Virtualização de Cluster em *Oracle Virtual Machine* para validação de Cluster físico

Cristian Rigatti de Campos
Universidade Franciscana UFN
Santa Maria, RS
cristian.campos@ufn.edu.br

Sylvio Andre Garcia Vieira
Universidade Franciscana UFN
Santa Maria, RS
sylvio@ufn.edu.br

Resumo - Grandes quantidades de dados requerem grande capacidade computacional para serem processados, assim, a reunião de vários computadores promove eficácia na redução do tempo de resposta para esta situação. Neste trabalho, realizou-se a construção de um cluster de modo virtualizado, sua análise de funcionamento e testes de desempenho, usando o cálculo de π como estudo de caso para demanda de tempo. Dessa forma, otimizou-se o tempo de processamento, permitindo que se possa traçar um comparativo entre ganho de tempo de resposta e adição de novos computadores ao cluster, avaliando sua viabilidade. Assim, ao serem adicionados novos nós ao cluster, o tempo de processamento diminuiu em cerca de 45% para cada novo nó.

***Index Terms*—Beowulf, Apache Hadoop, Número π**

I. Introdução

O desempenho de computadores atinge limites de processamento determinados pelo hardware, onde os sistemas estão embarcados. Isto frequentemente faz com que grandes volumes de dados como os de satélites que tem seu objetivo em gerar imagens tanto da atmosfera terrestre como do espaço observável, demandem muito tempo para obter resultados ao serem processados.

O agrupamento de computadores com o objetivo de somar suas capacidades vem sendo utilizado em muitas atividades em que se necessite de uma capacidade superior a uma única máquina. Isso permite superar a limitação de hardware de uma única máquina, se ocorrer alguma falha em um dos computadores que formam o cluster este nó pode ser isolado e tratado, isso vai ter um impacto no seu desempenho, mas permite que o processamento de dados não seja interrompido. O cluster utilizado pela NASA no estudo da atmosfera terrestre é o de *Earth Observing System Data and Information System* (EOSDIS). Ele é composto por mais de 30.000 nós de processamento. Os nós são compostos por uma variedade de CPUs e GPUs incluindo Intel Xeon, AMD EPYC e NVIDIA V100, e são conectados por uma rede de alta velocidade baseada no protocolo InfiniBand, o cluster é capaz de processar até 10 petabytes de dados por dia[1].

A utilização de diversos computadores com sistemas operacionais Linux tem sido empregada em alguns ambientes de hospedagem de site e serviços de armazenamento em nuvem, devido a ser um sistema confiável e estável, que pode ser executado em hardware de baixo custo, seguro por ser um software livre é constantemente atualizado e pode ser modificado de acordo com a necessidade. Neste trabalho, o Linux Ubuntu 22.04 foi escolhido por possuir suporte para clusterização de computadores, ser uma versão estável e segura do Linux, além de permitir uma instalação sem o modo gráfico, o que otimiza processamento.

Para o software foi utilizado o Apache Hadoop versão 3.3.4 por ser uma versão estável, trazer correção de bugs, pode ser escalonável para cluster maiores com milhares de nós. Este trabalho tem como objetivo a criação de um cluster virtual utilizando Apache Hadoop, realizar testar o desempenho do cluster conforme se aumenta o numero de nós e utilizar o sistema operacioanal no Linux Ubuntu 22.04.

A. Objetivos específicos

- Instalar e configurar o Linux(Ubuntu);
- Realizar a interconexão destes computadores utilizando o software Apache Hadoop de forma a partilhar processamentos;
- Descrever todo o passo a passo da instalação e configuração do cluster;
- Realizar testes de múltiplo processamento para calcular o valor de PI;
- Analisar e comparar a eficiencia do cluster conforme se aumenta o numero de nós.

Este trabalho pode ser replicado para um cluster físico, por utilizar o sistema operacional Linux traz a possibilidade da criação de um cluster Beowulf utilizando computadores mais velhos ou de menor capacidade computacional, permitindo que instituições de ensino ou empresas ao atualizarem seus laboratórios possam utilizar estes computadores de menor capacidade para criar um ambiente de testes e de treinamento.

II. Referencial Teórico

Aqui serão apresentados os concceitos necessários a compreensão do que foi desenvolvido neste tranbalho.

A. Cluster computacional

É chamado de cluster ou agrupamento de computadores quando dois ou mais computadores trabalham como se fosse

uma única máquina somando capacidade de processamento, recursos e armazenamento, e devem funcionar de uma forma em que o usuário tenha a impressão de estar usando apenas um computador [2]. Cada computador interligado ao cluster recebe o nome de "nó", e pode trabalhar de forma diferente dos demais.

B. Tipos de cluster

Conforme o autor [3], os cluster são classificados em:

- *Failover* ou *High Availability Computing Cluster* (Cluster de Alta Disponibilidade) Garante que uma rede permaneça sempre ativa. Para isso, caso um computador apresente falha e fique fora do ar, outro continua mantendo a rede operante.
- *Load Balancing* (Cluster para Balanceamento de Carga) Trata-se de uma estrutura na qual todos os computadores são responsáveis pela execução de uma determinada tarefa. Assim, caso um dos equipamentos apresente algum problema, ele é automaticamente retirado do sistema e a função inicial atribuída a ele é dividida entre os demais nós.
- *High Performance Computing Cluster* (Cluster de Alto Desempenho) Esse tipo de cluster é utilizado para desempenhar tarefas de alto desempenho, de modo que garantam a máxima performance da atuação. Utilizando o processamento em paralelo ele é capaz de transformar uma tarefa complexa em várias simples e as distribui entre os nós integrados ao sistema

  É possível haver um cluster virtual que segue o mesmo princípio de um cluster físico, ou seja, de integrar computadores em um mesmo sistema, porém por meio de uma rede online. Esse processo acaba sendo mais dinâmico e garante que mesmo máquinas que estejam distantes fisicamente possam ser conectadas para aumentar o seu desempenho.

  Neste projeto foi feito um cluster de alto desempenho, que consiste em uma integração virtual a entre quatro computadores.

C. Beowulf

O nome Beowulf é inspirado no herói da literatura inglesa Beowulf que era descrito como tendo a força de muitos homens. O nome condiz com o cluster por este tem a força de outros computadores juntos formando um sistema com maior eficiencia, velocidade e capacidade computacional [4]. Criado por Thomas Sterling e Don Becker em 1994 na NASA-Goddard (CESDIS, o Centro de Excelência em Dados Espaciais e Ciências da Informação) como alternativa para o elevado custo dos fornecedores e a falta de suporte, criando assim o primeiro supercomputador utilizando (*Mass Market Commodity-Of-The-Shelf*) (M²COTS) ou seja, em uma tradução literal, componentes disponíveis ao mercado consumidor comum tirados da prateleira [4]. Utilizando equipamentos de uso doméstico sem grande capacidade computacional, foi criado o primeiro Beowulf.

Beowulf é um cluster de desempenho escalável, baseado numa infra-estrutura de hardware comum, rede privada dedicada ao cluster e um Sistema Operacional de código aberto, um exemplo é o Linux (Ubuntu).

D. Apache Hadoop

É uma plataforma de software de código aberto para o armazenamento e processamento distribuído de grandes conjuntos de dados, utilizando clusters de computadores com *hardware commodity*, que são hardware comuns, de baixo custo e produzidos em grande quantidade, como memorias RAM, discos rigidos e processadores. Os serviços do Hadoop fornecem armazenamento, processamento, acesso, governança, segurança e operações de dados [2].

Foi utilizado o Hadoop devido a sua capacidade de armazenamento, gerenciar e analisar grandes quantidades de dados estruturados e não estruturados de forma rápida, confiável, flexível e de baixo custo. Confiável pois quando um nó de processamento falha, é redirecionado para os nós restantes no cluster e os dados são automaticamente replicados em preparação para falhas de nó futuras. É de baixo custo pois é um *open source*. É flexível pois não é necessário ter um esquema de dados estruturados criados anteriormente o armazenamento dos dados.

E. Linux

É um sistema operacional de software livre criada em 1991 por Linus Torvalds, sendo criado inicialmente como um hobby com a finalidade de criar um Sistema Operacional, sendo apresentada a um grupo de discussão da Usenet e convidando outras pessoas a participarem do projeto. Com a ajuda da comunidade em 1992, Linus Torvalds licenciou o kernel *Linux* sob os termos da GNU General Public License (GPL). Isso tornou o *Linux* um software de código aberto, garantindo que qualquer um pudesse ver, modificar e distribuir o código-fonte. Ao longo dos anos foram criadas diversas distribuições do *Linux*, exemplos Ubuntu, Fedora, Manjaro. Estas distribuições se diferenciam em seu foco principal podendo ser de uso geral como o Ubuntu, para servidores como o CentOS, segurança como o Kali Linux, entre outros objetivos. O *Linux* traz a flexibilidade de escolha para o usuario, permitindo que ele trabalhe com um ambiente mais otimizado, uma maior segunrança pois com um codigo aberto qualquer pessoa pode analisar este codigo e trazer soluções para problemas e opções de melhorias para o sistema operacional.

Neste projeto está sendo utilizado o Ubuntu 22.04 por ser uma versão atualizada e estável do *Linux* [5], que possui suporte para o Apache Hadoop o que nos permite executar o cluster em uma aplicação web.

F. VirtualBox VM Oracle

É um *software* de virtualização de máquina virtual (VM) gratuito e de código aberto desenvolvida pela empresa alemã Innotek GmbH e lançada em 2007. Posteriormente, a Oracle Corporation adquiriu a Innotek em 2008, tornando a *VirtualBox* parte de sua gama de produtos.

Este *software* permite que uma máquina física seja dividida em várias máquinas virtuais oque permite seja executado um ou varios outros sistemas operacionais em um unico computador. Através dessa virtualização, é possível criar ambientes isolados e independentes, proporcionando a flexibilidade de testar diferentes sistemas operacionais e configurações em um único dispositivo.
Com uma interface gráfica intuitiva, a VirtualBox simplifica a criação de uma máquina virtual. Ela permite a instalação do sistema operacional desejado e a configuração dos recursos computacionais que serão alocados para a máquina virtual. Essa flexibilidade possibilita defeinir a quantidade de memória, processadores e armazenamento de acordo com as necessidades específicas de cada máquina virtual.
As máquinas virtuais não se limitam à simples simulação de outros sistemas operacionais. Cada máquina virtual funciona como um sistema independente, atribuindo-se um endereço IP próprio e permitindo o acesso remoto. Isso reduz significativamente os custos de uma empresa com hardware, já que várias máquinas virtuais podem operar em um único equipamento, oferecendo a capacidade de acesso e controle remoto, além de otimizar recursos e reduzir despesas operacionais.
Neste trabalho foram criado quatro maquinas virtuais, com o sistema operacional Ubuntu 22.04, em que uma dela é o gerente do cluster e os demais são os agentes, estas maquinas virtuais foram criadas no computador pessoal do autor deste trabalho.

G. O que é o PI

A descoberta da constante de PI data deste a época dos Egipcios e dos Babilônios no século XVIII a.C, esta constante representa o calculo da razão entre a circunferência de um círculo e o seu diâmetro. O valor encontrado por estes povos foi aproximado entre 3,12 e 3,16. [6]
Arquimedes matemático grego que viveu no século III a.C, desenvolveu um novo método baseado em valores geométricos para poder calcular o valor de PI com maior precisão, que consiste em dividir uma circunferência em polígonos regulares de lados cada vez maiores, e calcular o perímetro de cada polígono. Ao aumentar o número de lados, foi possível aproximar o valor de PI com mais precisão para o valor entre 3,14084507 e 3,14285714 [7].
A dificuldade no cálculo do valor de PI se deve a ser um número irracional ou seja um número que não pode ser expressado como uma fração de dois números inteiros e por ser um número não periódico que é um número onde seus algoritmos decimanis não se repetem em uma sequência fixa. Atualmente com o auxílio de super computadores como o da Universidade de Tóquio foi possivel calcular o valor de PI o número de 1,5 trilhões de dígitos, as aplicações de Pi não tem necessidade desta precisão normalmente se utiliza 15 casas decimais 3.141592653589793.

H. Método de Monte Carlo

O método de Monte Carlo é uma técnica de simulação numérica que utiliza números aleatórios e estáticos para estimar valores de funções matemáticas ou resolver equações diferenciais. Ele recebe esse nome em homenagem ao famoso cassino Monte Carlo, associado à aleatoriedade [8].
O método funciona ao utilizar a aleatoriedade para resolver problemas determinísticos ou calcular resultados complexos. Ele opera em torno da premissa de que, se você gerar um número suficiente de amostras aleatórias dentro de um intervalo específico, pode-se estimar um resultado aproximado para uma determinada questão.
Podemos utilizar o método de Monte Carlo para calcular uma aproximação do valor de PI, este cálculo é realizado ao considerarmos um círculo de raio =1 dentro de um quadrado de lado =2, em seguida são gerados pontos aleatórios dentro do quadrado, os pontos que ficam dentro do círculo são contados e o número de pontos total é guardado[3].
Ao utilizar a fórmula:

$$\frac{\pi}{4} = \frac{\text{Pontos no círculo}}{\text{Pontos totais}}$$

Quanto mais pontos forem mapeados melhor será a aproximação do valor de PI, neste momento que temos a vantagem do cluster de processamento em paralelo, que nós permitimos dividir em o cálculo da equação entre os nós do cluster para assim termos um resultado mais rápido e de maior precisão.
O código utilizado para aplicar o método de Monte Carlos no cluster foi o seguinte:

```
package hadoop.mapreduce.examples;

import java.io.IOException;
import java.util.Random;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;

public class Pi {

    public static class PiMapper extends
        Mapper<Object, Text, Text,
        DoubleWritable> {

        private Random random = new Random();

        @Override
        public void map(Object key, Text value
            , Context context) throws
            IOException, InterruptedException
            {
            // Gerar um n mero aleat rio no
                intervalo [0, 1]
            double x = random.nextDouble();

            // Calcular a dist ncia do ponto
                (x, 0)    circunfer ncia
                unit ria
            double y = Math.sqrt(1 - x * x);
```

```
            // Se o ponto estiver dentro da
                circunfer ncia, adicionar 1
                ao contador
            if (y <= 1) {
                context.write(new Text("in"),
                    new DoubleWritable(1));
            } else {
                context.write(new Text("out"),
                     new DoubleWritable(1));
            }
        }
    }

    public static class PiReducer extends
        Reducer<Text, DoubleWritable, Text,
        DoubleWritable> {

        private double countIn = 0.0;
        private double countOut = 0.0;

        @Override
        public void reduce(Text key, Iterable<
            DoubleWritable> values, Context
            context) throws IOException,
            InterruptedException {
            for (DoubleWritable value : values
                ) {
                if (key.equals("in")) {
                    countIn += value.get();
                } else {
                    countOut += value.get();
                }
            }
        }

        @Override
        public void cleanup(Context context)
            throws IOException,
            InterruptedException {
            // Calcular o valor de PI
            double pi = 4.0 * countIn / (
                countIn + countOut);

            // Escrever o resultado
            context.write(new Text("pi"), new
                DoubleWritable(pi));
        }
    }

    public static void main(String[] args)
        throws Exception {
        // Executar o programa MapReduce
        org.apache.hadoop.util.ToolRunner.run(
            new Pi(), args);
    }
}
```

Listing 1. Código calculo PI

Explicação do Codigo:

- PiMapper: A classe "PiMapper"é responsável por gerar números aleatórios e calcular a distância dos pontos aleatórios à circunferência unitária.
- PiReducer: A classe "PiReducer"é responsável por calcular o valor de PI a partir dos resultados dos mappers.
- Cada mapper gera um número aleatório no intervalo [0, 1].
- O mapper calcula a distância do ponto (x, 0) à circunferência unitária, onde x é o número aleatório gerado.
- Se o ponto estiver dentro da circunferência, o mapper adiciona 1 ao contador "countIn".
- Se o ponto estiver fora da circunferência, o mapper adiciona 1 ao contador "countOut".
- O reducer soma os valores dos contadores "countIn"e "countOut".
- O reducer calcula o valor de PI a partir dos resultados dos mappers.

Explicação de como código envia os dados para os nós do cluster e depois retorna ao mestre está localizada nas funções map() e reduce():
Durante a fase de mapeamento map(), cada nó mapeador recebe um pedaço dos dados de entrada e gera pares de chave-valor intermediários:

- Entrada de dados: Os dados de entrada são divididos em pedaços menores e enviados para nós mapeadores individuais.
- Geração de números aleatórios: Cada nó mapeador gera um número aleatório no intervalo [0, 1].
- Cálculo da distância: O nó mapeador calcula a distância do ponto (x, 0) para a circunferência unitária, onde x é o número gerado aleatoriamente.
- Atualização do contador: Se o ponto cair dentro do círculo, o nó mapeador incrementa o contador in. Caso contrário, incrementa o contador out.
- Pares de chave-valor intermediários: O nó mapeador emite pares de chave-valor intermediários, onde a chave é in ou out e o valor é 1.

Durante a fase de redução reduce(), cada nó redutor recebe um conjunto de pares de chave-valor intermediários com a mesma chave. O nó redutor acumula os valores associados a cada chave e calcula o resultado final.

- Agregação de dados: Os pares de chave-valor intermediários são embaralhados e agrupados por chave para os respectivos nós redutores.
- Acumulação de contadores: Cada nó redutor acumula os valores associados a cada chave (in e out).
- Cálculo de PI: No método cleanup() do redutor, o valor de PI é calculado usando a fórmula: PI = 4.0 * countIn / (countIn + countOut).
- Saída do resultado: O nó redutor emite um par de chave-valor final, onde a chave é pi e o valor é o valor de PI calculado.

Após a execução do codigo o valor de PI encontrado quando são utilizados 1.000.000.000 bilhão de pontos foi de 3.1415926644.

III. Trabalhos Correlatos

Aqui serão apresentados outros trabalhos relevantes ao que está sendo desenvolvido.

A. Instalação e Configuração de um Cluster Beowulf.

No trabalho de Vítor Bruno Pacheco Pereira [3], foram Utilizados 12 computadores obsoletos colocados em paralelo para formar um cluster Beowulf para assim conseguir um poder considerável de processamento afim de ser utilizado pelo Instituto Politécnico da Guarda em aplicações que necessitam de alto desempenho.
Para a realização deste projeto foi criado um computador gerente chamado C1 e os demais computadores como agentes. Os computadores foram conectados por meio de um Switch Enterasys 24 portas interligando os 12 nós do cluster na mesma rede privada. Cada computador possuía o Sistema Operacional Linux - Ubuntu 12.04 LTS de 32bits.
Foi utilizada a biblioteca OpenMPI para transformar um conjunto de máquinas num cluster. Como mecanismo de segurança foi usado o OpenSSH para se fazer uma comunicação criptografada entre as máquinas do cluster, para acessar os computadores por *ssh* foi utilizada a ferramenta Putty, a partir do computador pessoal do autor para visualizar/editar/gerir as configurações.

Após a realização dos testes obtiveram-se os seguintes resultados: considerando a capacidade de processamento do cluster, foi utilizado um programa de computação paralela que implementa o método de Monte Carlo para calcular o valor de Pi. Quando é duplicado o número de nós utilizados no cálculo, o tempo necessário é reduzido sensivelmente para a metade do tempo anterior. Embora tenham evoluído até os 12 nós, o tempo foi reduzindo somente até o quarto nó, quando o tempo de processamento estabilizou.

O artigo em questão é importante e contribui para este trabalho, pois nele se detalha a construção e configuração tanto do hardware como dos softwares utilizados no Cluster, obstáculos encontrados na construção e de que forma o autor os superou.

B. Criação de um cluster Beowulf, desenvolvimento de uma aplicação utilizando processamento paralelo e análise de seu desempenho

No trabalho de Luis André Bazzi Morales [9] foi realizada a construção um cluster Beowulf além do desenvolvimento de uma aplicação que utiliza processamento paralelo, também tratou de mensurar e analisar o desempenho do sistema a nível de software. O cluster construído é composto de 8 computadores com processadores Pentium III 500Mhz interconectados por um switch de 100Mbps. Para a realização do processamento paralelo foi utilizada o MPICH2 que é uma implementação da interface de passagem de mensagens (*Message Passing Interface* – MPI). A aplicação desenvolvida foi escrita utilizando a linguagem de programação C padrão ANSI. Essa tem como proposta resolver uma série de fatoriais dividindo sua carga de trabalho entre os nós de processamento. Para a instalação dos nós foi realizado o processo de clonagem do sistema operacinal da CPU principal para gerar uma imagem que foi gravada no nó com menor capacidade de disco rigido para assim se ter certeza que os demais nós teriam capacidade para processar a imagem.

Um único computador demandou 4.144 segundos para terminar a execução do programa, enquando o cluster desenvolvido com 8 nós foi capaz de reduzir o tempo de execução para 0.52 segundos, o Cluster Beowulf foi capaz de reduzir o tempo de execução em aproximadamente 8 vezes.

A contribuição deste trabalho será da construção fisica do cluster e da implementação dos softwares utilizados para a clonagem dos nós, quais testes foram realizados e dos resultados obtidos com a adição de cada nó a capacidade de processamento do Cluster.

C. Implementação e Análise de Desempenho do Framework Apache Hadoop e da Biblioteca OpenMPI para Multiplicação de Matrizes com um Cluster de Baixo Custo na Plataforma Raspberry PI

No trabalho Elias Rabelo Matos [2], foi realizada a análise do desempenho de algoritmos de multiplicação de matrizes no Raspberry PI, utilizando o *framework* Apache Hadoop e a biblioteca OpenMPI. O ambiente de teste possui quatro dispositivos Raspberry PI 2 Model B, utilizando apenas um núcleo de cada Raspberry exceto no caso do gerente que se utilizou 2 núcleos,foi utilizado em cada placa um cartão de memória SanDisk Ultra class 10 com uma velocidade de leitura e escrita de 30MB/s, estes computadores foram conectados em rede local gigabit usando o switch D-Link DGS-1008A.
Os resultados obtidos nos testes realizados, apontam para um baixo desempenho do Apache Hadoop em comparação com o OpenMPI, indicando que a escolha do *framework* é um fator determinante para o desempenho de algoritmos em ambientes de alto desempenho. Assim como a linguagem de progamação escolhida, a linguagem Java, comumente utilizada no Apache Hadoop, é dezenas de vezes mais lenta do que a linguagem C do OpenMPI na execução de operações numéricas intensivas. A implementação do Apache Hadoop e os teste realizados são importantes referências para o presente trabalho, pois é mostrado como esta implementação deve ser realizada e como o Apache Hadoop interage com o Cluster.

D. Alternatives for Running Linux Applications in Windows

Este Trabalho de Conclução de Curso (TCC) de Valeriu Manuel Ionescu [10], apresenta e compara quatro formas diferentes de se executar aplicações no sistema operacional Linux e no sistema operacional Windows utilizando o mesmo computador, sendo elas: VirtualBox, Docker, Windows Linux Subsystem e Cygwin.
Para realizar os testes de desempenho e velocidade foi utilizado uma CPU Intel Core i5-6200U a 2,30 GHz 2,40 GHz, 8 GB de RAM, Windows 10 Pro, versão: 10.0.18362.116.
Os algoritmos criados utilizaram a linguagem C e foram compilados no gcc, o primeiro algoritmo foi desenvolvido com o objetivo de encontrar os primeiros 500.000 números primos, e o segundo algoritmo utilizou um arquivo especial FIFO para comunicação entre processo, este arquivo é uma referências

para os processos de leitura e gravação de dados, o último algoritmo cria uma “bomba fork” que é um ataque de negação de serviços em que um processo se replica continuamente para esgotar os recursos disponíveis do sistema, desacelerando ou travando o sistema devido à falta de recursos.
Os testes realizados mostraram que o consumo de memória RAM é menor ao utilizar o Cygwin, seguido por Docker, Virtual Box e WSL(Ubuntu). O Docker se mostrou o mais rápido em velocidade de execução seguido pelo WSL, Docker e Cygwin.
A contruibuição deste trabalho é a utilização do VirtualBox como forma de instalar o sistema operacional Linux em uma máquina Windows.

IV. Metodologia

No presente trabalho foi configurado um cluster Beowulf, utilizando quatro computadores sendo um deles o nó *master* e os demais *slaves*, cada um deles com 25 Gb de espaço de armazenamento, 2 GB de memória RAM e 2 núcleos de processamentos, as máquinas virtuais foram criadas no computador pessoal do autor.
Para virtualizar as máquinas foi utilizado o VirtualBox´7.0 VM Oracle, que permite a criação de máquinas independentes da máquina que foi dividida, com um sistema operacional próprio assim como um IP único.
O sistema operacional escolhido para este trabalho foi o Linux Ubuntu 22.04, por ser uma partição estável e confiável, requerer baixa capacidade computacional podendo ser instalada na maioria dos computadores, por ser um software livre traz redução no custo da criação de cluster, além de possuir suporte para a criação de cluster e compartilhamento de dados.
Para a conexão e compartilhamento da capacidade computacional de cada um dos nós, foi utilizado o software Apache Hadoop 3.3.4. Por ele permitir alta escalabilidade no cluster, é possível projetar o cluster para que caso aconteça falha em um dós nós, o Hadoop irá redistribuir automaticamente as tarefas para os demais nós do cluster e por ser um software livre permite uma economia na construção do cluster.
Para testar a eficiência do cluster foi realizado o cálculo do PI por aproximação, utilizando o método de Monte Carlo.

V. Configuração do Cluster

Nesta seção será descrito como o cluster virtual foi configurado, capacidade computacional de cada nó, instalação do Java JDK 08 e instalação do Apache Hadoop 3.3.4, para replicar esta configuração em um cluster físico se deve adicionar um *Switch* que tenha saídas de rede suficientes para todos os nós do cluster.

A. Arquitetura do Hardware do Cluster

Como demonstrado na Figura 1, esta é a configuração de cada um dos nós do cluster.

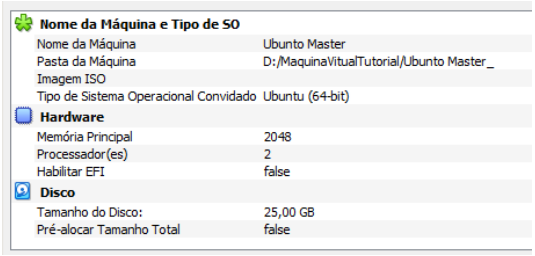

Figura 1: Hardware

Hardware utilizado

- 04 Máquinas vituais sendo UbuntuMaster, Slave1Ubunto até Slave3Ubuntu (2GB RAM, 25GB HDD, 2 núcleos de processamento

Sistema Operacional

- Linux - Ubuntu 22.04.02 LTS de 32bits em todas as máquinas virtuais

Software utilizados

- Criação das máquinas virtuais - *VM Oracle VirtualBox* 7.0
- Conexão das máquinas virtuais - SSH (*Secure Shell*)
- Execução de comandos no cluster - PDSH (*Parallel Distributed Shell*)
- Armazenamento e processamento distribuido no cluster - Apache Hadoop 3.3.4

VI. Resultados

Após realizar a construção e configuração do cluster (ver anexo), como mostrado na Figura 2 e Figura 3 os nós estão ativos.

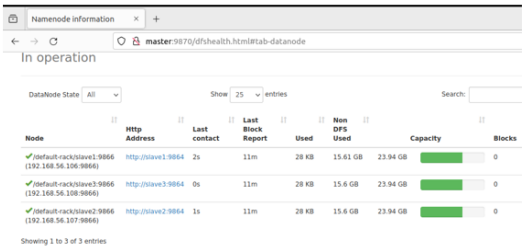

Figura 2: Nós no Apache Hadoop

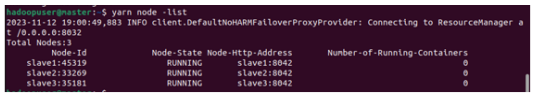

Figura 3: Nós no Linux

A medida que os nós foram duplicados obteve-se uma melhora na precisão e no tempo de processamento próxima aos 45% como observado na Figura 7, foi utilizado o algoritmo mostrado no lstlisting 1, com uma entrada de 1.000.000.000 (um bilhões) de pontos e os resultados foram:

Master + Salve1

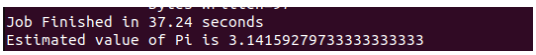

Figura 4: Resultado

Master + Salve1 + Slave2

```
Job Finished in 23.477 seconds
Estimated value of Pi is 3.14159253400000000
```

Figura 5: Resultado

Master + Salve1 + Slave2 + Slave3

```
Job Finished in 12.412 seconds
Estimated value of Pi is 3.14159272000000000
```

Figura 6: Resultado

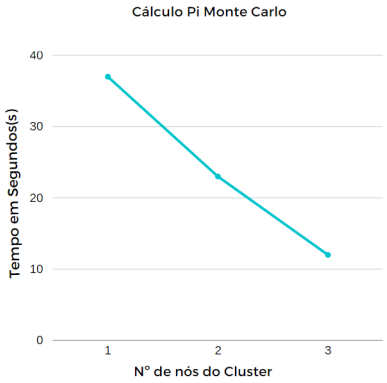

Figura 7: Gráfico de ánalise de resultados com uma entrada de 1.000.000.000 (um bilhão)

Quando foi realizado o teste com uma entrada de com uma entrada de 2.000.000.000 (dois bilhões), a medida que os nós foram duplicados como observado naa Figura 8, Figura 9 e Figura10, obteve-se uma melhora na precisão e no tempo de processamento próxima aos 42% como observado na Figura 11.

Master + Salve1

```
Job Finished in 72.348 seconds
Estimated value of Pi is 3.14159283000000000
```

Figura 8: Resultado

Master + Salve1 + Slave2

```
Job Finished in 47.679 seconds
Estimated value of Pi is 3.14159264400000000
```

Figura 9: Resultado

Master + Salve1 + Slave2 + Slave3

```
Job Finished in 23.459 seconds
Estimated value of Pi is 3.14159254800000000
```

Figura 10: Resultado

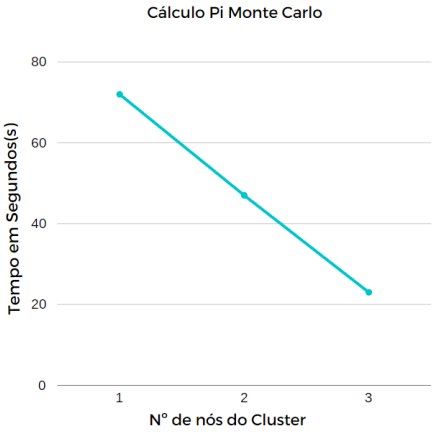

Figura 11: Gráfico de ánalise de resultados com uma entrada de 2.000.000.000 (dois bilhões)

VII. CONSIDERAÇÕES FINAIS

O anexo deste projeto demonstra com sucesso a configuração do cluster. Com a utilização do sistema operacional Linux, obtive um conhecimento mais aprofundado sobre o sistema e o software Apache Hadoop possibilitou a interconexão eficiente entre os nós do cluster para compartilhamento de processamento.

Os resultados alcançados foram satisfatórios, especialmente no contexto do método de Monte Carlo, que demanda um grande volume de pontos para cálcular com precisão o valor de Pi. A execução do algoritmo no cluster proporcionou resultados desejados em um intervalo de tempo reduzido como apresentado na Figura 4, Figura 5 e na Figura 6, evidenciando de forma eficaz a capacidade do cluster e cumprindo o objetivo central deste projeto.

Este trabalho possui aplicações contribuições técnicas para a construção de um cluster e configuração do Apache Hadoop. Tal montagem pode ser replicada tanto em ambientes virtuais quanto físicos, viabilizando testes virtuais antes da implementação física. Essa abordagem oferece a oportunidade de virtualização antes da implementação fisica do cluster, o que permite que seja avaliada a viabilidade do cluster antes da sua execução em um ambiente físico.

O potencial do cluster se estende além dos cálculos matemáticos. Em empresas de tráfego de dados ou marketing em redes sociais, ele pode eliminar a necessidade de um computador mais potente, utilizando os recursos dos computadores existentes para o processamento de dados quando não estiverem em uso. Por exemplo, pode ser usado para analisar grandes conjuntos de dados para identificar padrões ou tendências. Isso resulta em economia para a empresa, otimizando tempo e recursos de maneira significativa.

No mercado atual, é possível encontrar computadores com a configuração dos nós do cluster (2GB RAM, 320 GB HDD, 2 núcleos de processamento) por valores significativamente baixos. Em contrapartida, uma configuração semelhante, mas unificada, com 8GB de RAM, 500GB de HDD e 4 núcleos de processamento, pode ser adquirida por um valor que se aproxima do dobro da configuração anterior. Assim, a aquisição de máquinas mais modestas pode acabar custando mais caro para atingir uma capacidade de processamento teóricamente semelhante. Embora o custo total dos quatro computadores que formam o cluster seja mais alto do que o do computador único, avaliando a quantidade de núcleos e distribuição de memória, sua capacidade prática de processamento pode atingir o dobro e ainda ter uma capacidade de armazenamento, maior.

Para empresas que não necessitam de um supercomputador, a montagem de um cluster com computadores de baixa capacidade pode não ser a melhor opção, já que demanda mais espaço e conhecimento para o funcionamento do

cluster. No entanto, para empresas que precisam de grande capacidade computacional, como em mineração de dados ou de criptomoedas, o agrupamento de computadores com menor capacidade individual pode resultar em um ganho significativo em núcleos de processamento e espaço de armazenamento.

REFERÊNCIAS

[1] Jeanne Behnke. *NASA's Earth Observing System Data and Information System (EOSDIS)*. Rel. técn. 2017.

[2] Elias Rabelo Matos. "Implementação e análise de desempenho do Framework Apache Hadoop e da Biblioteca OpenMPI para Multiplicação de Matrizes com um Cluster de Baixo Custo na Plataforma Raspberry". Em: (2018).

[3] Vítor Pereira. "Instalação e configuração de um Cluster Beowulf". Em: (2016).

[4] Joseph D Sloan. *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI: A Comprehensive Getting-Started Guide*. "O'Reilly Media, Inc.", 2004.

[5] Jammy Jellyfish. "Official Ubuntu Documentation". Em: (2023). Disponível em <https://help.ubuntu.com/>.

[6] Lenimar Nunes de Andrade. "CALCULO NUMÉRICO". Em: ().

[7] Bill McKeeman. "The Computation of Pi by Archimedes". Em: *Matlab Central* (2012).

[8] Renato Ricardo de Paula et al. "Método de Monte Carlo e aplicações". Em: (2014).

[9] Luis André Bazzi Morales. "Criação de um cluster beowulf, desenvolvimento de uma aplicação utilizando processamento paralelo e análise de seu desempenho". Em: (2008).

[10] Valeriu Manuel Ionescu, Manan Patel e Drashti Hindocha. "Alternatives for Running Linux Applications in Windows". Em: *2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. IEEE. 2019, pp. 1–4.