13º Congresso de Inovação, Ciência e Tecnologia do IFSP - 2022

CLUSTER VIRTUALIZADO PARA PROGRAMAÇÃO PARALELA

WAGNER L. R. SERAFIM[1]
RENATO C. BARROS[2]

[1] Bacharelado em Engenharia da Computação, Bolsista PIBIFSP, IFSP, Câmpus Birigui, wagner.rebeque@aluno.ifsp.edu.br.
[2] Professor Dr. em Ciência da computação, Orientador PIBIFSP, Câmpus Birigui, rbarros@ifsp.edu.br.
Área de conhecimento (Tabela CNPq): 1.03.03.04-9 Sistemas de Informação

RESUMO: O projeto tem como objetivo desenvolver um *cluster beowulf* para aplicação e desenvolvimento de programação paralela, visando o estudo e análise de performance de algoritmos, além da construção de uma apostila didática que ensina como montar seu próprio *cluster beowulf*. Os nós de processamento do cluster foram criados por meio do *software* de virtualização Oracle VM Virtualbox. Com isso, foi possível desenvolver um algoritmo de programação paralela (utilizando a API MPI), para realizar testes de desempenho, execução paralela e comparar os benefícios em relação à programação tradicional de sistemas.

PALAVRAS-CHAVE: Cluster; Programação; Performance; Algoritmo.

VIRTUALIZED CLUSTER FOR PARALLEL PROGRAMMING

ABSTRACT:
The project aims to develop a beowulf cluster for the application and development of parallel programming, aiming at the study and analysis of algorithm performance, besides the construction of a didactic workbook that teaches how to build your own beowulf cluster. The processing nodes of the cluster were created using the Oracle VM Virtualbox, a virtualization software. With this, it was possible to develop a parallel programming algorithm (using the MPI API), to perform performance tests, parallel execution and compare the benefits in relation to traditional programming systems.

KEYWORDS: Cluster; Programming; Performance; Algorithm.

INTRODUÇÃO

Quando se utilizam dois ou mais computadores em conjunto para resolver um determinado problema, tem-se o que é denominado de *cluster* que, do inglês, significa agrupamento, aglomeração ou concentração (PITANGA, 2004).

No contexto computacional o *Cluster* relaciona-se a arquitetura capaz de unir dois ou mais computadores para trabalharem juntos como se fossem apenas um. Fraciona-se o processamento total entre os computadores buscando-se um melhor desempenho (PITANGA, 2002).

A ideia deste tipo de sistema foi aplicada pela IBM em 1960 com intuito de montar grandes mainframes, como alternativa viável comercialmente de paralelismo computacional. Os sistemas HASP (Houston Automated Spooling Program) e o JES (Job Entry System) conseguiram distribuir tarefas em mainframes interligados. Com a expansão da tecnologia o *cluster* evoluiu e se dividiu em alguns tipos de aplicações. O modelo de processamento paralelo tem como foco processar grandes tarefas.

Esse tipo de *cluster* consegue transformar uma tarefa complexa em várias outras tarefas mais simples, dividindo-as para os “nós” da rede. Cada “nó” pode executar as pequenas tarefas de forma

mais simples e ágil. Assim, o desempenho é multiplicado pela quantidade de “nós”, aumentando a agilidade no processamento das tarefas.

MATERIAL E MÉTODOS

Para a realização do projeto foram utilizadas as seguintes configurações:

- Processador: Intel i5 12400f de 2,5 GHz;
- Memória RAM: 16 GB;
- Placa gráfica integrada: RTX 3050 com 8 GB de RAM e 2560 Cuda Cores;
- Sistema operacional livre: Ubuntu 22.04 LTS;
- As virtualizações foram feitas utilizando o *software* VM VirtualBox da Oracle.

Configuração de adaptador de rede

Foi utilizado o VirtualBox para criar as máquinas virtuais utilizando o sistema Linux Debian 11. Os nomes utilizados pelas máquinas foram: Servidor, No1 e No2. Seguindo as informações contidas no (Servidor Debian, 2022). A rede foi configurada com o nome *cluster* e o CIDR de rede como 192.168.1.0/24 e suporte DHCP, como mostra a Figura 1:

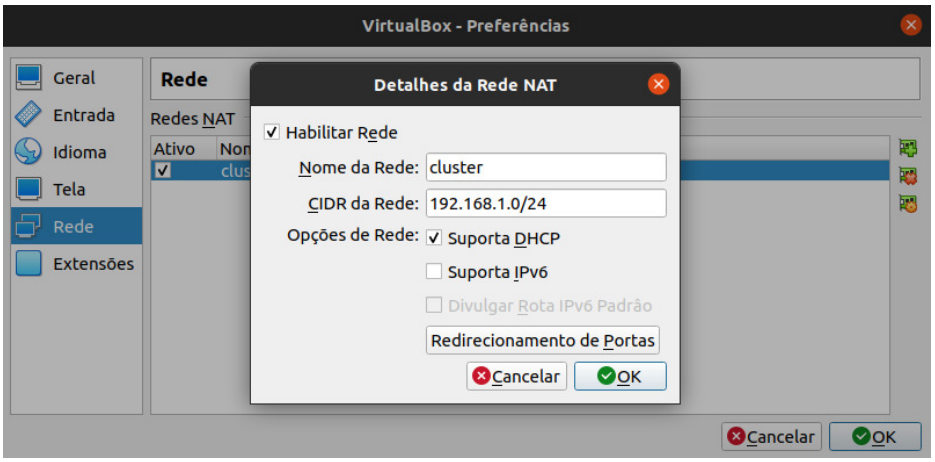

FIGURA 1. Configuração de uma rede no VirtualBox. Fonte: do autor.

Todas as máquinas foram conectadas à rede *cluster*. A máquina Servidor recebeu um segundo adaptador conectado à rede NAT. A Figura 2 apresenta as configurações de rede feitas na máquina Servidor.

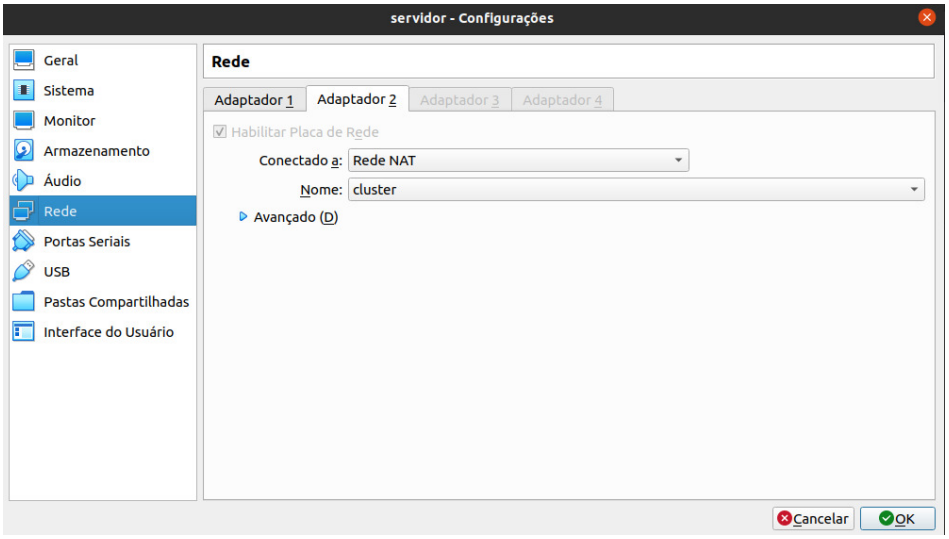

FIGURA 2. Configuração da rede do *cluster* para uma das máquinas criadas. Fonte: do autor.

Configuração dos IPs

Após as configurações iniciais das máquinas virtuais, foram realizadas as instalações do Sistema Operacional Linux e a configuração dos IPs estáticos de cada máquina no arquivo de interfaces localizado em: */etc/network/* (Servidor Debian. Capítulo 5: Configuração de rede, 2022).

Servidor: IP 192.168.1.130

No1: IP 192.168.1.131

No2: IP 192.168.1.132

No arquivo *hosts,* localizado no diretório */etc/* , configurou-se a resolução de nomes da rede.

Acesso Remoto

O acesso remoto foi configurado utilizando o OpenSSH. Para isso, foi executada a sequência de comandos apresentada na caixa de comandos (1):

```
# Instalação do OpenSSH                                          (1)
servidor@servidor: ~$ sudo apt-get install openssh-server
 # Habilitar ssh:
servidor@servidor: ~$ sudo systemctl enable ssh
# Verificar status de funcionamento da ferramenta:
servidor@servidor: ~$ sudo systemctl status ssh
```

Para permitir que o servidor tenha acesso a todas as máquinas do *cluster*, é necessário gerar uma chave ssh no servidor e movê-la para os "nós" da rede. O primeiro passo é criar a chave ssh o seguinte comando na máquina servidor:

```
servidor@servidor: ~$ ssh-keygen
```

Esse comando criará um diretório oculto com a chave ssh armazenada dentro da pasta do usuário (/home/servidor) utilizando o seguinte caminho: */home/servidor/.ssh/id_rsa*. O arquivo Id_rsa é o arquivo que representa a chave de acesso ssh. Para copiar a chave para os demais "nós", utilizou-se os comandos presentes na caixa de comandos (3):

```
#Acessando o diretório oculto .ssh, localizado na pasta home      (3)
servidor@servidor: ~$ cd .ssh
#Movendo a chave:
servidor@servidor: ~$ sudo ssh-copy-id  -i  ~/.ssh/id_rsa.pub no1@no1
servidor@servidor: ~$ sudo ssh-copy-id  -i  ~/.ssh/id_rsa.pub no2@no2
```

Pasta compartilhada para utilização do OpenMPI

Para trabalhar com o multiprocessamento das VMs (máquina virtual), utilizou-se a biblioteca *OpenMPI* em todas as máquinas. Nas máquinas No1 e No2, foi necessário a instalação de bibliotecas extras: *libopenmpi-dev, nfs-common* e *portmap*. Já o servidor, foi preciso apenas uma biblioteca a mais: *nfs-kernel-server*.

Foi necessário instalar o serviço NFS e criar o diretório */home*/clusterdir em todas as máquinas para compartilhar os dados e status do *Cluster.* A caixa de comando (4) apresenta os comandos de instalação citados:

```
sudo apt-get install nfs-kernel-server                           (4)
sudoapt-get install portmap
mkdir Clusterdir
```

Após a instalação das bibliotecas, é necessário alteração do arquivo *exports* da máquina Servidor, localizado no diretório */etc,* adicionando as permissões ao diretório clusterdir:

/home/servidor/clusterdir 192.198.1.0/24(rw, no_subtree_check, async, no_root_squash)

O comando apresentado anteriormente, garante que qualquer IP localizado na faixa da rede cluster, tenha as seguintes permissões no diretório: *rw* = Leitura e escrita; *no_subtree_check* = Não permite criação de subpastas; *assync* = Define o diretório como assíncrono, não permitindo mais de um acesso ao mesmo diretório; *no_root_squash* = Permite o acesso sem necessidade de estar em modo

root. Após esta fase, é necessário habilitar a conexão do serviço NFS e testar o acesso ao servidor, como mostra a caixa de comando (5):

```
sudo  systemctl enablenfs-kernel-server                    (5)
sudo  systemctl statusnfs-kernel-server
sudo /etc/init.d/nfs-kernel-server restart
```

Também é necessário configurar em todos os "nós" o arquivo */etc/fstab*, adicionando a linha de comando, como mostra a caixa de comando (6):

```
192.168.1.130:/home/servidor/clusterdir  /home/no1/clusterdir nfs rw, sync, int 0 0    (6)
```

Esta configuração refere-se a conexão do diretório clusterdir do servidor para o diretório com mesmo nome no "no1", onde é definido as permissões que o "no1" tem no diretório.

Para montar o diretório, executa-se os seguintes comandos nos "nós", apresentado na caixa de comando (7):

```
sudo  showmount –e 192.168.1.130                                                    (7)
sudo mount -t nfs 192.168.1.130:/home/servidor/clusterdir /home/servidor/clusterdir
```

Feito isso, já é possível rodar algoritmos de programação paralela e verificar o desempenho da rede *cluster*. O arquivo de configuração hostfile (onde slots é o número de núcleos alocados) do OpenMPI foi configurado como mostra a seguir:

```
localhost slots=1                                                                   (8)
no1 slots=1
no2 slots=
```

Para a compilação dos algoritmos de programação paralela, utilizando MPI, executou-se o comando apresentado na caixa de comando (8):

```
sudo mpic++ nome_do_Arquivo.c++ -o nome_do_arquivo_compilado                        (8)
```

Para execução dos testes de performance, foi utilizado o comando presente na caixa de comando (9):

```
mpirun -np n_de_processos –hostfile   ./arquivo_host   ./ nome_do_arquivo_compilado   (9)
```

RESULTADOS E DISCUSSÃO

Com as máquinas já configuradas, foi possível executar um teste de performance. O algoritmo de programação paralela desenvolvido executa um cálculo de números primos que resulta em 19 valores. A Figura 3 apresenta o resultado para um único núcleo de processamento. Na Figura 4, é apresentado o resultado para 12 núcleos de processamento.

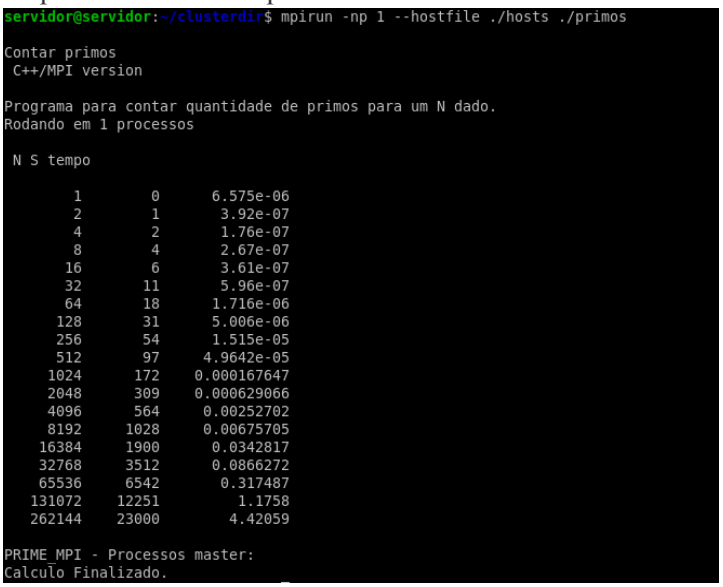

FIGURA 3. Resposta da execução local do algoritmo primo. Fonte: do autor.

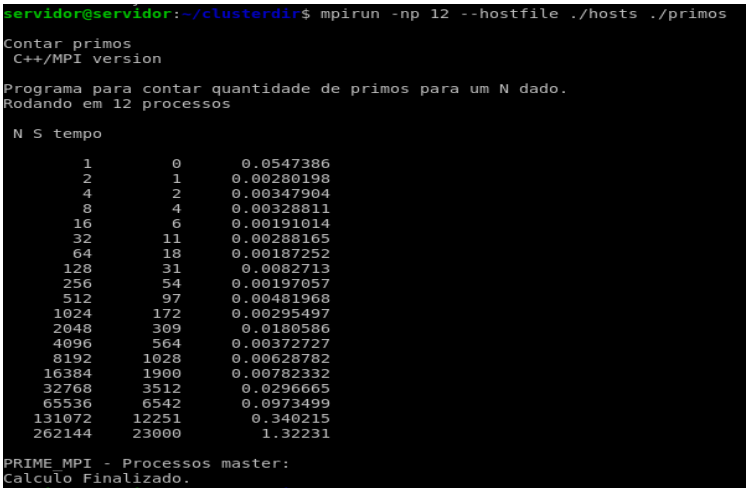

FIGURA 4. Resultado do teste de execução do algoritmo de programação paralela. Fonte: do autor.

A tabela completa com todos os valores testados é apresentada na Tabela 1.

Quantidade de núcleos	Tempo de resposta (segundos)
1	4,42059
2	4,37862
4	2,30951
8	1,20266
12	1,32231

TABELA 1. Resultado do teste de performance com OpenMPI. Fonte: do autor.

Pode-se notar que o tempo de resposta da execução do algoritmo com 12 núcleos, conseguiu superar a execução local, provando que é possível conseguir um aumento de performance executando algoritmos de programação paralela com auxílio do *cluster*.

CONCLUSÕES

Os resultados obtidos pelos testes foram satisfatórios. Observando os resultados das execuções em cada teste, o tempo de resposta da execução do algoritmo para 8 núcleos foi superior a execução com 12 núcleos. O baixo rendimento com 12 núcleos pode ser explicado pela interferência do sistema operacional da máquina host (Windows), pois todos os testes foram realizados utilizando uma máquina virtual. Sendo assim, parte dos recursos de *hardware* foram alocados para manter o Windows e o programa VirtualBox rodando na memória, comprometendo o resultado quando foi alocado muitos núcleos. Mesmo assim, o resultado ainda é superior a um único núcleo.

AGRADECIMENTOS

Agradeço ao Programa Institucional de Bolsas de Iniciação Científica e Tecnológica do Instituto Federal de Educação Ciência e Tecnologia de São Paulo (PIBIFSP), por me proporcionar a oportunidade e apoio para realizar a pesquisa.

REFERÊNCIAS

PITANGA, Marcos. Construindo supercomputadores com linux. 2002. 3º ed. Brasport

Servidor Debian. Como instalar, configurar e administrar um servidor Linux. Disponível em: <https://servidordebian.org/pt/start>. Acesso em 20 de julho de 2021.

Servidor Debian. Capítulo 5: Configuração de rede. Disponível em <https://www.debian.org/doc/manuals/debian-reference/ch05.pt.html#_the_network_interface_with_the_static_ip>. Acesso em 22 de julho de 2021.