UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

FLÁVIO BORGES DE LIMA

IMPLANTAÇÃO DE CLUSTER KUBERNETES EM COMPUTADORES COM PODER COMPUTACIONAL LIMITADO

GUARAPUAVA

2024

FLÁVIO BORGES DE LIMA

IMPLANTAÇÃO DE CLUSTER KUBERNETES EM COMPUTADORES COM PODER COMPUTACIONAL LIMITADO

Deployment of Kubernetes Cluster for Computer Netowork with Limited Computational Power

Projeto de Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Hermano Pereira

Coorientador: Profª. Drª. Sediane Carmem Lunardi Hernandes

GUARAPUAVA

2024

RESUMO

Este trabalho é o projeto de uma alternativa econômica, ecológica e flexível para reutilizar *hardware* de poder computacional limitado, mitigando os custos e impactos ambientais associados à aquisição de novos equipamentos. A projeto consiste em transformar esse *hardware* limitado em um cluster Kubernetes funcional capaz de aproveitar ao máximo seus recursos limitados. O objetivo é avaliar sua efetividade por meio de testes práticos, promovendo uma solução sustentável e acessível para a reutilização tecnológica.

Palavras-chave: kubernetes; clusters; recursos limitados; computação distribuída; otimização de desempenho.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

CPU	Unidade de Processamento Central, do inglês *Central Prossessing Unit*
HA	Alta Disponibilidade, do inglês *High Availability*
HPC	Computação de Alta Performance, do inglês *High-Performance Computing*
IoT	Internet das Coisas, do inglês *Internet of Things*
LAN	Rede de Área Local, do inglês *local area network*
NAT	tradução de endereços de rede , do inglês *Network address translation*
OS	Sistema Operacional, do inglês *Operating System*
PEAP	Protocolo de Autenticação Extensível Protegido, do inglês *Protected Extensible Authentication Protocol*
RAM	Memória de acesso aleatório, do inglês *Random Access Memory*
TCC	Trabalho de Conclusão de Curso

SUMÁRIO

1 INTRODUÇÃO

O conceito de clusters tem se tornado fundamental em ambientes que exigem processamento distribuído (MONTEIRO *et al.*, 2021). Um cluster é formado por um conjunto de computadores que trabalham de forma colaborativa, aumentando a capacidade de processamento ao distribuir tarefas entre os vários computadores desse cluster (IBM, 2024). Isso permite que sistemas computacionais com recursos limitados, como computadores antigos, possam ser reaproveitados, sem custos adicionais consideráveis, em configurações que maximizam o desempenho do sistema como um todo. O conceito de computador com poder computacional limitado neste trabalho é definido como computadores com recursos modestos (isto é, computadores que possuem entre 2 e 4 gigabytes de RAM, processadores com 2 a 4 núcleos de CPU com arquitetura x86-64) em relação aos requisitos típicos das aplicações modernas.

Aplicações web podem ser distribuídas, permitindo que o armazenamento e o processamento de dados sejam divididos entre os vários computadores em uma rede ou cluster de computadores. De forma mais abrangente, aplicações que seguem o paradigma cliente-servidor ou peer-to-peer podem se beneficiar do uso de clusters de computadores para um sistema distribuído.

Com tecnologias como *containers*, que isolam aplicações e garantem a portabilidade entre diferentes sistemas, e o Kubernetes, que orquestra e gerencia esses *containers* (CNCF, 2024b), torna-se possível utilizar clusters Kubernetes para diversas finalidades, como mencionado em (MONTEIRO *et al.*, 2021), p. 154, "em seu cluster próprio (de forma local), em um sistema de arquitetura híbrida ou até mesmo em qualquer provedor de computação na nuvem pública.". Um cluster de computadores é considerado um sistema de arquitetura híbrido. Logo, um cluster Kubernetes pode ser implantado considerando essa arquitetura.

Desta forma, este projeto busca explorar cluster Kubernetes em computadores com capacidade computacional limitada para aproveitar ao máximo o sistema computacional como um todo.

1.1 Objetivos

Os objetivos deste trabalho se dividem em objetivos gerais e específicos.

1.1.1 Objetivo geral

Este projeto tem como objetivo geral avaliar o desempenho de um cluster Kubernetes, para aplicações baseadas em microsserviços, o qual será implantado em máquinas com poder computacional limitado.

1.1.2 Objetivos específicos

Como objetivos específicos deste trabalho tem-se:

- Montar uma rede de computadores utilizando computadores com capacidade computacional limitada.
- Implantar um cluster Kubernetes na rede de computadores criada.
- Avaliar o desempenho do cluster Kubernetes criado, considerando suas limitações computacionais, utilizando aplicações simples baseadas em microsserviços, como aplicações web, por exemplo.
- Documentar o processo de configuração e implantação do cluster Kubernetes.
- Montar os gráficos resultantes das avaliações de desempenho (por exemplo, uso de Unidade de Processamento Central, do inglês *Central Prossessing Unit* (CPU), memória, rede e tempo de resposta) realizadas.

1.2 Justificativa

Com o avanço da tecnologia, muitos computadores tornam-se obsoletos e são descartados, apesar de ainda possuírem capacidade de processamento significativa para determinados tipos de aplicações. Em contra partida, a aquisição de *hardware* com capacidade computacional superior pode ser extremamente custosa.

A implementação de clusters Kubernetes em computadores com poder computacional limitado, oferece uma alternativa sustentável e de baixo custo para o uso desses equipamentos. Isso permite que através da distribuição de carga de trabalho, esses computadores trabalhem de forma colaborativa, aumentando sua eficiência. Além disso, o uso de containeres e orquestração com Kubernetes proporciona flexibilidade e escalabilidade, permitindo que o cluster seja utilizado para vários fins, desde projetos acadêmicos até experimentos práticos. Porém este Trabalho de Conclusão de Curso (TCC) vai focar na utilização do *cluster* para aplicações *web* distribuídas.

Assim, este projeto se justifica, portanto, pela relevância de explorar soluções tecnológicas sustentáveis e economicamente viáveis, reutilizando *hardware* ultrapassado, e pela necessidade de oferecer um ambiente prático, acessível e adaptável para estudantes e profissionais que desejam testar, desenvolver ou implementar suas aplicações em um cluster de computadores de baixo custo.

1.3 Estrutura do trabalho

O trabalho está dividido como segue. O capítulo 1 contextualiza o problema e apresenta os objetivos do projeto, a justificativa para o seu desenvolvimento e a estrutura geral do trabalho. O capítulo 2 refere-se ao Referencial Teórico, o qual aborda os principais conceitos e tecnologias utilizados no projeto, como clusters, *containers*, Kubernetes e Docker, além de discutir os desafios de uso de *hardware* ultrapassado em ambientes distribuídos. O capítulo 3 descreve os trabalhos relacionados e no capítulo 4 são citados alguns dos *software*, *hardware* e metodologias de testes que serão usados no decorrer do TCC. No capítulo 5, a projeto deste TCC é apresentado e no capítulo 6, o cronograma a ser seguido. Por fim, as referências bibliográficas são apresentadas.

2 REFERENCIAL TEÓRICO

Neste capítulo, serão apresentados os conceitos fundamentais relacionados ao desenvolvimento do trabalho, incluindo clusters, contêineres, orquestração de *containers* e Kubernetes, com foco na implementação de clusters Kubernetes em máquinas com poder computacional limitado.

2.1 Clusters

Segundo a IBM, clusters são aglomerados de computadores que trabalham juntos para executar tarefas, distribuindo a carga de trabalho entre os computadores que formam a cluster(IBM, 2024).

Cada computador em um cluster é chamado de nó (*node*), e cada nó é formado por um Sistema Operacional, do inglês *Operating System* (OS) e um *middleware*, que é responsável por gerenciar a comunicação e a execução de tarefas no cluster. A estrutura de um cluster pode variar entre um único nó até milhares de nós, geralmente conectados por uma Rede de Área Local, do inglês *local area network* (LAN) (IBM, 2024).

Os clusters podem ser divididos em dois grupos principais:

- Computação de Alta Performance, do inglês *High-Performance Computing* (HPC): Projetados para executar tarefas que precisam de um grande poder computacional, como simulações científicas ou processamento de grande volume de dados.
- Alta Disponibilidade, do inglês *High Availability* (HA): Desenvolvidos para garantir que os serviços e recursos permaneçam acessíveis, mesmo em caso de falhas em um ou mais nós, geralmente sendo utilizados na hospedagem de aplicações.

Neste TCC, será utilizado um cluster de HA, gerenciado pelo Kubernetes, que será explorado em capítulos posteriores. A escolha do Kubernetes foi feita com base na facilidade de adicionar novos nós (CNCF, 2024b) e na grande flexibilidade fornecida pelos contêineres (GOOGLE, 2024a), que são gerenciados pelo Kubernetes.

2.2 Contêineres

Como descrito por (GOOGLE, 2024a), "os contêineres são pacotes de software que contêm todos os elementos necessários para serem executados em qualquer ambiente". Os contêineres utilizam o mesmo *kernel* que a máquina hospedeira e, além disso, podem compartilhar camadas de sistema de arquivos, o que permite que múltiplos contêineres usem as mesmas dependências sem a necessidade de duplicá-las, poupando assim recursos computacionais como memória e armazenamento.

Algumas características fundamentais dos contêineres incluem a fácil e confiável recriação de um mesmo contêiner, o bom isolamento entre as aplicações que estão sendo executadas dentro de contêineres distintos em uma mesma máquina, e a leveza em comparação com máquinas virtuais (GOOGLE, 2024a). O isolamento é uma grande vantagem dos contêineres, garantindo que as aplicações executadas em contêineres diferentes não interfiram uma nas outras, apesar de estarem rodando no mesmo Sistema Operacional e compartilhando o mesmo *kernel*. Isso oferece segurança, evita conflitos de dependências entre os diferentes ambientes de execução e gera uma facilidade em hospedar aplicações completamente distintas em um mesmo ambiente (GOOGLE, 2024a).

Essas características tornam os contêineres ideais para maximizar a utilização do hardware disponível. Graças à sua leveza e flexibilidade, é possível executar múltiplos contêineres em uma única máquina ou distribuir a carga de trabalho entre contêineres alocados em diferentes máquinas, otimizando a eficiência e a escalabilidade dos recursos computacionais.

2.2.1 Orquestração de contêineres

De acordo com o Google Cloud (GOOGLE, 2024b), a orquestração de contêineres refere-se à automação do gerenciamento de recursos e do ciclo de vida dos contêineres. Essa prática desempenha um papel crucial na garantia da eficiência e escalabilidade em ambientes com múltiplos contêineres, seja em sistemas independentes ou operando em clusters.

Neste TCC, o Kubernetes será empregado como a principal ferramenta de orquestração. Sua implementação permitirá a gestão eficiente dos recursos computacionais disponíveis, bem como a distribuição equilibrada da carga de trabalho entre as máquinas do cluster, maximizando a utilização de hardware limitado.

2.3 Kubernetes

O Kubernetes, também conhecido como k8s, é uma plataforma de orquestração de contêineres desenvolvida pela Google, a qual foi projetada para automatizar a implantação, o dimensionamento e a operação de contêineres. Ele permite a criação de clusters de HA, compostos por um ou mais planos de controle e um ou mais nós (CNCF, 2024a).

Os planos de controle são responsáveis por manter o estado desejado do cluster dentro de parâmetros definidos pelo administrador, como a quantidade de cópias de um contêiner de uma aplicação ou a escolha do nó mais adequado para executar uma aplicação, avaliando os recursos disponíveis (CNCF, 2024a).

Os nós são responsáveis por executar um ou mais pods. Os pods são as menores unidades de trabalho no Kubernetes, e cada um pode conter um ou mais contêineres que compartilham recursos como rede e armazenamento (CNCF, 2024c).

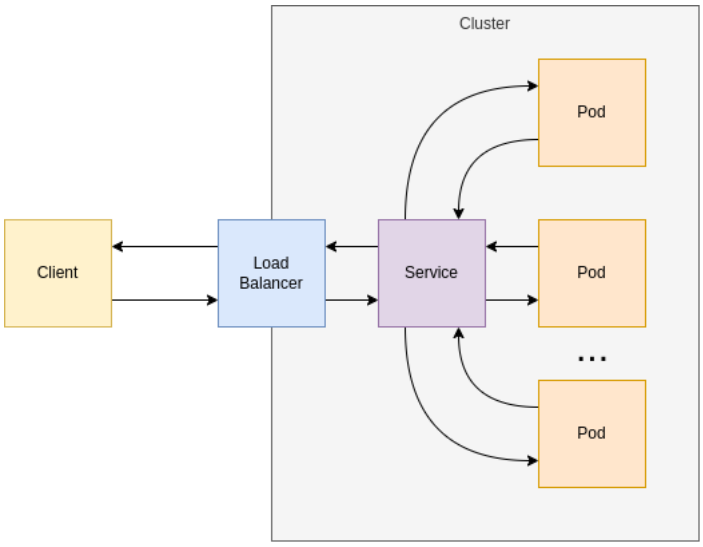
No contexto deste projeto, o Kubernetes desempenhará um papel fundamental na distribuição da carga de trabalho entre as máquinas do cluster utilizando um *LoadBalancer*, garantindo o uso eficiente do hardware disponível e maximizando o aproveitamento dos recursos computacionais.

2.3.1 Distribuição de Carga

Dentro do Kubernetes os pods criam uma rede interna, sem acesso a rede exterior, e para que clientes do servidor possam ter acesso a essa rede é necessário a criação de *Services*. Os *Services* expõem um ou mais pods para a rede externa através de uma ou mais portas do cluster. Existem diversos tipos de *Services*, mas nessa subseção será falado sobre o *Service* de *LoadBalancer*.

O *Service* de *LoadBalancer* no Kubernetes tem a capacidade de distribuir requisições entre um ou mais pods contendo a mesma aplicação, em qual nó do cluster esses pods estão é manejado automaticamente levando em conta os recursos disponíveis no cluster pelo próprio Kubernetes. Esta abordagem garante que o cluster tire o maior proveito possível dos recursos disponíveis.

Figura 1 – Distribuição de carga usando um *LoadBalancer* no Kubernetes



Fonte: Autoria própria (2025).

As requisições são recebidas pelo *LoadBalancer* e depois distribuídas entre os pods que contém a mesma aplicação utilizando o algoritmo *Round Robin*, esse processo juntamente com a administração automática de qual nó tem quais pod no cluster Kubernetes é responsável por distribuir a carga do processamento das requisições entre as máquinas do cluster.

2.3.2 Distribuições do Kubernetes

Assim como outros diversos projetos desendo código aberto, o Kubernetes tem diversas variações que foram criadas com base no projeto. E algumas dessas variações se especializaram na diminuição do consumo de recursos, como memória Memória de acesso aleatório, do inglês *Random Access Memory* (RAM) e processamento da CPU, podendo ser citadas:

- O *k3s*, desenvolvido para ambientes de Internet das Coisas, do inglês *Internet of Things* (IoT), é otimizado para ser implantado em uma ampla gama de arquiteturas de sistemas (Rancher Labs, 2024).
- O *MicroK8s* é uma versão simplificada e otimizada do Kubernetes, projetada para permitir a criação de clusters com facilidade e rapidez (CANONICAL, 2024).
- O *k0s* é uma variante *bare metal* do Kubernetes, ideal para ser utilizado em IoT, com flexibilidade para ser implantado em diversos ambientes (K0S PROJECT, 2024).

Neste TCC as diferentes variantes do Kubernetes serão implantadas no OS debian, pois o mesmo tem compatibilidade com todas as versões escolhidas do Kubernetes e tem um bom desempenho na sua versão mais minimalista (DEBIAN, 2024). Após a implementação do cluster, serão realizados testes para avaliar o consumo individual de recursos de RAM e CPU dos computadores, assim como o tempo de resposta, com o objetivo de comparar a performance de cada variação do Kubernetes em máquinas com recursos computacionais limitados.

2.4 Computadores com poder computacional limitado

O conceito de computadores com capacidade computacional limitada pode variar dependendo do contexto de uso. Para os propósitos deste TCC, define-se como máquinas com recursos modestos em relação aos requisitos típicos de aplicações modernas. Nesta definição, incluem-se computadores que possuem entre 2 e 4 gigabytes de RAM, processadores com 2 a 4 núcleos de CPU com arquitetura x86-64, características que restringem sua capacidade de executar tarefas intensivas em recursos. Os computadores que serão utilizados são descritos de melhor maneira na subseção 4.1.1.

3 TRABALHOS RELACIONADOS

Diversos estudos abordam a utilização de Kubernetes em ambientes com recursos limitados, destacando-se pela adaptação de arquiteturas para permitir a operação eficiente em hardware de baixo custo. Estes trabalhos são relevantes para este TCC, pois apresentam alternativas e abordagens que se alinham com a proposta de implantar um cluster Kubernetes em computadores com poder computacional limitado.

Um estudo realizado por Silva (SILVA, 2022), aborda a implementação de soluções computacionais em ambientes com hardware modesto utilizando tecnologias como o Kubernetes para otimização de recursos em sistemas com capacidade computacional limitada. Esse trabalho é particularmente interessante, pois explora as mesmas questões relacionadas ao uso de clusters Kubernetes em máquinas de baixo custo, oferecendo *insights* sobre os desafios de desempenho e eficiência.

Além disso, o trabalho de *Programming Group* (GROUP, 2023) explora diferentes distribuições de Kubernetes otimizadas para ambientes com recursos limitados, como o k3s, k0s e microk8s, destacando a importância dessas versões leves para a implementação de clusters em hardware modesto. Essa pesquisa complementa a proposta deste TCC, já que o uso de distribuições otimizadas pode ser uma solução crucial para garantir o funcionamento eficiente do cluster, mesmo em máquinas com pouca memória RAM e capacidade de processamento.

Outro estudo relevante é o de *Learn Fast Make Things* (MORSE, 2023), que discute como transformar hardware antigo em clusters Kubernetes, utilizando recursos limitados de maneira eficiente. Esse trabalho se aproxima diretamente da proposta deste projeto, ao sugerir métodos de aproveitamento de hardware obsoleto para a criação de clusters, alinhando-se à proposta de utilizar máquinas com poder computacional limitado para implementação de soluções escaláveis.

Esses trabalhos fornecem uma base teórica e prática sólida, que orienta a escolha de tecnologias e estratégias para a implementação de clusters Kubernetes em hardware com recursos limitados, contribuindo significativamente para o desenvolvimento deste TCC.

4 MATERIAIS E MÉTODOS

Este capítulo apresenta os materiais e métodos utilizados para a implementação e avaliação do *cluster* Kubernetes em máquinas com poder computacional limitado. Serão descritos os componentes de *hardware* e *software*, bem como os testes e métricas planejados para analisar o desempenho do sistema.

4.1 Materiais

Os materiais utilizados estão organizados em duas categorias principais: *hardware*, que abrange os computadores e equipamentos de rede, e *software*, que compreende as distribuições de Kubernetes.

4.1.1 Hardware

Os três computadores utilizados no *cluster* foram reaproveitados, reforçando a proposta de sustentabilidade do projeto. A Tabela 1 lista os principais componentes de cada máquina, enquanto a Tabela 3 apresenta suas especificações técnicas.

Tabela 1 – Peças que compõem os computadores.

Computador	CPU	Placa mãe	RAM	Disco	Placa de Rede
Plano de Controle	i7-870	bpc-hm55	kvr16n11/4	wd5000lpvx	Placa mãe
Nó 1	e4600	ga-g31m-s2l	kvr800d2n6/2gb (2)	st31000524as	Placa mãe
Nó 2	e2220	ipm42	mt16htf25664ay-800j1	hm321hi	37nb 12270 313

Fonte: Elaborado pelo autor.

Tabela 2 – Especificações técnicas dos computadores.

Computador	Núcleos CPU	*Clock* CPU	*Cache* CPU	RAM	Velocidade de Rede
Plano de Controle	4	2.95 GHz	8 MB	4 GB	100 Mbps
Nó 1	2	2.40 GHz	2 MB	2 GB	100 Mbps
Nó 2	2	2.40 GHz	1 MB	2 GB	100 Mbps

Fonte: Elaborado pelo autor.

Além dos computadores, será utilizado um *switch* de rede do modelo enh916p-nwy com *fast ethernet* (até 100mbps de velocidade) para interligar os dispositivos e um roteador para fornecer acesso à Internet.

Cabe salientar que dois dos computadores e o *switch* fazem parte do projeto de extensão **Tecno-lixo: oficina do aprender**, o qual é vinculado ao Curso de Tecnologia em Sistemas para Internet. Ainda, o projeto está sendo desenvolvido na sala em que o projeto funciona.

4.1.2 Software

As distribuições de Kubernetes escolhidas são otimizadas para ambientes de baixa capacidade computacional, conforme descrito a seguir:

- O *k3s*, desenvolvido para ambientes de IoT, é otimizado para ser implantado em uma ampla gama de arquiteturas de sistemas (Rancher Labs, 2024).
- O *MicroK8s* é uma versão simplificada e otimizada do Kubernetes, projetada para permitir a criação de clusters com facilidade e rapidez (CANONICAL, 2024).
- O *k0s* é uma variante *bare metal* do Kubernetes, ideal para ser utilizado em IoT, com flexibilidade para ser implantado em diversos ambientes (K0S PROJECT, 2024).

Para implementação dos testes e monitoramento, serão utilizadas as seguintes ferramentas:

- Node.js v22 + Express: Ambiente de execução JavaScript para desenvolvimento das APIs de teste, escolhido por sua simplicidade de uso na construção de microsserviços.
- Axios: Biblioteca HTTP para requisições entre os microsserviços.
- Apache JMeter: Ferramenta para testes de estresse e carga.
- MySQL 8.0: Banco de dados relacional para o teste CRUD, configurado em contêineres com persistência de dados.
- Python 3.9 + server-system-monitor: Biblioteca para monitoramento contínuo de recursos (CPU, RAM, disco) com coleta de dados via SSH.

4.2 Métodos

Nesta seção são descritos os testes e métricas planejados para avaliar o desempenho do cluster Kubernetes implementado em máquinas com poder computacional limitado.

4.2.1 Testes Individuais das Máquinas

O objetivo desses testes é medir o desempenho individual de cada máquina que compõe o cluster. Serão realizados os seguintes testes:

- Somente o OS sendo executado: Este teste visa diferenciar os recursos do sistema operacional dos recursos consumidos pela distribuição de Kubernetes.

- Somente a distribuição de Kubernetes sem estar conectada ao cluster: Este teste medirá o consumo de recursos computacionais da máquina executando a distribuição de Kubernetes isoladamente.
- Máquina conectada ao cluster: Este teste avaliará o desempenho da máquina quando integrada ao cluster Kubernetes.

4.2.2 Testes de Interação do Cluster

Os testes descritos nesta seção têm como objetivo avaliar o desempenho geral do cluster Kubernetes em diferentes cenários de uso. Serão realizados os seguintes experimentos:

- **Alto volume de requisições**: Uma aplicação de cálculo de um dígito da sequência de *Fibonacci* realizará requisições ao próprio cluster de forma recursiva, simulando um cenário de microsserviços com intensa comunicação entre componentes. O parâmetro variável será o dígito da sequência.
- **Computação intensiva**: Uma aplicação que executa a ordenação de um *array* invertido usando o algoritmo *Bubble Sort* será implantada para medir o desempenho do cluster em cargas de trabalho que demandam alto uso de CPU. Os parâmetros variáveis incluirão o tamanho do *array* e a quantidade de requisições por minuto.
- **Dados persistentes**: Uma aplicação CRUD (Criar, Ler, Atualizar, Excluir) simples será utilizada para analisar a eficiência do cluster no gerenciamento de operações persistentes. O parâmetro variável será a quantidade de dados a serem alterados.

Esses experimentos permitem avaliar o desempenho do cluster em situações diversas, desde alto tráfego de requisições até cargas computacionalmente intensivas e acesso a dados, identificando possíveis gargalos e oportunidades para otimização.

4.2.3 Métricas

Durante os testes, serão coletadas as seguintes métricas para avaliar o desempenho do sistema:

- Consumo de memória RAM: Será monitorada a utilização de memória pelas máquinas individualmente e pelo cluster como um todo, identificando o impacto das diferentes cargas de trabalho.
- Utilização de processamento (CPU): Acompanhamento da carga de processamento em cada máquina e no cluster com o objetivo de detectar possíveis gargalos.

- Tempo de resposta: Medição do tempo necessário para as aplicações implantadas no cluster processarem as requisições enviadas, avaliando a eficiência do sistema.
- Desempenho de leitura e escrita no disco: Análise do tempo gasto em operações de I/O nas máquinas, fundamental para cargas de trabalho que dependem de armazenamento.

5 PROJETO

Este capítulo descreve a metodologia e os procedimentos para avaliar o desempenho de um cluster Kubernetes em máquinas com poder computacional limitado. O objetivo é identificar os limites operacionais do cluster, utilizando um tempo de resposta médio de um segundo como critério de parada e coletando métricas de CPU, RAM e disco. A execução dos testes será realizada em um ambiente controlado, utilizando diversos mecanismos para garantir a consistência dos testes. O capítulo está organizado em escopo dos testes, modelagem das aplicações, execução dos testes e implementação do cluster.

5.1 Escopo dos Testes

Os testes tem o objetivo de determinar os limites e gargalos na operação do cluster. O limite será determinado como um tempo de resposta médio de um segundo e os gargalos serão determinados utilizando medições frequentes de RAM, CPU e taxa de leitura e escrita do disco.

5.2 Modelagem dos Testes

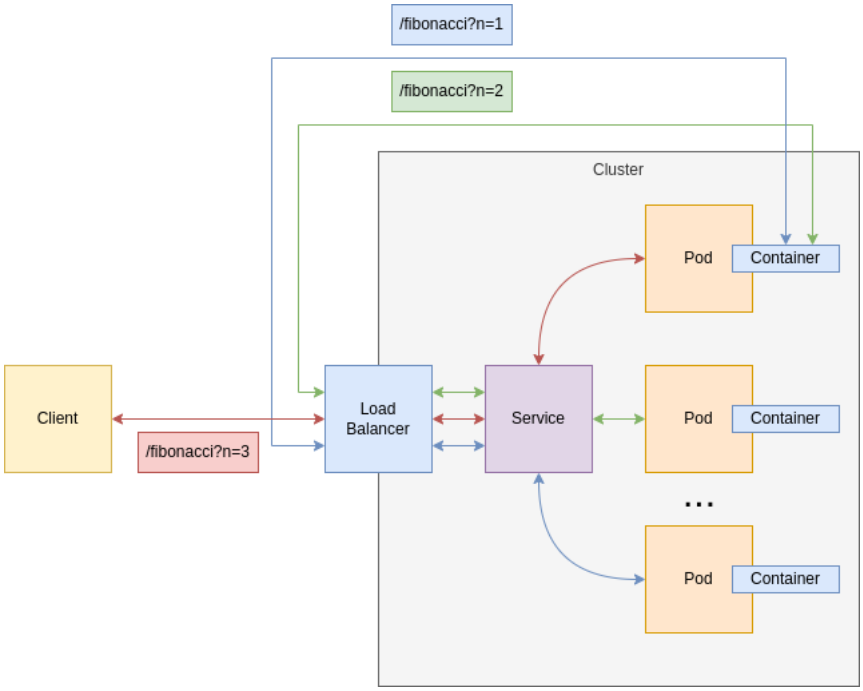
Para avaliar o desempenho do cluster, serão desenvolvidos três módulos de uma aplicação de diferentes naturezas computacionais:

- Implementação do algoritmo do calculo de um número de *Fibonacci*usando requisições para formar uma recursão, assim simulando uma grande quantidade de requisições simultâneas, mas as requisições feitas pelo cliente que estará testando o servidor serão feitas sequencialmente e não simultaneamente. O número calculado irá aumentar linearmente, de um em um. A figura 2 ilustra o inicio das requisições do dígito *n* da sequência de *Fibonacci* por meio do cliente ao *LoadBalancer*, e em seguida o redirecionamento das requisições à um dos *pods*. O *pod* redireciona a requisição para o contêiner contendo o algoritmo, sendo que ele var realizar duas requisições dos dígitos *n* -1 e *n* -2 ao *LoadBalancer* para que sejam somados, e assim até que o digito requisitado seja 1 ou 2 retornando 1 como resposta.

- Desenvolvimento de um algoritmo de ordenação utilizando Bubble Sort para processar listas invertidas de diferentes tamanhos. Esse teste visa simular uma quantidade razoável de requisições em um processo computacional relativamente pesado. Os tamanhos em potências de dois começando em dois elevado a dez e com quantidade de requisições simultâneas em potências de dois começando com dois elevado a quatro.

- Implementação de uma aplicação CRUD (Criar, Ler, Atualizar, Excluir) para testar a eficiência do cluster na manipulação de dados persistentes. Esse experimento avalia

Figura 2 – Ciclo de vida de um requisição do número de Fibonacci.



Fonte: Autoria própria (2025).

o impacto da comunicação entre serviços e o desempenho do armazenamento, além de avaliar o quão bem o Kubernetes vai manejar o container com o armazenamento persistente num ambiente com capacidade computacional limitada. O teste vai consistir em criar, ler, atualizar e excluir massas de dados sequencialmente, medindo o tempo de resposta entre essas operações, o parâmetro alterado será a quantidade de dados.

Em todos os testes os dados de consumo de CPU, RAM e quantidade de dados em escrita e leitura do disco serão coletados a cada um segundo, além do tempo de resposta médio que cada requisição vai levar para ser concluída. Esses dados serão comparados entre as distribuições de Kubernetes.

5.3 Execução dos Testes

Será realizada a coleta de dados como CPU, RAM e leitura/escrita em disco através do uso do servidor SSH que será instalado nas máquinas, esse processo será realizado antes e depois das distribuições Kubernetes serem instaladas.

A duração dos testes será de cinco minutos por parâmetro alterado e entre um teste e outro o cluster sera mantido ligado por cinco minutos sem responder nenhuma requisição para poder se resfriar, assim evitando inconsistências causadas por sobreaquecimento nos testes anteriores.

A troca de uma distribuição do Kubernetes e outra consistira em reinstalar o sistema operacional nos computadores, o servidor SSH e a próxima distribuição do Kubernetes a ser testada, esse processo tem a função de evitar que pacotes instalados e configurações feitas pelas versões testadas anteriormente afetem os resultados, assim garantido consistência entre os testes.

5.4 Implementação do Cluster

A implementação do cluster constituirá de três computadores com e um switch configurações heterogêneas, essas configurações são individualmente exploradas na subseção 4.1.1. Aonde estão citadas desde as características gerais até os modelos das peças.

O cluster irá utilizar uma arquitetura de rede em estrela, aonde cada computador poderá conectar aos outros através de um switch, a rede vai ser configurada para utilizar IPs estáticos. Os IPs estáticos são um requisito para o funcionamento normal da comunicação entre os computadores do cluster.

Devido a dificuldade do acesso a Internet cabeada no lugar aonde o cluster esta sendo ser implementado, a proibição da utilização de roteadores para a realização de uma tradução de endereços de rede , do inglês *Network address translation* (NAT) e dificuldades de acesso a rede que utiliza o Protocolo de Autenticação Extensível Protegido, do inglês *Protected Extensible Authentication Protocol* (PEAP) pelos computadores do cluster, os pacotes necessários para o funcionamento estão sendo instalados através de *pendrives* e utilizando outros computadores para fazer o NAT.

5.5 Resultados Esperados

É esperado que o cluster tenha um desempenho melhor em atividades que utilizem mais de paralelismo como o algoritmo de *Fibonacci* implementado ou bancos de dados distribuídos (não abordados neste trabalho) em relação a atividades sequenciais como o *Bubble Sort*. Porém é possível que em atividades que utilizem de paralelismo o cluster tenha um desempenho similar ou até melhor do que alguns computadores com uma capacidade computacional maior operando individualmente. Assim demonstrando a viabilidade da utilização de *hardware* limitado para alguns tipos de aplicação, além de indicar quais tipos de aplicação e em quais configurações elas se comportariam melhor dentro do cluster.

Outro ponto importante a se considerar é que clusteres Kubernetes são melhores respondendo a um grande número de requisições que utilizam um baixo poder computacional do que respondendo uma pequena quantidade de requisições que utilizam um grande poder computacional. Pois o algoritmo de distribuição de carga utilizado pelo Kubernetes distribui as requisições, mas não o processamento que é realizado por elas, com exceção de algoritmos

que utilizem requisições a outros serviços do cluster para distribuir a carga (arquitetura de microsserviços).

6 CRONOGRAMA

O cronograma a seguir apresenta as atividades e entregas planejadas para a realização do TCC, distribuídas ao longo do período estipulado.

Tabela 3 – Cronograma de atividades.

Atividade	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul
Revisão dos apontamentos da banca	X	X							
Revisão bibliográfica	X	X	X	X	X	X	X	X	X
Redação do projeto de TCC			X	X					
Defesa do projeto de TCC				X					
Escrita da Monografia de TCC					X	X	X	X	
Implantação e teste do cluster					X	X	X	X	
Elaboração da apresentação final								X	X
Defesa final do TCC									X

Fonte: Elaborado pelo autor.

REFERÊNCIAS

CANONICAL. **MicroK8s Documentation**. 2024. Disponível em: https://microk8s.io/docs. Acesso em: 18 nov. 2024.

CNCF. **Cluster Architecture**. Online, 2024. Disponível em: https://kubernetes.io/docs/concepts/architecture/. Acesso em: 18 nov. 2024.

CNCF . **Overview**. Online, 2024. Disponível em: https://kubernetes.io/docs/concepts/overview/. Acesso em: 18 nov. 2024.

CNCF . **Pods**. Online, 2024. Disponível em: https://kubernetes.io/docs/concepts/workloads/pods/. Acesso em: 20 nov. 2024.

DEBIAN. **Debian – Reasons to use Debian**. 2024. Accessed: 20 Nov. 2024. Disponível em: https://www.debian.org/intro/why_debian.

GOOGLE . **What Are Containers?** 2024. Disponível em: https://cloud.google.com/learn/what-are-containers. Acesso em: 6 nov. 2024.

GOOGLE . **What is container orchestration?** 2024. Disponível em: https://cloud.google.com/discover/what-is-container-orchestration?hl=en. Acesso em: 18 nov. 2024.

GROUP, P. **Lightweight Kubernetes Distributions**. 2023. Disponível em: https://programming-group.com/assets/pdf/papers/2023_Lightweight-Kubernetes-Distributions.pdf. Acesso em: 20 nov. 2024.

IBM. **What is Cluster Computing?** 2024. Disponível em: https://www.ibm.com/think/topics/cluster-computing. Acesso em: 18 nov. 2024.

K0S PROJECT. **K0s Documentation**. 2024. Disponível em: https://k0sproject.io/. Acesso em: 20 nov. 2024.

MONTEIRO, E. R. *et al.* **DevOps**. SAGAH, 2021. 154 p. ISBN 9786556901725. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9786556901725/. Acesso em: 6 nov. 2024.

MORSE, G. **How to turn your old hardware into a Kubernetes cluster**. 2023. Disponível em: https://learnfastmakethings.com/p/how-to-turn-your-old-hardware-into-a-kubernetes-cluster-129d17aa8704. Acesso em: 20 nov. 2024.

Rancher Labs . **K3s Documentation**. 2024. Disponível em: https://k3s.io/. Acesso em: 20 nov. 2024.

SILVA, J. **Implementando um sistema de conteinerização com Kubernetes usando GitOps**. 2022. Trabalho de Conclusão de Curso (Curso de Ciência da Computação) – Universidade Federal de Pernambuco, Recife. Disponível em: https://www.cin.ufpe.br/~tg/2022-1/tg_CC/tg_pgrr.pdf. Acesso em: 20 nov. 2024.