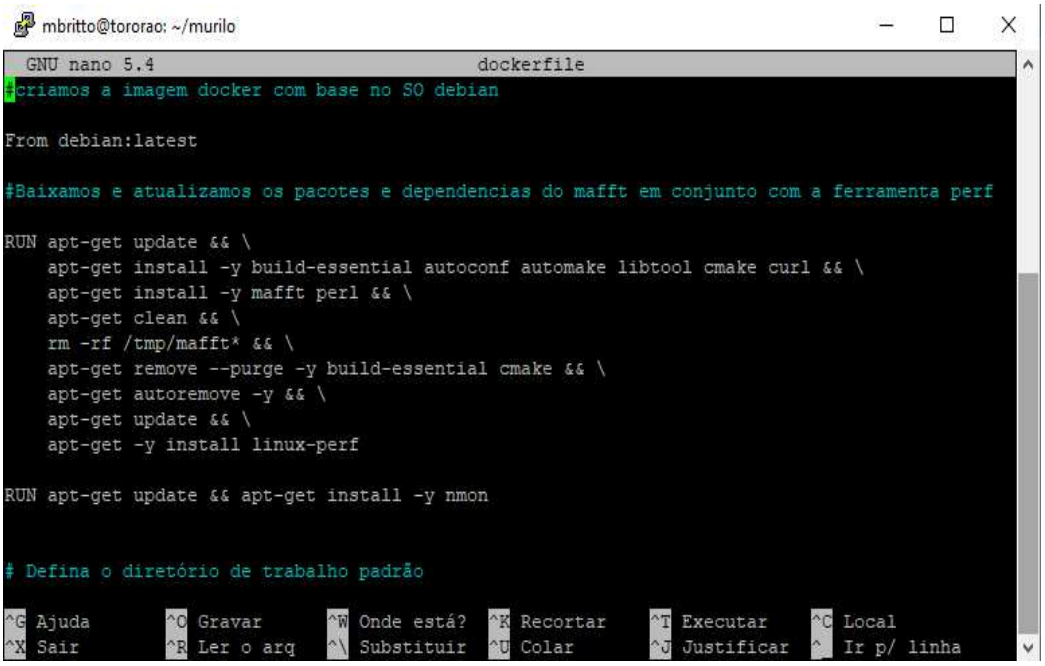


Para verificar se o Docker foi instalado



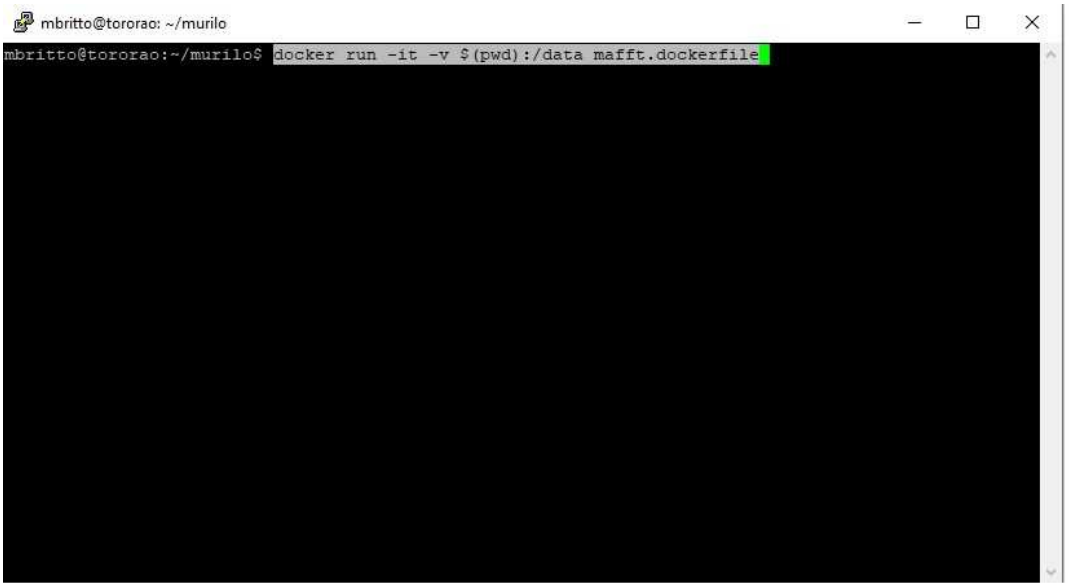
Para criar um arquivo do tipo dockerfile

Lista de dependências e os programas Mafft e nmol para serem instalados no arquivo

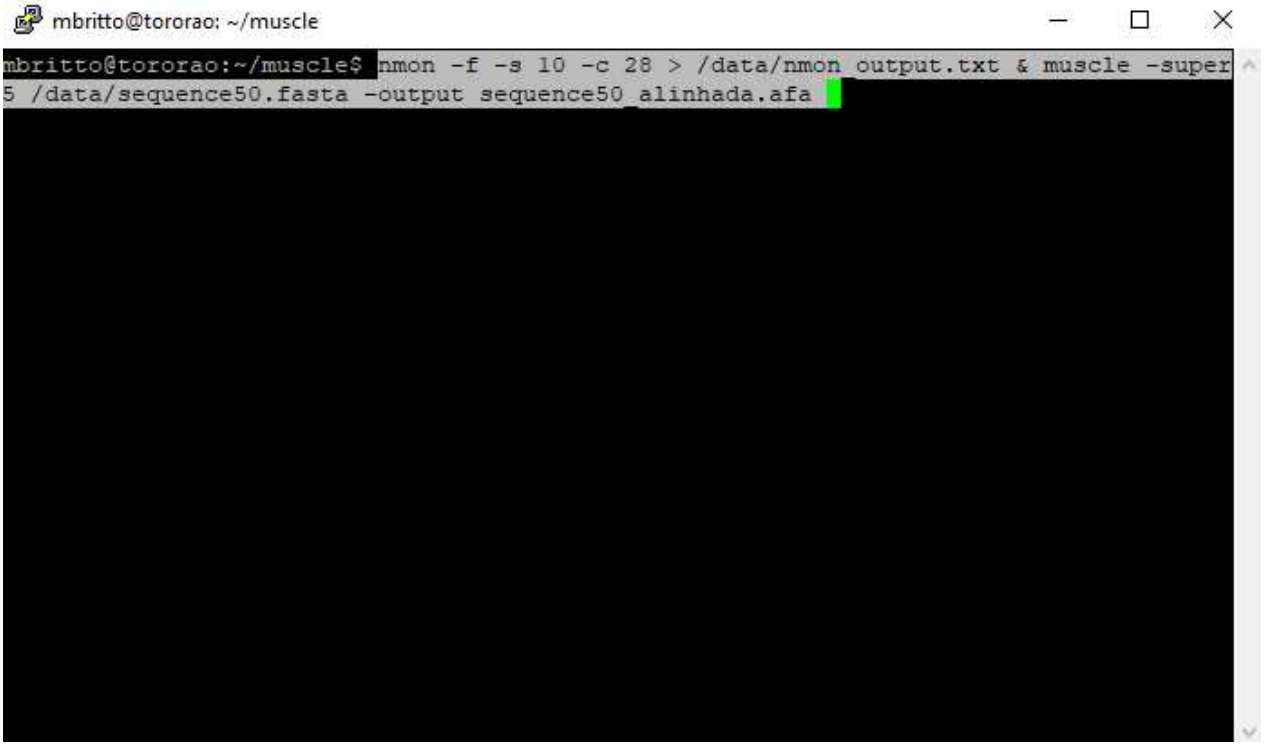


Para construir a imagem a partir do arquivo dockerfile

Para criar o volume dentro do container

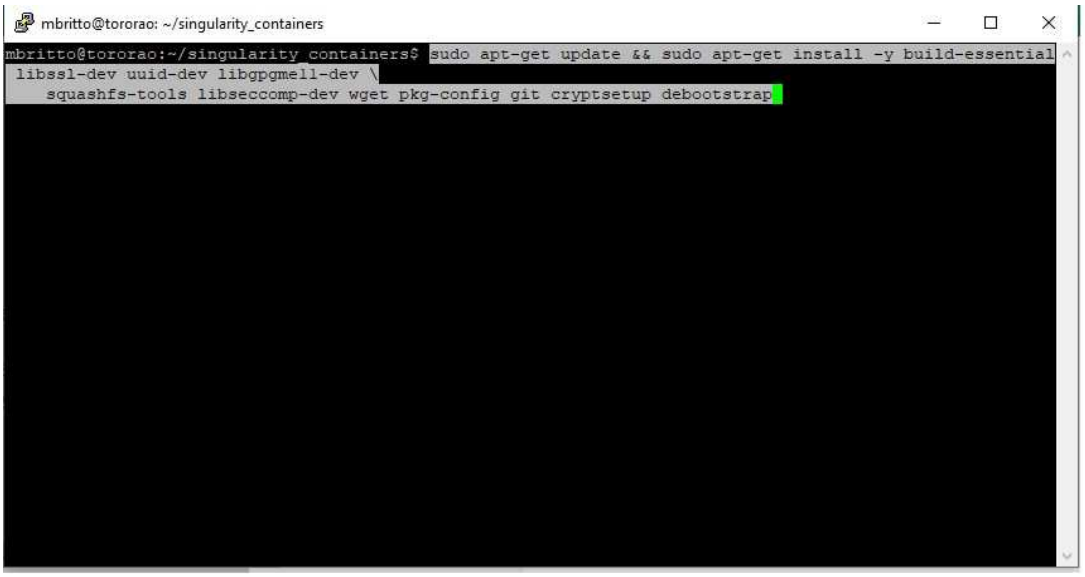


Comandos para operar ambas as ferramentas instaladas

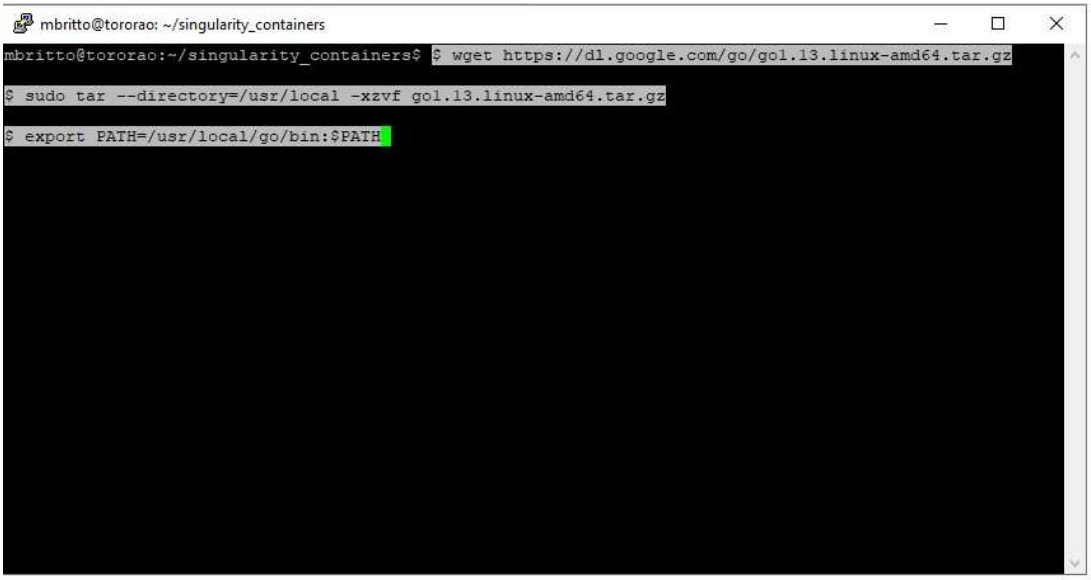


APÊNDICE B - CONFIGURAÇÃO DOS CONTAINERS SINGULARITY

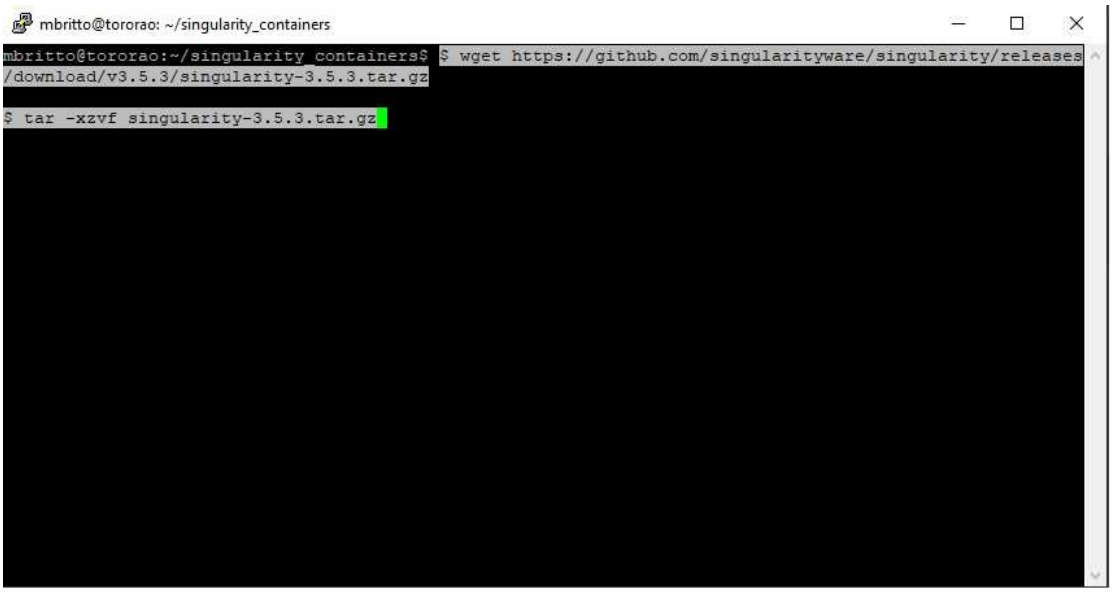
Atualiza a lista de pacotes disponíveis e, em seguida, instala os pacotes listados em seu sistema



Comandos para baixar o Go compactado e descompactar no diretório local e o adicionando ao Path do sistema

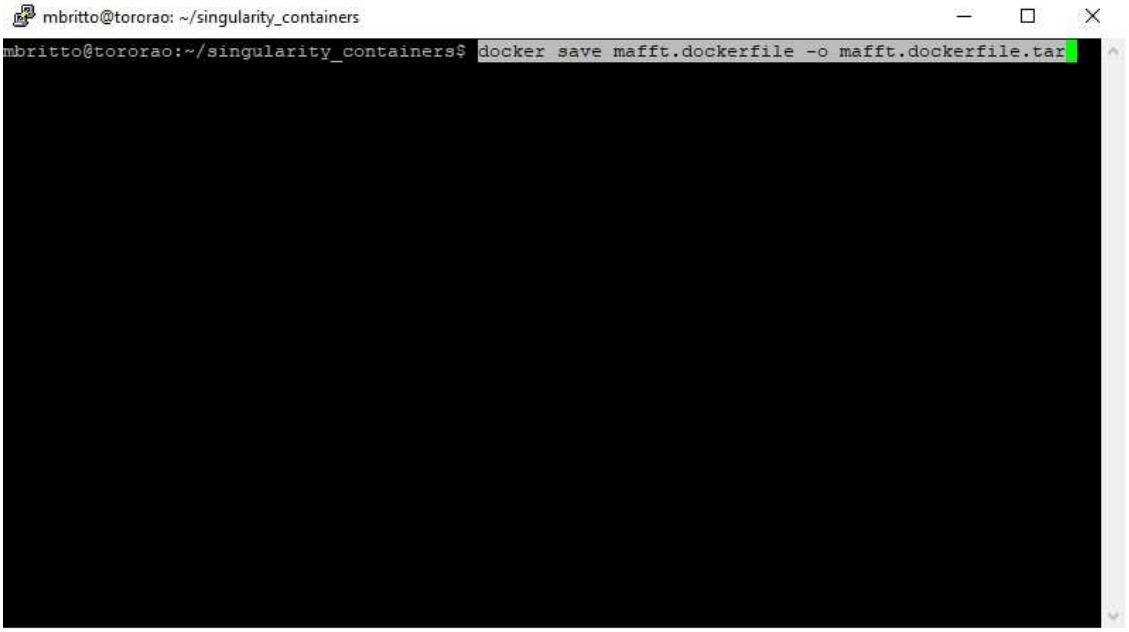


Comando para clonar o repositório do Singularity a partir do GitHub

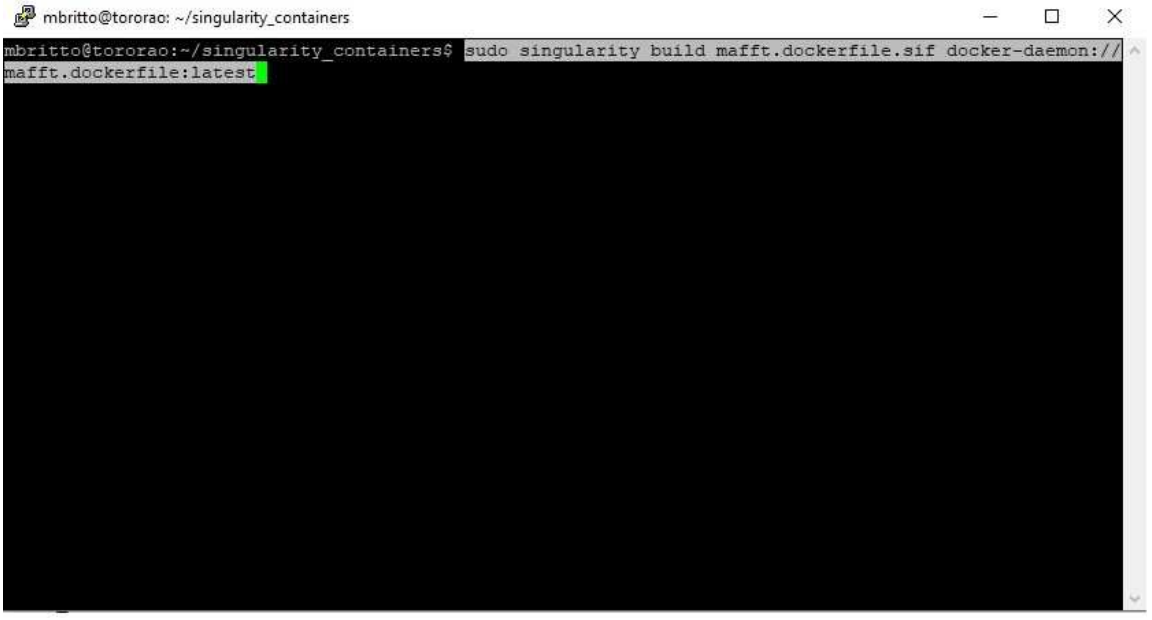


Construa e instale o Singularity no diretório

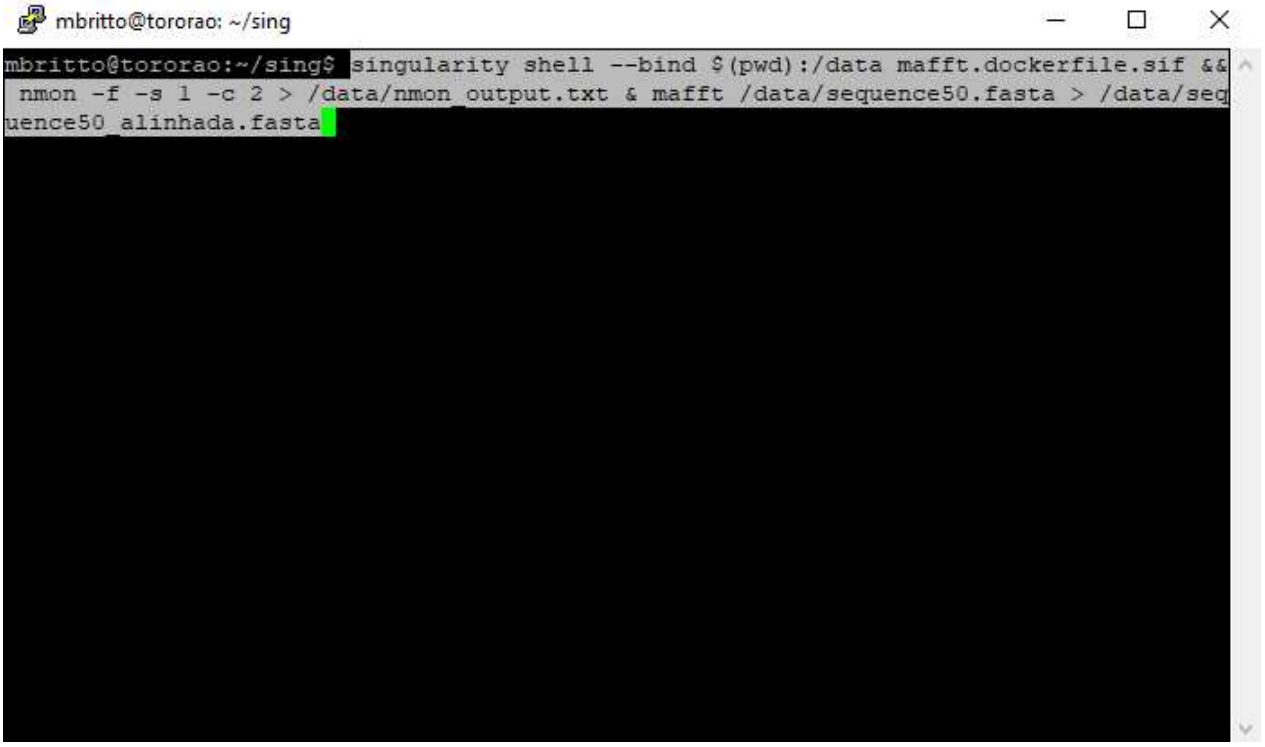
Para exportar uma imagem docker existente e transformando-a em tipo tar



Para converter a imagem em tipo Sif e construir a imagem



Comando para executar o container criando volume e informando os parâmetros que serão executados dentro do mesmo.



Script de filtro de comando do AWK.