Cluster de alta disponibilidade com balanceamento de carga em máquinas virtuais: gerenciando banco de dados MariaDB com galera *cluster*

Samuel Antonio Vieira
Faculdade de Tecnologia – Centro
Paula Souza
Tatuí, SP, Brasil
https://orcid.org/0000-0002-1408-4183

***Abstract*—Clustering is a common issue within data centers. Information is one of the most valuable assets nowadays and data is the raw material for this. In this way, access to information needs to be always available and secure. Security is paramount, from its level of access, to its availability, and availability is what this article is about. Data centers work with redundancy and load balancing systems and employ this to offer a better service in the cloud, *On-Premises* systems can also offer specific services in the same way. These systems rely on services ranging from high availability, load balancing and high performance, in a *cluster* of *hosts* called a *cluster*. MariaDB is an *Livre* database that offers clustering through Galera *Cluster*, with a very complete documentation on the developer's website and it will be shown in a practical way how to install and configure a high availability *cluster* with load balancing in virtual machines on Desktop *hardware*. This article was born as an arm of the research that the author develops in the institution of Higher Education, Faculdade de Tecnologia de Tatuí-SP - Prof. Wilson R. R. de Camargo whose title is "Using free *software* for server clustering and virtualization".**

***Resumo ou Resumen*— A clusterização é um assunto comum dentro dos data centers. A informação é um dos bens mais valiosos nos dias atuais e os dados são a matéria prima para tal. Desta forma o acesso a informação precisa estar sempre disponível e em segurança. A segurança é primordial, desde o seu nível de acesso, até a sua disponibilidade e é na disponibilidade que este artigo fala. Os data centers trabalham com sistemas de redundância e balanceamento de carga e empregam isso para oferecer um serviço melhor na nuvem, sistemas *On-Premises* também podem oferecer serviços específicos da mesma forma. Esses sistemas contam com serviços que variam entre alta disponibilidade, balanceamento de carga e alto desempenho, em um aglomerado de *hosts* chamado *cluster*. O MariaDB é um banco de dados de código aberto que oferece uma clusterização através do Galera *Cluster*, será mostrado de forma prática a sua instalação e configuração de um *cluster* de alta disponibilidade com balanceamento de carga em máquinas virtuais em um *hardware* Desktop. Este artigo nasceu como um braço da pesquisa que o autor desenvolve na instituição de Ensino Superior, a Faculdade de Tecnologia de Tatuí-SP - Prof. Wilson R. R. de Camargo cujo título é "Utilização de *software* livre para clusterização e virtualização de servidores".**

***Palavras-chave—cluster* 1; mariadb 2; linux 3.**

I. INTRODUÇÃO

Cluster não é um assunto novo dentro do mundo da Tecnologia da Informação e Comunicação. O termo *cluster* é definido como um aglomerado de computadores que fornecem uma acesso ao seu usuário como se estivesse utilizando uma única máquina. Cada computador (ou nó) trabalha de forma síncrona (ou assíncrona) para executar as tarefas em conjunto [1].

Sistemas em nuvem utilizam esta tecnologia para fornecer segurança, disponibilidade, poder de processamento e armazenamento, tudo empregado em seus serviços: Software como Serviço, Plataforma como Serviço e Infraestrutura como Serviço.

Desta forma as BigTechs (Google, Amazon, Microsoft, Apple, etc…) têm um sistema *"clusterizado"* para oferecer o melhor serviço para seus clientes, de maneira ininterrupta e segura.

Com isso, sistemas *On-Premises* podem dar a impressão que não se aplicam a esta tecnologia, ou que ela está atrelada a sistemas de alto-desempenho dentro de um *Cluster* Bewolf, ou que uma clusterização tende a fazer mais sentido utilizando máquinas reais.

Outro aspecto sobre sistemas *On-Premises* é o uso da virtualização, muito popular por trazer o isolamento dos serviços nos servidores, entretanto é difícil encontrar um sistema de virtualização que tenha as máquinas virtuais *"clusterizadas"*.

O objetivo deste trabalho é montar um *cluster* de alta-disponibilidade com o sistema gerenciador de banco de dados MariaDB, dentro de máquinas virtuais com sistema Oracle VirtualBox.

Como objetivos específicos será feito:

- Instalação do Sistema Operacional Debian Linux (no hospedeiro e convidados),
- Instalação do Oracle VirtualBox,
- Instalação e configuração do:
- MariaDB,
- Galera *Cluster*

O trabalho visa demonstrar que a clusterização em sistemas virtuais são fáceis de instalação e uso, de forma a

desmistificar seu uso. A clusterização será feita diretamente nos *hosts* convidados.

II. CLUSTER E VIRTUALIZAÇÃO

Em 1960, a Industry Business Machine (IBM) interligou seus *mainframes* para obter processamento paralelo. Sua função era aumentar a eficiência e oferecer viabilidade neste tipo de processamento [2]. Desde então, a clusterização evoluiu, de acordo com as tecnologias emergentes.

A evolução dos *clusters* ganhou força a partir do fim da década de 90, com a depreciação da escala vertical de recursos. No novo milênio, com a massificação da Internet, houve uma necessidade de aumento de desempenho maior que a capacidade de escalabilidade do *hardware*. Nessa época surgiu a escala horizontal de recursos. A cada nova necessidade de aumento de desempenho, um novo host era adicionado ao sistema. O S*oftware Livre* teve um papel importante para este desenvolvimento [3].

Os *hosts* de um *cluster* são chamados de nós (nodos), que estão interligados de forma a executar aplicações ou tarefas. O usuário não percebe que está num sistema *"clusterizado"*. Por exemplo, quando se faz uma consulta do Google, a requisição vai para o seu sistema e é processado pelo seu data center, o usuário não tem controle para onde foi a requisição [1].

O autor afirma que os *clusters* estão divididos em:

Alta disponibilidade (*High Availability (HA) and Failover*): são aqueles que visam oferecer um serviço ininterrupto. Caso haja falha em algum nó outro assumirá a tarefa de disponibilizar o recurso e/ou serviço, é muito comum encontrá-los no modelo nó-mestre e nó(s)-escravo(s);

Balanceamento de carga (*Load Balancing*): neste tipo de *cluster*, trabalha-se com o paradigma nó-mestre-nó-mestre, a cada requisição recebida, o *cluster* envia ao nó disponível. Outra familiaridade é que este tipo de *cluster* trabalha com a combinação HA e *Load Balancing*.

Processamento Distribuído ou Processamento Paralelo: este visa melhorar o desempenho geral de processamento do sistema. As aplicações rodam de forma distribuída nos nodos, formando um super computador.

Cluster Beowulf: Este *cluster* está empregado dentro dos sistemas de Processamento Distribuído, ele foi implementado por Thomas Sterling e Donald Becker em 1994. Sua peculiaridade é usar máquinas de baixo desempenho para criar um super computador para uso científico.

A clusterização capacita a potencialização de um sistema computacional, definir o tipo de *cluster* a ser utilizado pode aumentar a segurança, desempenho e diminuir os custos dos sistemas computacionais. Com máquinas de uso cotidiano pode-se criar máquinas com grande capacidade de processamento, sem a necessidade de hardwares padronizados ou especializados. Por fim, devido a essa heterogeneidade sistêmica, é possível uma escalabilidade, aumentando a quantidade de nós devido às demandas vindouras [1-3].

O VirtualBox é uma ferramenta de virtualização que pertence à Oracle. Trabalha sob a licença GNU General Public License (GPL) versão 2. Este sistema suporta os principais sistemas operacionais de mercado como hospedeiros e convidados [4]. “Virtualização é uma tecnologia que permite criar serviços de TI valiosos usando recursos que estão tradicionalmente vinculados a um determinado *hardware*.” [5]. Desta forma, pode-se dimensionar os serviços em VMs isolados. Graças aos *hypervisors*, o *hardware* é particionado virtualmente, disponibilizando-os aos convidados de acordo com a necessidade de cada.

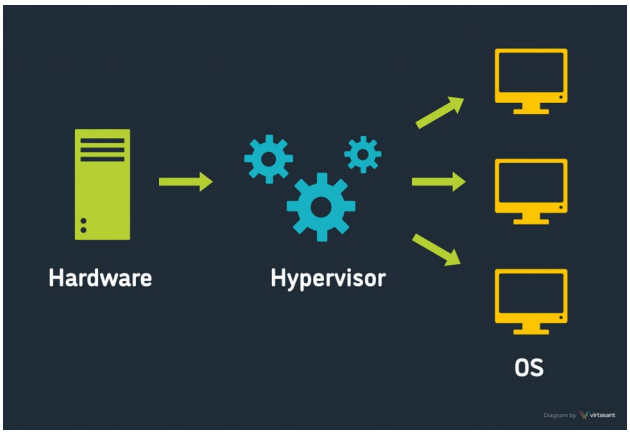

Figura 1: Hypervisors. Fonte: <https://www.virtasant.com/blog/hypervisors-a-comprehensive-guide>. Acesso em: 19 set. 2022.

O *hypervisor* é responsável pelo controle e monitoramento das VMs, é responsável por gerenciar os recursos do *hardware*, como processamento e memória. [6]

III. OS MATERIAIS

Para a máquina hospedeira, foi utilizado o computador:

- Intel(R) Core(TM) i3 CPU 550 @ 3.20GHz,
- 4GB Ram,
- 500 GB de armazenamento,
- Sistema Debian 5.10.136-1 (2022-08-13) x86_64 GNU/Linux.

Com virtualizador:

- Oracle Virtual Box Versão 6.1.36 r152435 (Qt5.15.2).

Para cada máquinas virtuais (VM):

- Um núcleo,
- 1GB de Ram e 20GB de armazenamento,
- Sistema Operacional: Operating System: Debian GNU/Linux 11 (bullseye),
- Kernel: Linux 5.10.0-17-amd64,
- Architecture: x86-64.

Para a versão do MariaDB instalado: 10.5.15-MariaDB-0+deb11u1 for debian-linux-gnu on x86_64 (Debian 11), a replicação ficou por conta do galera-4, 26.4.11-0+deb11u1 em ambos nodos denominados: db-node01 e db-node02.

O balanceamento de carga foi utilizado o HAProxy, versão: 2.2.9-2+deb11u3.

IV. INSTALAÇÃO

A instalação feita foi padrão do sistema, sem interface gráfica, utilizando todo o espaço disponível. As partições de disco no hospedeiro ficaram.

```
root@host1:~# fdisk -l
Disco /dev/sda: 465,76 GiB, 500107862016 bytes, 976773168 setores
Modelo de disco: SAMSUNG HD502HJ
Unidades: setor de 1 * 512 = 512 bytes
Tamanho de setor (lógico/físico): 512 bytes / 512 bytes
Tamanho E/S (mínimo/ótimo): 512 bytes / 512 bytes
Tipo de rótulo do disco: dos
Identificador do disco: 0xcbad9b45

Dispositivo Inicializar    Início       Fim   Setores Tamanho Id Tipo
/dev/sda1   *                2048 974772223 974770176  464,8G 83 Linux
/dev/sda2               974774270 976771071   1996802    975M  5 Estendida
/dev/sda5               974774272 976771071   1996800    975M 82 Linux swap / So
```

Figura 2: Partições de Discos - Físico. Fonte: Autoria Própria.

Nos convidados ficaram:

```
root@db-node01:~# fdisk -l
Disco /dev/sda: 19,53 GiB, 20971520000 bytes, 40960000 setores
Modelo de disco: VBOX HARDDISK
Unidades: setor de 1 * 512 = 512 bytes
Tamanho de setor (lógico/físico): 512 bytes / 512 bytes
Tamanho E/S (mínimo/ótimo): 512 bytes / 512 bytes
Tipo de rótulo do disco: dos
Identificador do disco: 0x75907776

Dispositivo Inicializar   Início      Fim  Setores Tamanho Id Tipo
/dev/sda1   *             2048 37750783 37748736    18G 83 Linux
/dev/sda2              37750784 40959999  3209216    1,5G  5 Estendida
/dev/sda5              37752832 40959999  3207168    1,5G 83 Linux
root@db-node01:~#
```

Figura 3: Partições de Discos - Físico. Fonte: Autoria Própria.

Em ambos, não foi criada nenhuma outra partição. Para as máquinas virtuais, foi criada uma padrão, que foi clonada da seguinte forma:

- Uma máquina com sistema Linux;
- No clone desta primeira máquina, foi instalado o MariaDB e o Galera *Cluster*;
- A segunda máquina foi clonada integralmente;
- A primeira máquina foi instalada o HA Proxy após ela ser clonada (sem o MariaDB instalado)

Nas máquinas virtuais com banco de dados[7], foram instalados o mariadb-server-10.5 e o galera-4, respectivamente, como *root*[1]:

```
# apt update

# apt upgrade (se necessário, fazer o reinício da máquina)

# apt install mariadb-server
```

Após a instalação foi feita a configuração básica no nó. Foi parado o banco de dados para configuração do *cluster:*

```
# service mariadb stop
```

A. *Nó 1 (db-node01):*

```
[galera]
wsrep_on=ON
wsrep_provider=/usr/lib/galera/
libgalera_smm.so
wsrep_cluster_address="gcomm://db-
node01,db-node02"
binlog_format=row
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0
wsrep_cluster_name="galera_cluster"
wsrep_node_address="db-node01"
```

Arquivo 1: /etc/mysql/mariadb.conf.d/60-galera.cnf. Fonte: Autoria Própria.

B. *Nó 2 (db-node02):*

```
[galera]
wsrep_on=ON
wsrep_provider=/usr/lib/galera/
libgalera_smm.so
wsrep_cluster_address="gcomm://db-
node01,db-node02"
binlog_format=row
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0
wsrep_cluster_name="galera_cluster"
wsrep_node_address="db-node02"
```

Arquivo 2: /etc/mysql/mariadb.conf.d/60-galera.cnf. Fonte: Autoria Própria.

1Administrador do sistema.

A configuração dos Arquivos 1 e 2 habilita o *cluster*, configura-se qual biblioteca[2] a ser utilizada, quais os *hosts* que farão parte do *cluster*, o nível de replicação, o tipo de banco de dados, o método de trava de arquivos, quais endereços ips o banco vai responder, o nome do *cluster* e o nome do nó no *cluster*. Após a configuração, o sistema de banco de dados foi iniciado:

```
# galera_new_cluster
# service mariadb start
```

No Nó balanceador, foi instalado o HA-Proxy:

```
# apt update
# apt install haproxy
```

E feito a configuração (db-servidor):

```
global
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon
    # turn on stats unix socket
    stats socket /var/lib/haproxy/stats

defaults
    mode                    tcp
    log                     global
    option                  dontlognull
    option                  redispatch
    retries                 3
    timeout queue           45s
    timeout connect         5s
    timeout client          1m
    timeout server          1m
    timeout check           10s
    maxconn                 3000

frontend  main
    bind 10.132.229.1:3306
    default_backend             app

backend app
    balance     roundrobin
    server  app1 10.132.231.1:3306 maxconn 151 check
    server  app2 10.132.231.2:3306 maxconn 151 check
```

Arquivo 3: /etc/haproxy/haproxy.cfg. Fonte: Autoria Própria.

A configuração do Arquivo 3 é dividida em grupos: *global*, *defaults frontend* mais e *backended app*. No *frontend main* coloca-se o ip do *host* que vai fazer o balanceamento e a porta do serviço balanceado e um apelido para os *hosts* com os serviços reais:

```
frontend  main
    bind 10.132.229.1:3306
    default_backend             app
```

No *backended* é setado os *hosts* com os serviços a serem balanceado, o tipo de balanceamento, ips, portas e quantidade de conexões:

```
backend app
    balance     roundrobin
    server  app1 10.132.231.1:3306
maxconn 151 check
    server  app2 10.132.231.2:3306
maxconn 151 check
```

Então foi iniciado o serviço:

```
# service haproxy start
```

Com todos os serviços funcionando a estrutura ficou conforme a figura:

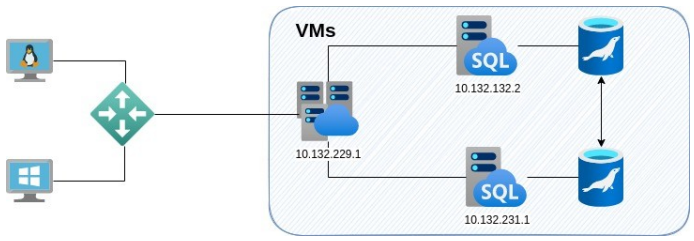

Figura 4: Estrutura Cluster. Fonte: Autoria Própria.

A Figura 4 demonstra como estão organizadas as máquinas virtuais (VMs). Duas máquinas com MariaDB e o Galera *Cluster* trocam informações dentro de uma Alta Disponibilidade *Master-Master*, respectivamente db-node01 (10.132.231.1) e db-node02 (10.132.231.2). O acesso será feito pela máquina chamada db-servidor (10.132.229.1) que fará o balanceamento das requisições.

Para facilitar a comunicação entre elas, também foram alterados os arquivos /etc/*hosts* de todas as máquinas virtuais, adicionando as linhas:

```
10.132.229.1 db-servidor
10.132.231.1 db-node01
10.132.231.2 db-node02
```

Com esta configuração em todas as máquinas do *cluster*, é conhecido os nomes e endereços ips dos nodos, sem a necessidade de um Domain Name System (DNS) para isso.

2Arquivos da biblioteca do sistema, normalmente, arquivos .so nos sistemas Linux. No clsuter: libgalera_smm.so

Dentro do Oracle VirtualBox, as máquinas ficaram conforme a Figura 5.

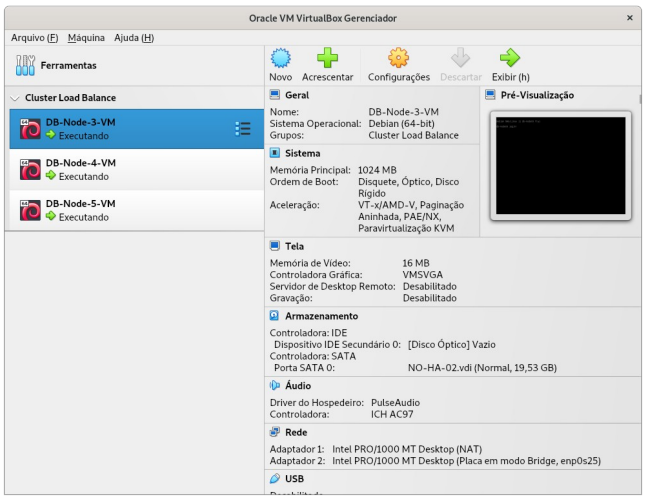

Figura 5: VirtualBox. Fonte: Autoria Própria.

A Figura 5 ilustra a organização do *cluster* dentro do aplicativo, o grupo criado para aninhar as máquinas, chamado *Cluster* Load Balance, possui três VMs: DB-Node-0-VM (db-servidor) que receberá as requisições e fará a distribuição, DB-Node-1-VM (db-node01) e DB-Node-2-VM (db-node02) que são as máquinas virtuais com o MariaDB instalado. Esta organização também facilita a Escalabilidade Horizontal, neste caso, se necessário, clona-se a VM, configura e adiciona ao *cluster*.

V. OS TESTES

Os testes aplicados tiveram o objetivo de avaliar o funcionamento do *cluster*, sem elevar o sistema à sua capacidade máxima de processamento ou fazer algum tipo de *benchmark*. Foi criado um *script* para executar um comando SQL simples com a função de agregação SUM para calcular o atreaibuto *salarioBruto* da tabela *Salarios*. A base de dados foi criada a partir de sites de teste para a criação de bases com dados aleatórios e fictícios.

A. Balanceamento de Carga

A Figura 6 demonstra que quando executados de forma externa, com a utilização da opção *-e* na chamada do *MySQL/MariaDB*, cada requisição é enviada para um servidor diferente, deste modo pode-se verificar o balanceamento de carga no sistema:

```
samuel@samuel-pc:~$ mysql -u $u -p$s -h 10.132.229.1 -e "select @@hostname;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------+
| @@hostname |
+------------+
| db-node01  |
+------------+
samuel@samuel-pc:~$ mysql -u $u -p$s -h 10.132.229.1 -e "select @@hostname;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------+
| @@hostname |
+------------+
| db-node02  |
+------------+
```

Figura 6: Teste de balanceamento. Fonte: Autoria Própria

Foi importada uma base de dados de testes:

```
    # mysql -h 10.132.229.1 -u $u -p$s -e "\. ./db_cluster.sql"
```

O resultado do comando da Figura 7 demonstra que a replicação de dados está funcional. O balanceamento de carga e a alta disponibilidade estão ativas no sistema.

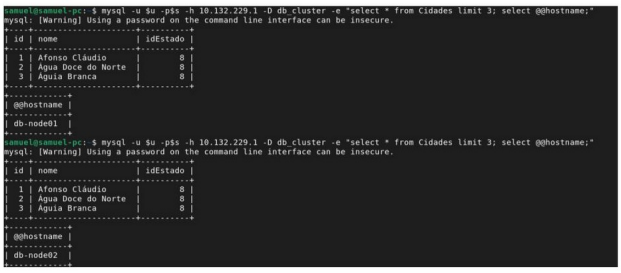

Figura 7: Teste balanceamento com SQL. Fonte: Autoria Própria

Nos testes para execução dos comandos, por questões de segurança, o usuário e senha do banco de dados foram exportados como variável ambiente dentro da sessão, ao fim dos testes, essas variáveis foram apagadas.

Para os testes das Figuras 8 e 9, o *script* criado foi configurado para rodar o comando SQL mil vezes em cada terminal empregado. Foram três terminais que rodam o *script* de forma simultânea, totalizando três mil requisições, e o balanço delas foi:

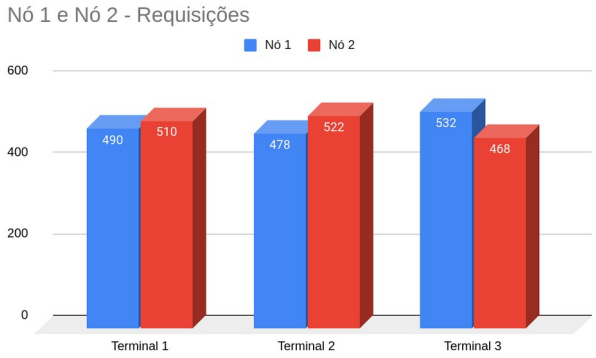

Figura 8: Requisições. Fonte: Autoria Própria.

A Figura 9 demonstra que o sistema manteve as requisições mais balanceadas possível, os números são bem próximos aos terminais. No Terminal 1, o Nó 1 respondeu 490 vezes e o Nó 2 respondeu 510 vezes, uma diferença de 20 requisições a mais para o Nó 2. No Terminal 2, o Nó 1 respondeu 478 vezes e o Nó 2 respondeu 522, uma diferença de 44 requisições a mais para o Nó 2. O Terminal 3 apontou um resultado diferente dos outros, o Nó 1 respondeu 532

vezes e o Nó 2 respondeu 468 vezes, uma diferença de 64 requisições a mais para o Nó 1.

A Figura 10 demonstra o tempos total que cada nó levou para atender todas as requisições.

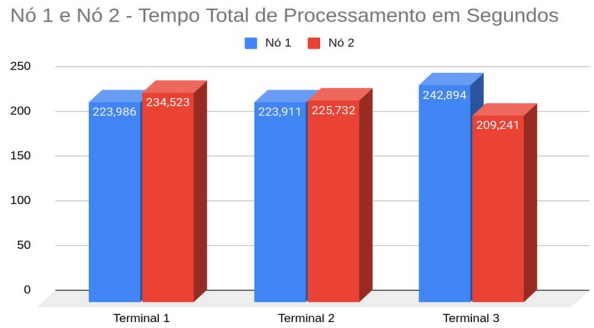

Figura 9: Tempo Total de Execução. Fonte: Autoria Própria.

Para o Terminal 1, o Nó 1 levou 223,98 segundos para responder suas 490 requisições, o Nó 2 levou 234,52 segundos para responder suas 510 requisições, mantendo, visualmente nos gráficos, as proporções das quantidade de respostas em relação ao seu tempo total. Para o Terminal 3, o Nó 1 levou 242,89 segundos para responder suas 532 requisições, o Nó 2 levou 209,24 segundos para responder suas 468 requisições, mantendo também, visualmente nos gráficos, as proporções das quantidade de respostas em relação ao seu tempo total. A diferença ficou para o Terminal 2, onde o Nó 1 levou 223,91 segundos para responder suas 478 requisições, o Nó 2 levou 225,73 segundos para responder suas 522 requisições, distorcendo, visualmente nos gráficos, as proporções das quantidade de respostas em relação ao seu tempo total.

Estes testes preliminares se demonstraram satisfatórios neste momento da pesquisa, possibilitando o andamento para adição de novos nós para execução de novos testes e estresse do sistema.

B. ALTA-DISPONIBILIDADE

O sistema foi proposto para trabalhar na Alta-Disponibilidade no modo *Master-Master*, com base nisso os testes preliminares foram feitos com esse intuito. A sincronização dos dados trouxe um resultado bem satisfatório dentro do modelo empregado. Em relação às máquinas clonadas, o sistema “entendeu” e não fez o um ressincronismo de dados, uma economia de tempo, processamento e recursos. Foram feitos alguns teste de inserção, alteração de exclusão de registros, também foi testado alteração estrutural no banco de dados, todos sincronizados com sucesso, teste também satisfatório.

Para verificar o número de nodos no *cluster*, pode-se utilizar o comando da Figura 10:

Figura 10: Variável Ambiente. Fonte: Autoria Própria.

A variável ambiente wsrep_*cluster*_size (Figura 10) mostra a quantidade de nodos integrado ao *cluster*.

O teste de *failover* não surtiu o resultado esperado, o conceito de uma *cluster* de Alta-Disponibilidade, tende a ter, ao menos, duas máquinas (virtuais ou não) interligadas para manter o sistema funcional. Quando um nó foi retirado do ar, dentro da Alta-Disponibilidade, o sistema parou, necessitando de intervenção para voltar a funcionar. Consultando a documentação do Galera *Cluster*, consta que para um bom funcionamento, o ideal é ter, no mínimo, três nodos. Neste ponto o trabalho esbarrou numa limitação de *hardware*. Um sistema com quatro *threads* e quatro gigabytes de memória poderia não suportar a adição de um novo nó virtual no mesmo *host*. Para este trabalho, este teste foi insatisfatório.

VI. CONCLUSÃO

Este trabalho teve por objetivo demonstrar a instalação de um *cluster* de balanceamento de carga com alta disponibilidade do banco de dados MariaDB com Galera *Cluster* em máquinas virtuais com Oracle VirtualBox. Para isso utilizou-se um computador *desktop*, com sistema Debian Linux instalado, dentro de uma rede interna da Faculdade de Tecnologia de Tatuí.

O sistema foi instalado, configurado e testado. A configuração e documentação para tal foi utilizada a do próprio desenvolvedor do MariaDB. Os testes de importação e manipulação de dados demonstraram a sincronia dos nós que foram gerenciados pelo balanceador de carga Dentro deste mesmo aspecto, o balanceador de carga demonstrou eficiência em redirecionar as requisições para ambos os servidores.

O trabalho esbarrou na limitação de *hardware* ao utilizar um único *host* para virtualização das máquinas e também não optando por deixar o balanceador e um dos *guests* do banco de dados juntos, mantendo o isolamento dos serviços.

Assim este trabalho abre a possibilidade para continuidade da pesquisa, visando a escalabilidade do sistema com a adição de novos *hosts* e *guests*. Com base nos dados coletados, foi possível criar o ponto de partida para continuar este estudo e fazer novos testes com a agregação de novos recursos à estrutura do *cluster*. Com a expansão do modelo, *softwares* de *benchmark* e/ou testes de estresse poderão ser utilizados, assim como o aumento dos terminais para acesso ao sistema, de forma a exigir um maior processamento do *cluster*.

Este trabalho foi concebido dentro da pesquisa de "*clusterização*" que o autor desenvolve dentro da Faculdade de Tecnologia de Tatuí-SP, Centro Paula Souza.

AGRADECIMENTOS

Agradeço à Deus e à minha família pelo apoio e compreensão incondicionais para o desenvolvimento desta e de outras pesquisas ao longo de minha vida profissional.

À equipe organizadora do 19º Congresso Latino-americano de Software Livre e Tecnologias Abertas e Latin.Science pela oportunidade de mostrar parte do meu trabalho de pesquisa.

Ao Centro Paula Souza pelo apoio, confiança e patrocínio do trabalho da minha pesquisa.

À Faculdade de Tecnologia de Tatuí-SP - Prof. Wilson R. R. de Camargo, a quem agradeço nas pessoas do senhor Diretor Prof. Dr. Anderson Luiz de Souza e da senhora Vice-Diretora Prof. Me. Patrícia Glaucia Moreno pelo incentivo e apoio à pesquisa.

À comunidade do Software Livre, por tudo que tem feito.

REFERÊNCIAS

[1] Pitanga M. Computação em *cluster* 2003 Clube do *hardware*. [Internet] 2003 [citado em 01 de abril de 2022] Disponível em: https://www.clubedohardware.com.br/artigos/redes/computa%C3%A7%C3%A3o-em-*cluster*-r33711/

[2] Teles F. O que é *Cluster* e como essa estrutura pode ser benéfica para você. Desk Manager [Internet] 2018 [citado em 25 de abril de 2022] Disponível em: https://deskmanager.com.br/blog/*cluster*/

[3] Perez A. Sobre a História da Computação Distribuída e *clusters* Kubernetes. Data Team Stone [Internet] 2021. [citado em 25 de abril de 2022] Disponível em: https://medium.com/team-data-stone/sobre-a-hist%C3%B3ria-da-computa%C3%A7%C3%A3o-distribu%C3%ADda-e-*clusters*-kubernetes-3d0fe331db7

[4] About VirtualBox. VirtualBox [Internet] 2022. [citado em 19 de setembro de 2022] Disponível em: https://www.virtualbox.org/

[5] O que é virtualização?. RedHat [Internet] 2018. [citado em 01 de agosto de 2022] Disponível em: https://www.redhat.com/pt-br/topics/virtualization/what-is-virtualization

[6] Hajdarbegovic, M. Hypervisors: A Comprehensive Guide. Virtasant [Internet] 2020. [citado em 03 de março de 2022] Disponível em: https://www.virtasant.com/blog/*hypervisors*-a-comprehensive-guide

[7] Getting Started with MariaDB Galera *Cluster*. MariaDB. [Internet] 2013. [citado em 03 de março de 2022] Disponível em: https://mariadb.com/kb/en/getting-started-with-mariadb-galera-*cluster*/