

**Licenciatura em Segurança Informática em Redes de Computadores**

**Sistemas Críticos**

**Trabalho Prático**



**Fábio da Cunha – 8210619**

**Dezembro de 2024**

## Conteúdo

<b>Introdução .....</b>	<b>1</b>
<b>Infraestrutura .....</b>	<b>2</b>
<b>Configuração Comum entre as máquinas .....</b>	<b>8</b>
<b>Configuração do GlusterFs .....</b>	<b>9</b>
<b>Configuração dos servidores Sql1 e Sql2 .....</b>	<b>12</b>
<b>Configuração do pacemaker com o crm .....</b>	<b>14</b>
<b>Configuração dos balanceadores de carga proxy1 e proxy2 .....</b>	<b>15</b>
<b>Instalação do Ganglia .....</b>	<b>18</b>
<b>Instalação do Cacti .....</b>	<b>20</b>
<b>Avaliação do Cluster .....</b>	<b>20</b>
<b>Criação dos Gráficos no Ganglia .....</b>	<b>28</b>
<b>Análise Qualitativa dos resultados .....</b>	<b>34</b>
<b>Anexo .....</b>	<b>37</b>

## ***Introdução***

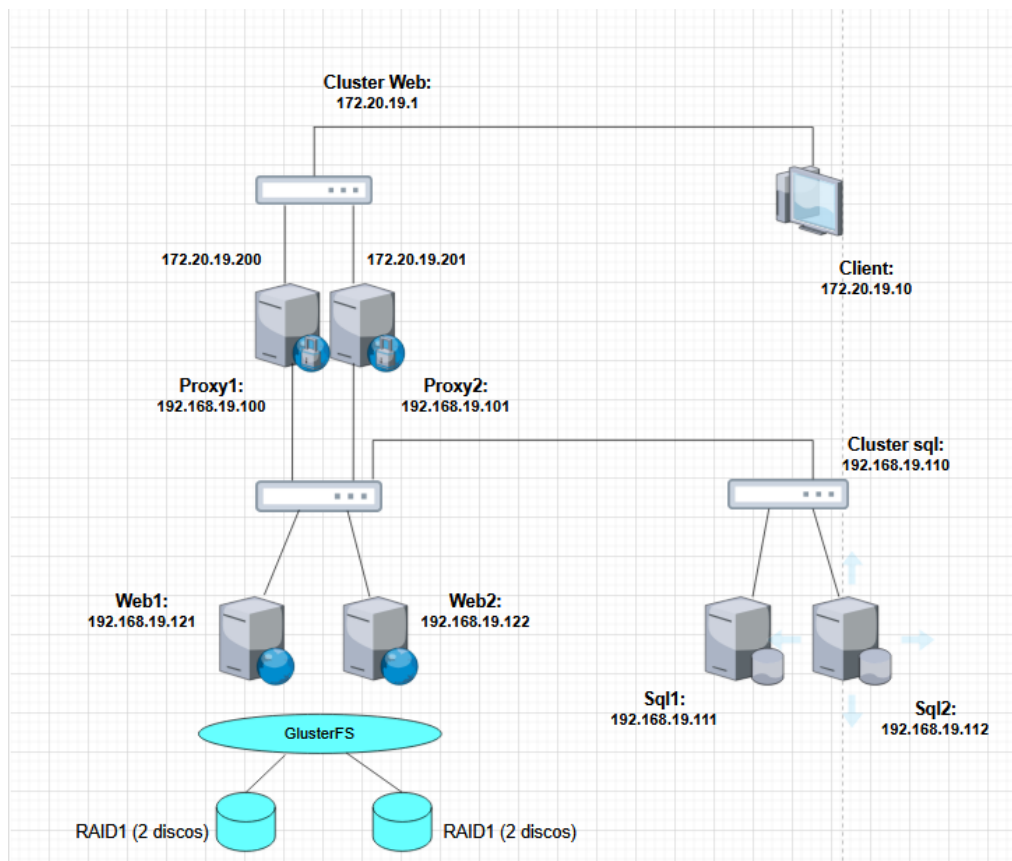
O presente relatório documenta a implementação de uma infraestrutura de cluster altamente disponível, tolerante a falhas e com balanceamento de carga, projetada para atender às necessidades de uma empresa de e-commerce. Dada a criticidade do ambiente de negócios online, a empresa demanda uma solução robusta para hospedar sua página web principal e suas operações relacionadas.

Os objetivos principais deste trabalho incluem a configuração de servidores web e SQL em alta disponibilidade, o uso de sistemas de arquivos distribuídos para garantir a redundância de dados, e a aplicação de balanceamento de carga para otimizar o desempenho e a resiliência. Adicionalmente, o projeto explora a automação da infraestrutura, onde utilizei o Vagrant, bem como, avalia o impacto de diferentes níveis de carga no desempenho do sistema.

Ao longo deste relatório, serão detalhadas as escolhas de configuração, os procedimentos realizados e os resultados obtidos. Este trabalho é de suma importância, pois reflete desafios reais enfrentados em infraestruturas críticas, proporcionando uma visão prática das melhores práticas em engenharia de sistemas.

## Infraestrutura

O diagrama da infraestrutura (Figura 1) ilustra os componentes essenciais que compõem o cluster, incluindo servidores web, proxies, servidores de base de dados e um sistema de ficheiros distribuído. A configuração proposta inclui dois servidores web (Web1 e Web2), que operam em balanceamento de carga através dos proxies Proxy1 e Proxy2. O armazenamento partilhado entre os servidores é garantido pelo GlusterFS, com discos configurados em RAID1 para redundância. A base de dados é implementada com duas instâncias (SQL1 e SQL2) utilizando o MariaDB em modo ativo/passivo. O balanceamento de carga também é aplicado para garantir a disponibilidade contínua, mesmo em caso de falha de um dos componentes.



Para montar este cenário principalmente na criação e configurações iniciais das máquinas utilizei a ferramenta vagrant para a automatização das mesmas.

## Configuração

### Vagrant

Optei por utilizar o Vagrant na implementação do cluster devido à sua capacidade de automatizar e facilitar a criação de máquinas virtuais de forma consistente e replicável. Além disso, o Vagrant permite o provisionamento automático dos serviços necessários, como Nginx, MariaDB e GlusterFS, tornando o processo mais rápido e eficiente.

Visto que o primeiro ponto consiste em criar as máquinas virtuais, é necessário criar e configurar o ficheiro Vagrantfile onde definimos todas as máquinas e as suas especificações.

```
Vagrant.configure("2") do |config|

  # Função para configurar o RAID em uma máquina virtual
  def configurar RAID(machine)
    machine.vm.provision "shell", inline: <<-SHELL
      sudo apt-get update
      sudo apt-get install -y mdadm
      sudo systemctl stop mdadm
      sudo mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/sdc /dev/sdd <<EOF
yes
EOF
      sudo mdadm --wait /dev/md0
      sudo mkfs.ext4 /dev/md0
      sudo mkdir -p /raid1
      sudo mount /dev/md0 /raid1
      echo '/dev/md0 /raid1 ext4 defaults,nofail 0 0' | sudo tee -a /etc/fstab
      sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf
      sudo update-initramfs -u
    SHELL
  end
end
```

Nesta parte temos uma função que vai ser invocada na configuração de cada uma das máquinas com o objetivo de configurar o RAID1.

A configuração raid instala o pacote necessário para criar a raid, seguida estamos a definir o sistema de ficheiros como Ext4, criar uma pasta onde será montado (/raid1), montar temporariamente e acrescentar uma entrada no ficheiro /etc/fstab para que a Raid seja montada no arranque da máquina.

É possível especificar para cada máquina o sistema operativo, redes e endereços IP, pastas partilhadas entre o host e a máquina, correr comandos na Shell, etc.

```

# Configuração da máquina 'web1'
config.vm.define "web1" do |web1| ...
end

# Configuração da máquina 'web2'
config.vm.define "web2" do |web2| ...
end

# Configuração do banco de dados 'sql1'
config.vm.define "sql1" do |sql1| ...
end

# Configuração do banco de dados 'sql2'
config.vm.define "sql2" do |sql2| ...
end

# Configuração do proxy 'proxy1'
config.vm.define "proxy1" do |proxy1| ...
end

# Configuração do proxy 'proxy2'
config.vm.define "proxy2" do |proxy2| ...
end

end

```

Na imagem podemos constatar que temos 6 máquinas sendo dois servidores web (web1 e web2) que vão suportar os serviços (Cacti, Ganglia e Nginx), dois servidores de base de dados (Sql1 e Sql2) com os serviços (MariaDB, Pacemaker, Corosync) e por fim os balanceadores de carga (proxy1 e proxy2) que vão suportar os serviços (HAproxy, Pacemaker, Corosync).

## Máquina web1

```
# Configuração da máquina 'web1'
config.vm.define "web1" do |web1|
  web1.vm.box = "ubuntu/bionic64"
  web1.vm.network "private_network", ip: "192.168.19.121"
  web1.vm.hostname = "web1"

  config.vm.synced_folder "partilha/", "/vagrant"

  web1.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

  # Adicionar discos para RAID
  web1.vm.disk :disk, size: "3GB", name: "sdc"
  web1.vm.disk :disk, size: "3GB", name: "sdd"

  # Provisionamento para configurar RAID
  configurar_raid(web1)

  # Provisionamento com scripts organizados inline
  web1.vm.provision "shell", inline: <<-SHELL
    sudo hostnamectl set-hostname web1
    chmod +x /vagrant/*.sh
    bash /vagrant/setup-inicial.sh
    bash /vagrant/setup-nginx.sh
    bash /vagrant/setup-glusterfs.sh
    sudo reboot
  SHELL
end
```

Na imagem temos a definição da primeira máquina a web1 onde configurei o hostname, o endereço ip, pasta compartilhada.

Também criei dois discos de tamanho de 3GB cada para a configuração do RAID como é solicitado no enunciado do trabalho, em baixo temos a invocação da função responsável por todo o processo de criação e montagem do RAID.

Por fim temos a parte do provisionamento onde tem scripts que vão ser executados na inicialização da máquina, esses scripts possuem alguns comandos de configuração, vou demonstrar mais abaixo nesse relatório

As outras máquinas seguem o mesmo processo, o que diferem são os scripts, hostname e os endereços IPs.

## Máquina web2

```
# Configuração da máquina 'web2'
config.vm.define "web2" do |web2|
  web2.vm.box = "ubuntu/bionic64"
  web2.vm.network "private_network", ip: "192.168.19.122"
  web2.vm.hostname = "web2"
  config.vm.synced_folder "partilha/", "/vagrant"
  web2.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

  # Adicionar discos para RAID
  web2.vm.disk :disk, size: "3GB", name: "sdc"
  web2.vm.disk :disk, size: "3GB", name: "sdd"

  # Provisionamento para configurar RAID
  configurar_raid(web2)

  # Provisionamento com scripts organizados inline
  web2.vm.provision "shell", inline: <<-SHELL
    sudo hostnamectl set-hostname web2
    chmod +x /vagrant/*.sh
    bash /vagrant/setup-inicial.sh
    bash /vagrant/setup-nginx.sh
    bash /vagrant/setup-glusterfs.sh
    sudo reboot
  SHELL
end
```

## Máquina Sql1

```
# Configuração do banco de dados 'sql1'
config.vm.define "sql1" do |sql1|
  sql1.vm.box = "ubuntu/bionic64"
  sql1.vm.network "private_network", ip: "192.168.19.111"
  sql1.vm.hostname = "sql1"
  config.vm.synced_folder "partilha/", "/vagrant"
  sql1.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

  # Adicionar discos para RAID
  sql1.vm.disk :disk, size: "3GB", name: "sdc"
  sql1.vm.disk :disk, size: "3GB", name: "sdd"

  # Provisionamento para configurar RAID
  configurar_raid(sql1)

  # Provisionamento com scripts organizados inline
  sql1.vm.provision "shell", inline: <<-SHELL
    sudo hostnamectl set-hostname sql1
    chmod +x /vagrant/*.sh
    bash /vagrant/setup-inicial.sh
    bash /vagrant/setup-mariadb.sh
    bash /vagrant/setup-glusterfs.sh
    sudo reboot
  SHELL
```



## Máquina Sql2

```
# Configuração do banco de dados 'sql2'
config.vm.define "sql2" do |sql2|
  sql2.vm.box = "ubuntu/bionic64"
  sql2.vm.network "private_network", ip: "192.168.19.112"
  sql2.vm.hostname = "sql2"
  config.vm.synced_folder "partilha/", "/vagrant"
  sql2.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

  # Adicionar discos para RAID
  sql2.vm.disk :disk, size: "3GB", name: "sdc"
  sql2.vm.disk :disk, size: "3GB", name: "sdd"

  # Provisionamento para configurar RAID
  configurar_raid(sql2)

  # Provisionamento com scripts organizados inline
  sql2.vm.provision "shell", inline: <<-SHELL
    sudo hostnamectl set-hostname sql2
    chmod +x /vagrant/*.sh
    bash /vagrant/setup-inicial.sh
    bash /vagrant/setup-mariadb.sh
    bash /vagrant/setup-glusterfs.sh
    sudo reboot
  SHELL
end
```

## Máquina proxy1

```
# Configuração do proxy 'proxy1'
config.vm.define "proxy1" do |proxy1|
  proxy1.vm.box = "ubuntu/bionic64"
  proxy1.vm.network "private_network", ip: "192.168.19.100"
  proxy1.vm.network "private_network", ip: "172.20.19.200"
  proxy1.vm.hostname = "proxy1"
  config.vm.synced_folder "partilha/", "/vagrant"
  proxy1.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

  # Provisionamento com scripts organizados inline
  proxy1.vm.provision "shell", inline: <<-SHELL
    sudo hostnamectl set-hostname proxy1
    chmod +x /vagrant/*.sh
    bash /vagrant/setup-inicial.sh
    bash /vagrant/setup-haproxy.sh
    sudo reboot
  SHELL
end
```

Neste já não temos a criação dos discos porque não vai ser utilizado o RAID, temos também a configuração de dois adaptadores, uma para a comunicação com os

servidores WEB e o outro para a comunicação com o cliente. O mesmo segue para o proxy2.

Máquina proxy2

```
# Configuração do proxy 'proxy2'
config.vm.define "proxy2" do |proxy2|
  proxy2.vm.box = "ubuntu/bionic64"
  proxy2.vm.network "private_network", ip: "192.168.19.101"
  proxy2.vm.network "private_network", ip: "172.20.19.201"
  proxy2.vm.hostname = "proxy2"
  config.vm.synced_folder "partilha/", "/vagrant"
  proxy2.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

  # Provisionamento com scripts organizados inline
  proxy2.vm.provision "shell", inline: <<-SHELL
    sudo hostnamectl set-hostname proxy2
    chmod +x /vagrant/*.sh
    bash /vagrant/setup-inicial.sh
    bash /vagrant/setup-haproxy.sh
    sudo reboot
  SHELL
end
end
```

A máquina cliente vai ser o meu próprio pc.

## Configuração Comum entre as máquinas

```
$ setup-inicial.sh X
C: > Users > fabio > OneDrive > Área de Trabalho > Cluster SC > partilha > $ setup-inicial.sh
1  #!/bin/bash
2
3  # Configuração Inicial
4  sudo apt-get update -y && sudo apt-get upgrade -y
```

A configuração comum em todas as máquinas consiste em atualizar o gestor de pacotes para obter as versões mais atualizadas e atualizar o sistema e os seus pacotes para a versão mais recente. Este script é executado automaticamente em todas as máquinas.

## Configuração do GlusterFs

Para configurar o GlusterFS o primeiro passo consiste em instalar o glusterfs-server, ativar o serviço e criar as pastas onde irá funcionar. Este script é executado automaticamente nas máquinas web e sql.

```
C: > Users > fabio > OneDrive > Área de Trabalho > Cluster SC > partilha > $ setup-glusterfs.sh
1  #!/bin/bash
2
3  # Install GlusterFS
4  sudo apt install glusterfs-server -y
5
6  # Enable and start the service
7  sudo systemctl enable --now glusterd
8
9  # Create the Folder where the GlusterFS volume will be mounted
10 sudo mkdir -p /raid1/glusterfs
```

Depois das máquinas necessárias já terem o Gluster instalado, fui na máquina web1 usar o script glusterfs.sh que vai criar os volumes, conectar as máquinas através do gluster peer probe e inciar o volume www\_storage para os servidores web e sql\_storage para os servidores sql

```
$ glusterfs.sh X $ mount-glusterfs.sh U Vagrantfile U
partilha > $ glusterfs.sh
1  #!/bin/bash
2
3  set -e # Parar o script em caso de erro
4
5  # Variáveis
6  STORAGE_DIR="/raid1/glusterfs"
7  SQL_VOLUME="sql_storage"
8  WWW_VOLUME="www_storage"
9  WEB1="192.168.19.121"
10 WEB2="192.168.19.122"
11 SQL1="192.168.19.111"
12 SQL2="192.168.19.112"
13
14 LOG_FILE="/var/log/glusterfs_create_volumes.log"
15
16 log() {
17     echo "$1" | tee -a "$LOG_FILE"
18 }
19
20 # Adicionar peers ao cluster
21 log "Adicionando peers ao cluster GlusterFS..."
22 for PEER in $WEB1 $WEB2 $SQL1 $SQL2; do
23     log "Adicionando peer: $PEER"
24     sudo gluster peer probe "$PEER" || log "Peer $PEER já adicionado ou indisponível."
25 done
26
27 # Criar volume para SQL
28 log "Criando volume GlusterFS para SQL..."
29 sudo gluster volume create "$SQL_VOLUME" replica 2 transport tcp \
30     "$SQL1:$STORAGE_DIR" "$SQL2:$STORAGE_DIR" force || log "Volume SQL já existe."
31 sudo gluster volume start "$SQL_VOLUME"
32
33 # Criar volume para WWW
34 log "Criando volume GlusterFS para WWW..."
35 sudo gluster volume create "$WWW_VOLUME" replica 2 transport tcp \
36     "$WEB1:$STORAGE_DIR" "$WEB2:$STORAGE_DIR" force || log "Volume WWW já existe."
37 sudo gluster volume start "$WWW_VOLUME"
38
39 log "Volumes GlusterFS criados com sucesso."
40
```

Com o comando:

1. `Sudo gluster peer status`

Para verificar os peers conectados ao cluster

```
vagrant@web1:/vagrant$ sudo gluster peer status
Number of Peers: 3

Hostname: 192.168.19.122
Uuid: 0820d077-8202-482c-b531-00d58b55156d
State: Peer in Cluster (Connected)

Hostname: 192.168.19.111
Uuid: 1df325c6-d6f4-4b44-9d61-a24e510612a6
State: Peer in Cluster (Connected)

Hostname: 192.168.19.112
Uuid: 9e120c05-5bbc-4edc-8a64-004da7d38a68
State: Peer in Cluster (Connected)
vagrant@web1:/vagrant$
```

Agora com o Gluster instalado nas máquinas web e sql e com os peers devidamente configurados passamos para a parte de montar o volume nas respectivas pastas, para web (cluster/www) e para sql (cluster/sql), para isso criei um script mount-glusterfs.sh que será o responsável pela montagem.

```
$ glusterfs.sh U $ mount-glusterfs.sh U X Vagrantfile U
partilha > $ mount-glusterfs.sh
1  #!/bin/bash
2
3  set -e # Parar o script em caso de erro
4
5  # Variáveis
6  SQL_VOLUME="sql_storage"
7  WWW_VOLUME="www_storage"
8  SQL1="192.168.19.111"
9  WEB1="192.168.19.121"
10
11 LOG_FILE="/var/log/glusterfs_mount.log"
12
13 log() {
14     echo "$1" | tee -a "$LOG_FILE"
15 }
16
17 # Função para montar o GlusterFS no SQL
18 configure_sql_mount() {
19     log "Configurando montagem do GlusterFS para SQL em /cluster/sql..."
20
21     # Criar o diretório
22     sudo mkdir -p /cluster/sql
23
24     # Atualizar /etc/fstab
25     sudo sed -i "/$SQL1:/V$SQL_VOLUME \cluster\sql glusterfs/d" /etc/fstab
26     echo "$SQL1:$SQL_VOLUME /cluster/sql glusterfs defaults,_netdev 0 0" | sudo tee -a /etc/fstab
27
28     # Montar o volume se necessário
29     if mount | grep -q '/cluster/sql'; then
30         log "GlusterFS já está montado em /cluster/sql."
31     else
32         log "Montando o GlusterFS para SQL..."
33         sudo mount -t glusterfs "$SQL1:$SQL_VOLUME" /cluster/sql
34     fi
35
36     # Ajustar permissões
37     log "Ajustando permissões para mysql:mysql em /cluster/sql..."
38     sudo chown -R mysql:mysql /cluster/sql
39     sudo chmod -R 0755 /cluster/sql
40 }
41
42 # Função para montar o GlusterFS no Web
43 configure_www_mount() {
44     log "Configurando montagem do GlusterFS para WWW em /cluster/www..."
45
46     # Criar o diretório
47     sudo mkdir -p /cluster/www
48
49     # Atualizar /etc/fstab
50     sudo sed -i "/$WEB1:/V$WWW_VOLUME \cluster\www glusterfs/d" /etc/fstab
51     echo "$WEB1:$WWW_VOLUME /cluster/www glusterfs defaults,_netdev 0 0" | sudo tee -a /etc/fstab
52
53     # Montar o volume se necessário
54     if mount | grep -q '/cluster/www'; then
55         log "GlusterFS já está montado em /cluster/www."
56     else
57         log "Montando o GlusterFS para WWW..."
58         sudo mount -t glusterfs "$WEB1:$WWW_VOLUME" /cluster/www
59     fi
60
61     # Ajustar permissões
62     log "Ajustando permissões para www-data:www-data em /cluster/www..."
63     sudo chown -R www-data:www-data /cluster/www
64     sudo chmod -R 0755 /cluster/www
65 }
66
67 # Identificar o tipo de máquina
68 HOSTNAME=$(hostname)
69
70 log "Iniciando configuração do GlusterFS na máquina $HOSTNAME..."
71
72 if [[ "$HOSTNAME" == "sql1" || "$HOSTNAME" == "sql2" ]]; then
73     configure_sql_mount
74 elif [[ "$HOSTNAME" == "web1" || "$HOSTNAME" == "web2" ]]; then
75     configure_www_mount
76 else
77     log "Erro: Este script deve ser executado em sql1, sql2, web1 ou web2." && exit 1
78 fi
79
80 log "Configuração do GlusterFS concluída na máquina $HOSTNAME."
81
```

```
$ glusterfs.sh U $ mount-glusterfs.sh U X Vagrantfile U
partilha > $ mount-glusterfs.sh
43 configure_www_mount() {
44     sudo mkdir -p /cluster/www
45
46     # Atualizar /etc/fstab
47     sudo sed -i "/$WEB1:/V$WWW_VOLUME \cluster\www glusterfs/d" /etc/fstab
48     echo "$WEB1:$WWW_VOLUME /cluster/www glusterfs defaults,_netdev 0 0" | sudo tee -a /etc/fstab
49
50     # Montar o volume se necessário
51     if mount | grep -q '/cluster/www'; then
52         log "GlusterFS já está montado em /cluster/www."
53     else
54         log "Montando o GlusterFS para WWW..."
55         sudo mount -t glusterfs "$WEB1:$WWW_VOLUME" /cluster/www
56     fi
57
58     # Ajustar permissões
59     log "Ajustando permissões para www-data:www-data em /cluster/www..."
60     sudo chown -R www-data:www-data /cluster/www
61     sudo chmod -R 0755 /cluster/www
62 }
63
64 # Identificar o tipo de máquina
65 HOSTNAME=$(hostname)
66
67 log "Iniciando configuração do GlusterFS na máquina $HOSTNAME..."
68
69 if [[ "$HOSTNAME" == "sql1" || "$HOSTNAME" == "sql2" ]]; then
70     configure_sql_mount
71 elif [[ "$HOSTNAME" == "web1" || "$HOSTNAME" == "web2" ]]; then
72     configure_www_mount
73 else
74     log "Erro: Este script deve ser executado em sql1, sql2, web1 ou web2." && exit 1
75 fi
76
77 log "Configuração do GlusterFS concluída na máquina $HOSTNAME."
78
79
80
81
```

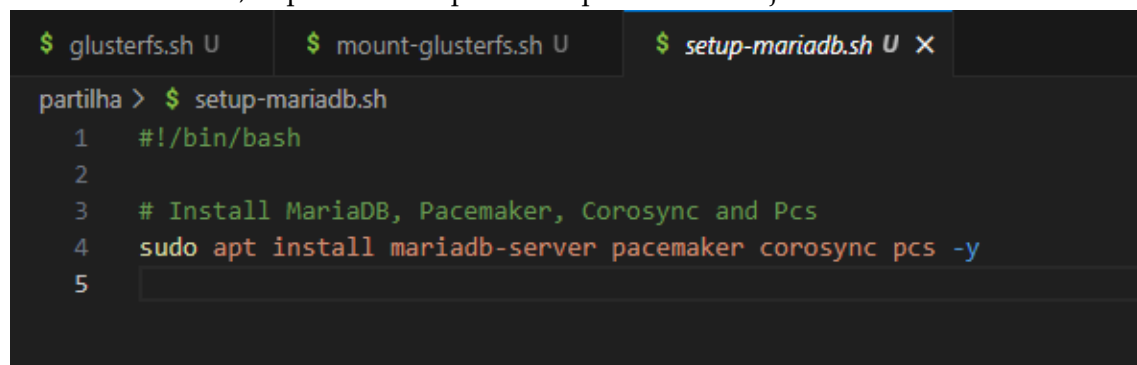
Este script automatiza a configuração e montagem do sistema de ficheiros distribuído **GlusterFS** nos servidores do cluster, com base no hostname da máquina onde é executado. Ele realiza as seguintes ações:

- ⊗ Configura e monta volumes do GlusterFS em diretórios específicos:
  - /cluster/sql para servidores SQL.
  - /cluster/www para servidores Web.
- ⊗ Garante que as permissões corretas sejam aplicadas aos diretórios montados.
- ⊗ Atualiza o arquivo /etc/fstab para que os volumes do GlusterFS sejam montados automaticamente em reinicializações futuras.
- ⊗ Ajusta as configurações automaticamente com base no tipo de máquina (servidores SQL ou Web).

## Configuração dos servidores Sql1 e Sql2

No processo de configuração desses servidores utilizei ferramentas como MariaDB para o suporte à base de dados e para garantir que fossem executadas em modo ativo/passivo utilizei as ferramentas corosync e pacemaker

No vagrantfile executo um script setup-mariadb.sh que fica responsável por instalar essas ferramentas, o que faz com que as máquinas iniciam já com essas ferramentas.



The screenshot shows three terminal windows at the top. The first window has the command `$ glusterfs.sh U`. The second window has `$ mount-glusterfs.sh U`. The third window has `$ setup-mariadb.sh U` with a close button. Below these, a larger terminal window shows the execution of `partilha > $ setup-mariadb.sh`. The script content is as follows:

```
1  #!/bin/bash
2
3  # Install MariaDB, Pacemaker, Corosync and Pcs
4  sudo apt install mariadb-server pacemaker corosync pcs -y
5
```

Já feitas as instalações fiz o move de todos os ficheiros do diretório padrão do mariadb para o diretório /cluster/sql onde devem ficar salvos os ficheiros da base de dados que vão ser replicados pelo GlusterFs para as duas máquinas SQL.

```
1. sudo mv /var/lib/mysql/* /cluster/sql/
```

Depois tinha de configurar o mariadb para que apontasse para o diretório, mudar o datadir para → datadir = /cluster/sql

```
1. sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
2.
```

O ficheiro de configuração encontra-se disponível na pasta do projeto

A seguir reiniciar o mariadb

1. `sudo systemctl start mariadb`
  - 2.

Feito isso, fiz a configuração para que esses dois servidores funcionem em modo ativo/passivo.

Primeiro configurei o ficheiro `/etc/hosts` para que os dois servidores possam reconhecer um ao outro (nas duas máquinas).

```
$ hosts_sql U X
partilha > $ hosts_sql
1 127.0.0.1 localhost
2
3 # The following lines are desirable for IPv6 capable hosts
4 ::1 ip6-localhost ip6-loopback
5 fe00::0 ip6-localnet
6 ff00::0 ip6-mcastprefix
7 ff02::1 ip6-allnodes
8 ff02::2 ip6-allrouters
9 ff02::3 ip6-allhosts
10 127.0.1.1 ubuntu-bionic ubuntu-bionic
11
12 127.0.2.1 sql1 sql1
13 192.168.19.111 sql1
14 192.168.19.112 sql2
15
```

Passei depois para a configuração do corosync, editando o ficheiro `/etc/corosync/corosync.conf` de modo a incluir os dois servidores

```
corosync-sql.conf U X
partilha > corosync-sql.conf
25 logging {
43     logger_subsys {
45         debug: off
46     }
47 }
48
49 quorum {
50     # Enable and configure quorum subsystem (default: off)
51     # see also corosync.conf.5 and votequorum.5
52     provider: corosync_votequorum
53     expected_votes: 1
54 }
55
56 nodelist {
57     # Change/uncomment/add node sections to match cluster configuration
58
59     node {
60         name: node1
61         nodeid: 1
62         ring0_addr: 192.168.19.111
63     }
64     node {
65         name: node2
66         nodeid: 2
67         ring0_addr: 192.168.19.112
68     }
69 }
```

## Configuração do pacemaker com o crm

Para entrar no crm para iniciar a configuração do pacemaker executamos o comando

```
1. crm configure
```

No crm:

```
1. primitive p_mariadb ocf:heartbeat:mysql params config="/etc/mysql/mariadb.conf.d/50-server.cnf" datadir="/cluster/sql" user="mysql" op monitor interval="20s" timeout="30s"
2.
3. primitive p_vip ocf:heartbeat:IPaddr2 params ip="192.168.19.110" cidr_netmask="24" op monitor interval="10s"
4.
5. property cib-bootstrap-options: stonith-enabled=false no-quorum-policy=ignore cluster-infrastructure=corosync
6.
7. colocation col_mysql_with_vip inf: p_mariadb p_vip
8. order o_mysql_after_vip inf: p_vip p_mariadb
9.
```

A configuração define dois recursos principais no cluster para alta disponibilidade:

- **p\_mariadb:** Gerencia o MariaDB com monitoramento periódico e armazena os dados em /cluster/sql.
- **p\_vip:** Configura o IP virtual 192.168.19.110 para acesso ao serviço, monitorando-o constantemente.



As regras de colocação garantem que o MariaDB só funcione em um nó com o IP virtual ativo, e a ordem especifica que o IP deve ser iniciado antes do MariaDB. A política do cluster desativa STONITH e permite operação mesmo sem quórum.

Essa configuração garante alta disponibilidade do serviço MariaDB no cluster. Esses mecanismos asseguram que o MariaDB permaneça disponível com o menor tempo de interrupção possível.

Assim temos os dois servidores SQL funcionando com MariaDB em modo ativo/passivo, com ele configurado também o virtualIp que será o ponto fixo de acesso dos clientes ao MariaDB independentemente do servidor ativo.

## ***Configuração dos balanceadores de carga proxy1 e proxy2***

A configuração desses balanceadores é bastante parecida com a configuração dos SQLs, porque eles também funcionam em modo ativo/passivo a configuração dos ficheiros /etc/hosts e do /etc/corosync/corosync.conf são os mesmos só muda mesmo os endereços IPs

```
$ hosts_proxy U X
partilha > $ hosts_proxy
1 127.0.0.1 localhost
2
3 # The following lines are desirable for IPv6 capable hosts
4 ::1 ip6-localhost ip6-loopback
5 fe00::0 ip6-localnet
6 ff00::0 ip6-mcastprefix
7 ff02::1 ip6-allnodes
8 ff02::2 ip6-allrouters
9 ff02::3 ip6-allhosts
10 127.0.1.1 ubuntu-bionic ubuntu-bionic
11
12 127.0.2.1 proxy1 proxy1
13 172.20.19.200 proxy1
14 172.20.19.201 proxy2
15
```

```
proxy_corosync.conf U X
partilha > proxy_corosync.conf
78 logging {
97     # (unless you are only logging to syslog, where double
98     # timestamps can be annoying).
99     timestamp: on
100     logger_subsys {
101         subsys: QUORUM
102         debug: off
103     }
104 }
105
106 quorum {
107     # Enable and configure quorum subsystem (default: off)
108     # see also corosync.conf.5 and votequorum.5
109     provider: corosync_votequorum
110     two_node: 1
111 }
112
113 nodelist {
114     # Change/uncomment/add node sections to match cluster configuration
115
116     node {
117         name: proxy1
118         nodeid: 1
119         ring0_addr: 172.20.19.200
120     }
121     node {
122         name: proxy2
123         nodeid: 2
124         ring0_addr: 172.20.19.201
125     }
126 }
127
128
```

Nos proxies também configurei o ficheiro `/etc/haproxy/haproxy.cfg`

```
haproxy.cfg U X
partilha > haproxy.cfg
22
23 defaults
24     log global
25     mode http
26     option httplog
27     option dontlognull
28     timeout connect 5000
29     timeout client 50000
30     timeout server 50000
31     errorfile 400 /etc/haproxy/errors/400.http
32     errorfile 403 /etc/haproxy/errors/403.http
33     errorfile 408 /etc/haproxy/errors/408.http
34     errorfile 500 /etc/haproxy/errors/500.http
35     errorfile 502 /etc/haproxy/errors/502.http
36     errorfile 503 /etc/haproxy/errors/503.http
37     errorfile 504 /etc/haproxy/errors/504.http
38
39 frontend http_front
40     bind 0.0.0.0:80
41     default_backend http_back
42
43 backend http_back
44     balance roundrobin
45     server web1 192.168.19.121:80 check
46     server web2 192.168.19.122:80 check
47
48
```

Acrescentei a parte do frontend e do backend

© Seção **frontend http\_front**: Configura o ponto de entrada para o tráfego HTTP, escutando na porta 80 e encaminhando solicitações para o backend http\_back.

© Seção **backend http\_back**: Configura dois servidores backend (web1 e web2) para receber o tráfego com balanceamento do tipo **round-robin**, garantindo distribuição equitativa entre os servidores.

A seguir configurei o pacemaker com o crm da mesma forma que anteriormente tinha configurado nos SQLs

No crm:

```
1. primitive p_haproxy systemd:haproxy op monitor interval=30s
2. primitive p_vip ocf:heartbeat:IPaddr2 params ip="172.20.19.1" cidr_netmask="24" op
   monitor interval="10s"
3. colocation col_haproxy_with_vip inf: p_haproxy p_vip
4. order o_haproxy_after_vip inf: p_vip p_haproxy
5. property cib-bootstrap-options: stonith-enabled=false no-quorum-policy=ignore cluster-
   infrastructure=corosync
6.
```

A configuração define dois recursos principais no cluster para alta disponibilidade:

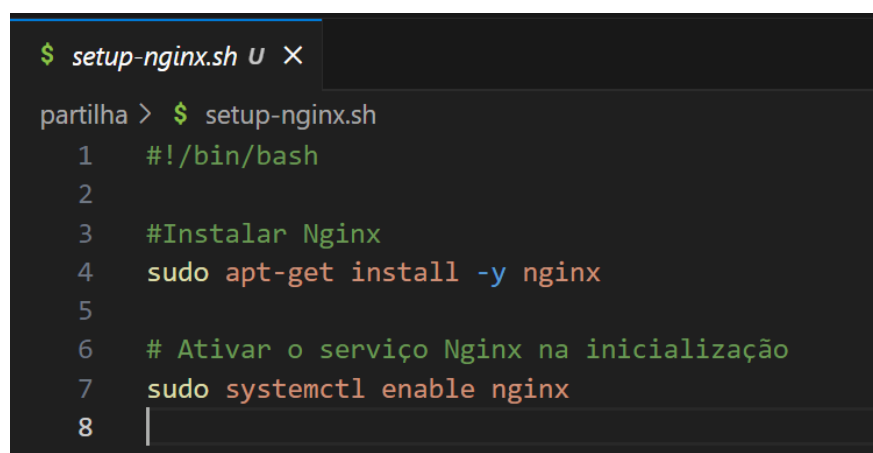
- **p\_haproxy**: Gerencia o serviço HAProxy utilizando o systemd, com monitoramento periódico a cada 30 segundos.
- **p\_vip**: Configura o IP virtual 172.20.19.1 para acesso ao balanceador de carga, monitorando-o constantemente.

As regras de colocação garantem que o HAProxy só funcione em um nó onde o IP virtual esteja ativo, e a ordem especifica que o IP deve ser iniciado antes do HAProxy. A política do cluster desativa STONITH e permite operação mesmo sem quórum.

Configuração dos servidores Web1 e Web2

A configuração dessas máquinas baseou-se em instalar nginx, cacti e ganglia, o nginx funcionará como suporte para o cacti e ganglia, permitindo que o cliente tenha acesso à interface web das mesmas.

O nginx já vem instalado nas máquinas devido ao vagrantfile que executa o script setup-nginx.sh



```
$ setup-nginx.sh U X
partilha > $ setup-nginx.sh
1  #!/bin/bash
2
3  #Instalar Nginx
4  sudo apt-get install -y nginx
5
6  # Ativar o serviço Nginx na inicialização
7  sudo systemctl enable nginx
8  |
```

Primeira coisa que fiz foi mudar a pasta root do nginx para /cluster/www no ficheiro /etc/nginx/sites-available/default

```
server {
    listen 80 default_server;
    |    listen [::]:80 default_server;

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    server_name _;

    location / {
        root /cluster/www;
        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;
    }
}
```

## Instalação do Ganglia

No servidor principal web, que hora pode ser web1 ou web2 fiz a instalação do Ganglia, da sua interface web e também do gmetad, onde vou configurar os servidores a serem monitorados.

A configuração do Gmetad é feita a partir do ficheiro /etc/ganglia/gmetad.conf colocando os ips de cada máquina e também a porta 8649 onde o ganglia vai estar a escuta.

```
# data_source "my cluster" 10 localhost my.machine.edu:8649 1.2.3.5:8655
# data_source "my grid" 50 1.3.4.7:8655 grid.org:8651 grid-backup.org:8651
# data_source "another source" 1.3.4.7:8655 1.3.4.8

data_source "web1" 192.168.19.121:8649
data_source "web2" 192.168.19.122:8649
data_source "sql1" 192.168.19.111:8649
data_source "sql2" 192.168.19.112:8649
#
```

Os servidores a serem monitorados são os WEBS e os SQLs

Depois de configurar fazer o restart do gmetad

Nos servidores sql1 e sql2 também precisamos instalar o ganglia

Agora é configurar o nginx para servir o ganglia

No ficheiro /etc/nginx/sites-available/default vamos adicionar um location específico para acessar a interface web do Ganglia

```
#Ganglia Location

location /ganglia {
    alias /usr/share/ganglia-webfrontend; # O diretório onde os arquivos do Ganglia estão$
    index index.php;
}

location ~ /\.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass unix:/var/run/php/php7.2-fpm.sock; # Ajuste de acordo com sua versão do $
    fastcgi_param SCRIPT_FILENAME $request_filename;
    include fastcgi_params;
}

location ~ /\.ht {
    deny all;
}

# pass PHP scripts to FastCGI server
#
```

Fiz o restart do nginx, com isso podemos acessar a interface web do ganglia, utilizando:

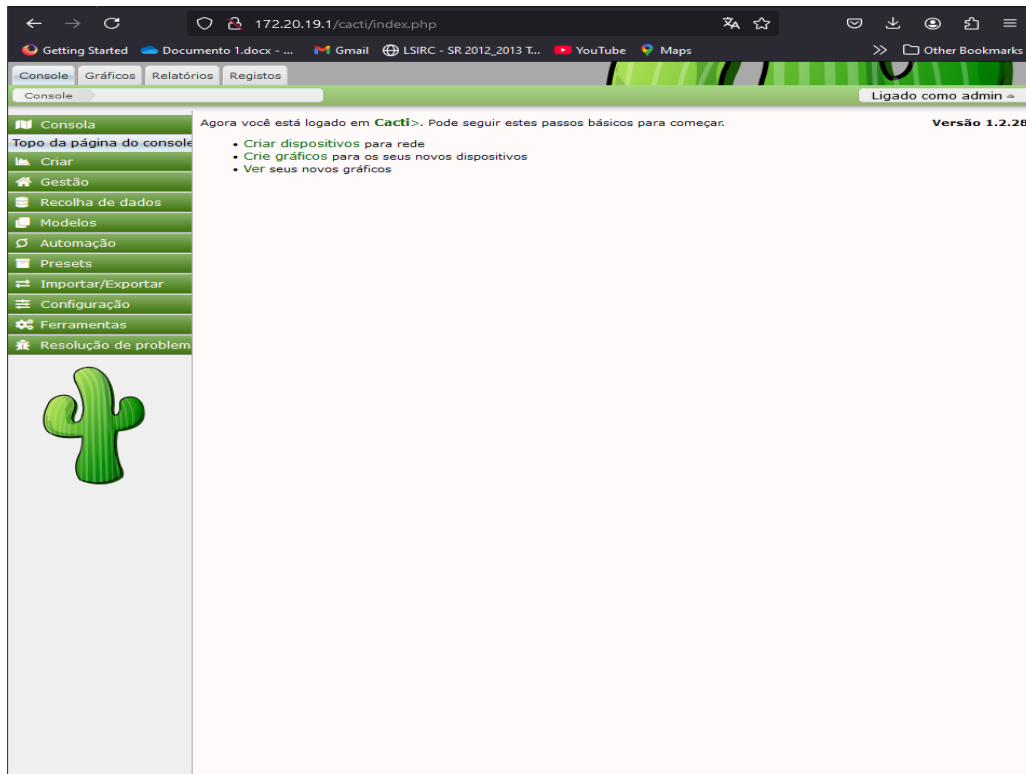
<http://192.168.19.121/ganglia> Web1

<http://192.168.19.122/ganglia> Web2

<http://172.20.19.1/ganglia> VirtualIp dos proxies'

## Instalação do Cacti

Para instalar o Cacti segui uma guia de instalação encontrada na internet, disponibilizo mais abaixo no anexo o link do site



## Avaliação do Cluster

Ferramenta de teste utilizado siege

Testes com 2 máquinas Web

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 30 -t 1m http://172.20.19.1/ganglia
{
  "transactions": 7177,
  "availability": 100.00,
  "elapsed_time": 59.88,
  "data_transferred": 242.10,
  "response_time": 0.25,
  "transaction_rate": 119.86,
  "throughput": 4.04,
  "concurrency": 29.90,
  "successful_transactions": 7178,
  "failed_transactions": 0,
  "longest_transaction": 2.46,
  "shortest_transaction": 0.01
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 60 -t 1m http://172.20.19.1/ganglia
```

```
{
  "transactions": 7119,
  "availability": 100.00,
  "elapsed_time": 59.86,
  "data_transferred": 241.44,
  "response_time": 0.50,
  "transaction_rate": 118.93,
  "throughput": 4.03,
  "concurrency": 59.59,
  "successful_transactions": 7122,
  "failed_transactions": 0,
  "longest_transaction": 5.72,
  "shortest_transaction": 0.01
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 120 -t 1m http://172.20.19.1/ganglia
```

```
{
  "transactions": 7637,
  "availability": 100.00,
  "elapsed_time": 59.73,
  "data_transferred": 264.76,
  "response_time": 0.93,
  "transaction_rate": 127.86,
  "throughput": 4.43,
  "concurrency": 118.84,
  "successful_transactions": 7640,
  "failed_transactions": 0,
  "longest_transaction": 7.09,
  "shortest_transaction": 0.00
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 240 -t 1m http://172.20.19.1/ganglia
```

```
{
  "transactions": 8456,
  "availability": 100.00,
  "elapsed_time": 59.19,
  "data_transferred": 291.14,
  "response_time": 1.64,
  "transaction_rate": 142.86,
  "throughput": 4.92,
  "concurrency": 234.01,
  "successful_transactions": 8507,
  "failed_transactions": 0,
  "longest_transaction": 18.72,
  "shortest_transaction": 0.04
}
```

## Testes com 1 máquina Web

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 30 -t 1m http://172.20.19.1/ganglia
```

```
{
  "transactions": 6984,
  "availability": 100.00,
  "elapsed_time": 59.93,
  "data_transferred": 234.99,
  "response_time": 0.26,
  "transaction_rate": 116.54,
  "throughput": 3.92,
  "concurrency": 29.84,
  "successful_transactions": 6990,
  "failed_transactions": 0,
  "longest_transaction": 6.46,
  "shortest_transaction": 0.00
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 60 -t 1m http://172.20.19.1/ganglia
{
  "transactions": 7014,
  "availability": 100.00,
  "elapsed_time": 59.58,
  "data_transferred": 235.90,
  "response_time": 0.49,
  "transaction_rate": 117.72,
  "throughput": 3.96,
  "concurrency": 57.35,
  "successful_transactions": 7047,
  "failed_transactions": 0,
  "longest_transaction": 9.72,
  "shortest_transaction": 0.00
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 120 -t 1m http://172.20.19.1/ganglia
{
  "transactions": 6550,
  "availability": 100.00,
  "elapsed_time": 59.43,
  "data_transferred": 219.56,
  "response_time": 0.87,
  "transaction_rate": 110.21,
  "throughput": 3.69,
  "concurrency": 95.54,
  "successful_transactions": 6670,
  "failed_transactions": 0,
  "longest_transaction": 26.75,
  "shortest_transaction": 0.00
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 240 -t 1m http://172.20.19.1/ganglia
{
  "transactions": 2475,
  "availability": 68.01,
  "elapsed_time": 35.31,
  "data_transferred": 83.37,
  "response_time": 1.58,
  "transaction_rate": 70.09,
  "throughput": 2.36,
  "concurrency": 111.06,
  "successful_transactions": 3639,
  "failed_transactions": 1164,
  "longest_transaction": 26.52,
  "shortest_transaction": 0.12
}
```

Os quadros com os resultados para os dois testes estão disponíveis no ficheiro excel que disponibilizei na pasta do projeto (Testes)

### Quadro 1: Testes com Duas Máquinas Web

- **Descrição dos testes:**  
Foram simulados acessos a um site com **duas máquinas web a trabalhar em conjunto**, dividindo o trabalho para atender as solicitações.
- **Análise dos resultados:**
  - **Transações:** Com 30 conexões, realizaram-se 7.177 transações. À medida que o número de conexões aumentou, o número de transações também cresceu, atingindo 8.456 com 240 conexões.
  - **Disponibilidade:** Mantém-se a **100% em todos os casos**, o que significa que todas as solicitações foram atendidas sem falhas.



- **Tempo de resposta:** Foi muito rápido para cargas leves (0,25 segundos com 30 conexões). Com 240 conexões, o tempo aumentou para 1,64 segundos, mas manteve-se aceitável.
- **Throughput:** A quantidade de dados transferidos por segundo aumentou com as conexões, alcançando **4,92 MB/s** no cenário de maior carga.
- **Conclusão:** As duas máquinas web geriram bem tanto cargas leves como pesadas, oferecendo um desempenho estável, com tempos de resposta rápidos e sem falhas.

## Quadro 2: Testes com Uma Máquina Web

- **Descrição dos testes:**  
Aqui, os testes foram realizados com apenas **uma máquina web ativa**, que teve de lidar com todas as solicitações sozinha.
- **Análise dos resultados:**
  - **Transações:** Para 30 e 60 conexões, o número de transações foi semelhante ao teste com duas máquinas. No entanto, com 240 conexões, houve uma descida abrupta para **2.475 transações**, mostrando que a máquina estava sobrecarregada.
  - **Disponibilidade:** Foi de **100%** para até 120 conexões, mas caiu para **68,01%** com 240 conexões, indicando que quase 32% das solicitações falharam.
  - **Tempo de resposta:** Para cargas baixas (30 conexões), o tempo de resposta foi de 0,26 segundos. Com 240 conexões, subiu para 1,58 segundos, com algumas transações a demorar mais de 26 segundos, o que é muito lento.
  - **Falhas:** O sistema apresentou **1.164 falhas** com 240 conexões, refletindo as limitações da máquina em lidar com cargas elevadas.
  - **Throughput:** A transferência de dados diminuiu para **2,36 MB/s** com 240 conexões, muito abaixo do desempenho com duas máquinas.
- **Conclusão:** Uma máquina web funciona bem para cargas leves, mas demonstra limitações claras ao suportar grandes volumes de acessos simultâneos, resultando em falhas e tempos de resposta elevados.

Esses testes foram feitos para o site da ganglia que basicamente esforça mais os servidores web, agora para testar os SQLs vou testar o site do cacti. o Cacti processa requisições dos clientes e, para responder, interage frequentemente com o MariaDB nos servidores SQL. Isso significa que o desempenho do site depende não apenas dos servidores web, mas também da eficiência da base de dados.

### Testes com 2 máquinas Web

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 30 -t 1m http://172.20.19.1/cacti
```

```
{
  "transactions":          3058,
  "availability":          100.00,
  "elapsed_time":          60.86,
  "data_transferred":      288.91,
  "response_time":         0.57,
  "transaction_rate":       50.25,
  "throughput":             4.75,
  "concurrency":            28.80,
  "successful_transactions": 3074,
  "failed_transactions":    0,
  "longest_transaction":    18.49,
  "shortest_transaction":   0.00
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 60 -t 1m http://172.20.19.1/cacti
```

```
{
  "transactions":          2854,
  "availability":          100.00,
  "elapsed_time":          59.19,
  "data_transferred":      285.16,
  "response_time":         1.22,
  "transaction_rate":       48.22,
  "throughput":             4.82,
  "concurrency":            58.68,
  "successful_transactions": 2854,
  "failed_transactions":    0,
  "longest_transaction":    31.89,
  "shortest_transaction":   0.01
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 120 -t 1m http://172.20.19.1/cacti
```

```
{
  "transactions":          2913,
  "availability":          100.00,
  "elapsed_time":          59.65,
  "data_transferred":      281.87,
  "response_time":         2.35,
  "transaction_rate":       48.83,
  "throughput":             4.73,
  "concurrency":            114.88,
  "successful_transactions": 2913,
  "failed_transactions":    0,
  "longest_transaction":    46.71,
  "shortest_transaction":   0.01
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 240 -t 1m http://172.20.19.1/cacti
```

```
{
  "transactions":          2722,
  "availability":          96.39,
  "elapsed_time":          59.28,
  "data_transferred":      260.63,
  "response_time":         4.76,
  "transaction_rate":       45.92,
  "throughput":             4.40,
  "concurrency":            218.35,
  "successful_transactions": 2944,
  "failed_transactions":    102,
  "longest_transaction":    51.26,
  "shortest_transaction":   0.01
}
```

## Testes com 1 máquina Web

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 30 -t 1m http://172.20.19.1/cacti
```

```
{
  "transactions": 2705,
  "availability": 100.00,
  "elapsed_time": 59.39,
  "data_transferred": 254.22,
  "response_time": 0.62,
  "transaction_rate": 45.55,
  "throughput": 4.28,
  "concurrency": 28.39,
  "successful_transactions": 2718,
  "failed_transactions": 0,
  "longest_transaction": 25.17,
  "shortest_transaction": 0.01
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 60 -t 1m http://172.20.19.1/cacti
```

```
{
  "transactions": 2678,
  "availability": 100.00,
  "elapsed_time": 60.01,
  "data_transferred": 259.27,
  "response_time": 1.30,
  "transaction_rate": 44.63,
  "throughput": 4.32,
  "concurrency": 57.97,
  "successful_transactions": 2688,
  "failed_transactions": 0,
  "longest_transaction": 40.12,
  "shortest_transaction": 0.02
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 120 -t 1m http://172.20.19.1/cacti
```

```
{
  "transactions": 3287,
  "availability": 98.44,
  "elapsed_time": 59.85,
  "data_transferred": 311.00,
  "response_time": 1.62,
  "transaction_rate": 54.92,
  "throughput": 5.20,
  "concurrency": 88.95,
  "successful_transactions": 3439,
  "failed_transactions": 52,
  "longest_transaction": 50.97,
  "shortest_transaction": 0.00
}
```

```
fabio@LAPTOP-2MBDDPHD:/mnt/c/Users/fabio$ siege -c 240 -t 1m http://172.20.19.1/cacti
```

```
{
  "transactions": 1106,
  "availability": 48.19,
  "elapsed_time": 60.00,
  "data_transferred": 106.11,
  "response_time": 7.23,
  "transaction_rate": 18.43,
  "throughput": 1.77,
  "concurrency": 133.19,
  "successful_transactions": 2343,
  "failed_transactions": 1189,
  "longest_transaction": 51.21,
  "shortest_transaction": 0.00
}
```

## Quadro 1 Cacti

### Análise Detalhada

#### 1. Número de Transações e Disponibilidade

- O número de transações bem-sucedidas diminui ligeiramente com o aumento do número de conexões (de 3.058 com 30 conexões para 2.722 com 240 conexões).
- A disponibilidade é **100%** até 120 conexões, mas cai para **96,39%** com 240 conexões, indicando que algumas solicitações falharam sob cargas mais altas (102 falhas).

#### 2. Tempo de Resposta

- O **tempo médio de resposta** cresce à medida que o número de conexões aumenta:
  - 0,57 segundos com 30 conexões.
  - 4,76 segundos com 240 conexões.
- Em cenários de alta carga (240 conexões), o tempo de resposta mais longo foi de 51,26 segundos, o que indica que o sistema levou muito tempo para processar algumas requisições.

#### 3. Taxa de Transações e Throughput

- A **taxa de transações por segundo** (quantas transações são concluídas por segundo) mantém-se estável até 120 conexões (cerca de 48 transações/segundo). No entanto, com 240 conexões, a taxa diminui ligeiramente para **45,92 transações/segundo**.
- O **throughput** (quantidade de dados transferidos por segundo) permanece próximo de 4,75 MB/s até 120 conexões, mas cai para **4,40 MB/s** com 240 conexões, mostrando que o sistema está a atingir os seus limites.

#### 4. Confiabilidade e Falhas

- Até 120 conexões, não houve falhas. Com 240 conexões, ocorreram **102 falhas**, representando 3,61% das transações totais, o que reflete dificuldades do sistema em lidar com a carga.

## Análise Detalhada

### 1. Desempenho em Baixa Carga (30 e 60 conexões)

- **Disponibilidade:** A disponibilidade foi de **100%**, ou seja, nenhuma requisição falhou.
- **Tempo de Resposta:** Os tempos de resposta foram razoáveis, com 0.62 segundos (30 conexões) e 1.30 segundos (60 conexões).
- **Transações e Throughput:** O número de transações processadas foi consistente (cerca de 2.700), e o throughput manteve-se em cerca de 4.3 MB/s.
- **Conclusão:** O servidor conseguiu lidar bem com cargas baixas.

### 2. Desempenho em Carga Média (120 conexões)

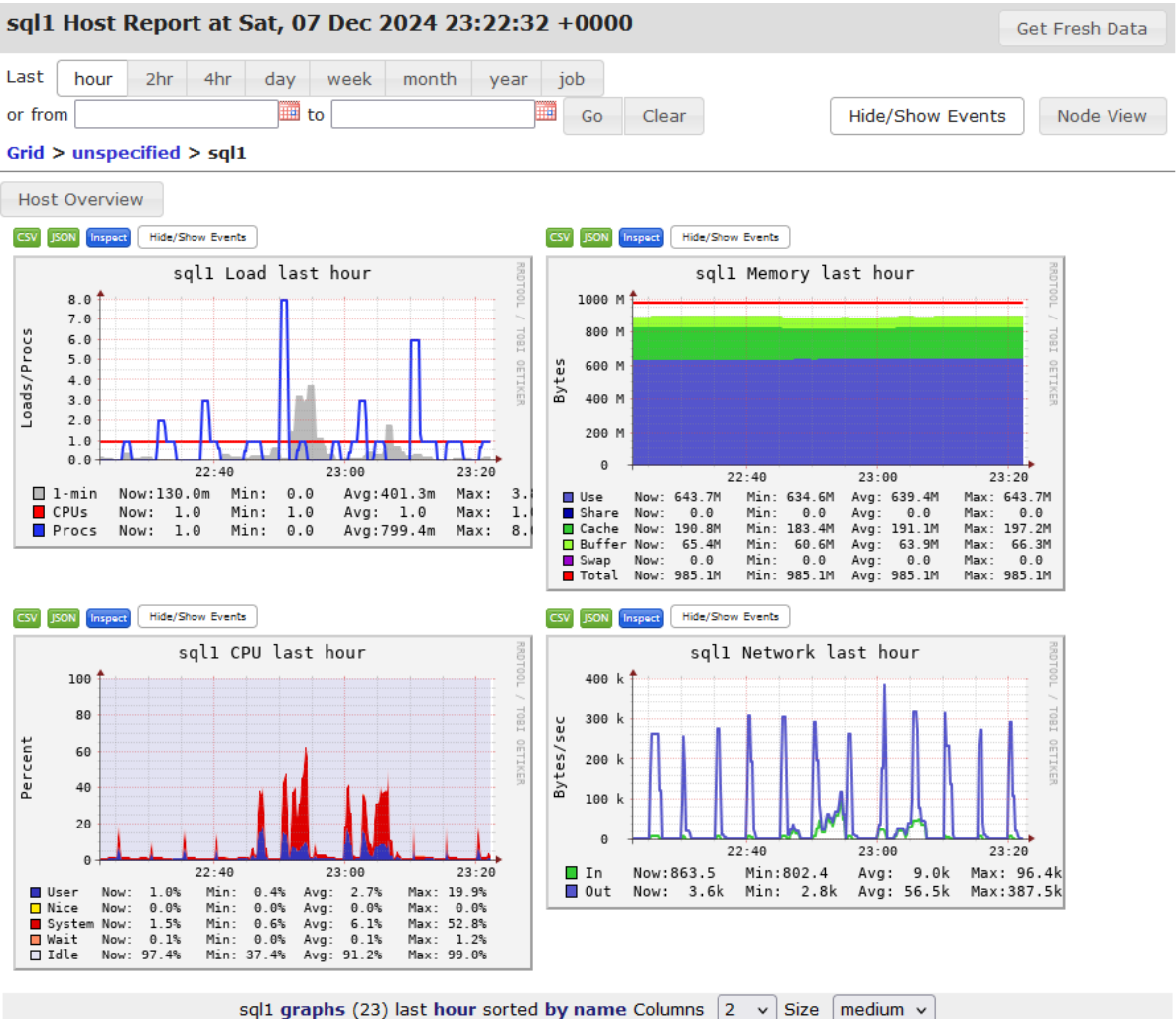
- **Disponibilidade:** A disponibilidade caiu ligeiramente para **98.44%**, com 52 transações falhadas.
- **Tempo de Resposta:** O tempo médio de resposta aumentou para **1.62 segundos**, e algumas transações demoraram até **50.97 segundos**, indicando que o sistema começou a sentir a pressão da carga.
- **Transações e Throughput:** Apesar das falhas, o número de transações aumentou para 3.287, e o throughput cresceu para 5.20 MB/s.
- **Conclusão:** O sistema conseguiu lidar com a carga, mas sinais de sobrecarga começaram a aparecer.

### 3. Desempenho em Alta Carga (240 conexões)

- **Disponibilidade:** A disponibilidade caiu drasticamente para **48.19%**, com **1.189 transações falhadas**, mostrando que o servidor estava significativamente sobrecarregado.
- **Tempo de Resposta:** O tempo médio de resposta subiu para **7.23 segundos**, com algumas transações a demorar até **51.21 segundos**, o que é um desempenho muito lento.
- **Transações e Throughput:** Apenas 1.106 transações foram concluídas com sucesso, e o throughput caiu para **1.77 MB/s**, demonstrando que o servidor já não conseguia lidar com a carga.
- **Conclusão:** O servidor não foi capaz de suportar esta carga elevada, resultando em falhas significativas e tempos de resposta muito altos.

## Criação dos Gráficos no Ganglia

Para criar os gráficos no Ganglia acedi ao separador de Agregar Gráficos para conseguir comparar entre as diferentes máquinas.



## Servidor SQL1

### 1. Load (Carga):

- A carga média foi de 401.3%, indicando alta utilização do servidor.
- Picos chegaram a 1300%, o que sugere momentos de uso muito intenso, provavelmente devido a múltiplas consultas simultâneas.

### 2. Memory (Memória):

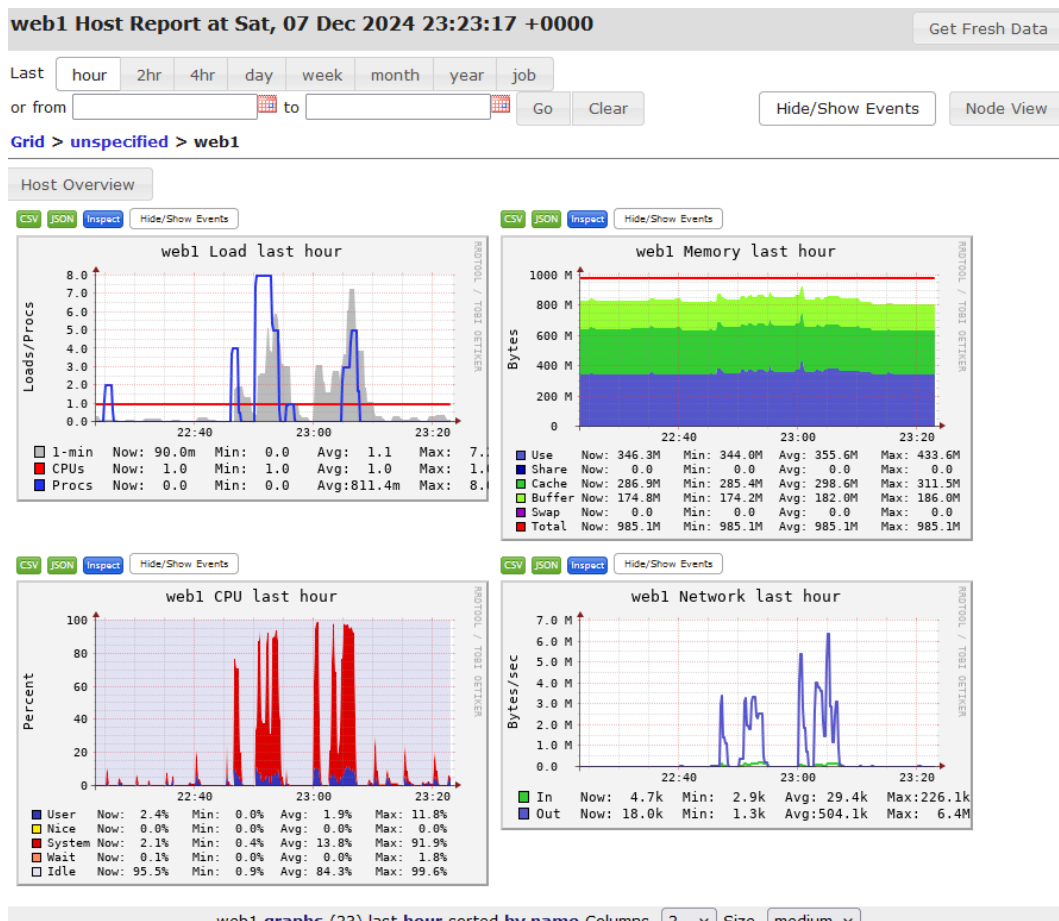
- Uso consistente da memória, com aproximadamente 643.7 MB ocupados.
- Buffer e cache utilizam uma porção significativa, o que é esperado para otimização de acessos a dados em bases de dados.

### 3. CPU (Processador):

- Uso moderado da CPU, com o processador maioritariamente ocioso (97.4% de idle).
- Indica que a carga elevada foi mais devida a acessos ao disco ou rede, e não processamento.

### 4. Network (Rede):

- Atividade moderada de rede com um pico de 387.5 KB/s de entrada, mostrando transferência de dados regular.
- Indica consultas ou interações frequentes entre o servidor SQL e outros sistemas.



## Servidor WEB1

### 1. Load (Carga):

- A carga média foi de 811.4%, com picos de até 7000%, indicando alta demanda, especialmente em momentos específicos.
- Estes picos podem refletir picos de acesso simultâneo por usuários ou chamadas ao backend.

### 2. Memory (Memória):

- Uso consistente de memória, com 355 MB ocupados em média.
- O servidor mantém cerca de 300 MB de cache e buffers, o que é ideal para otimizar desempenho.

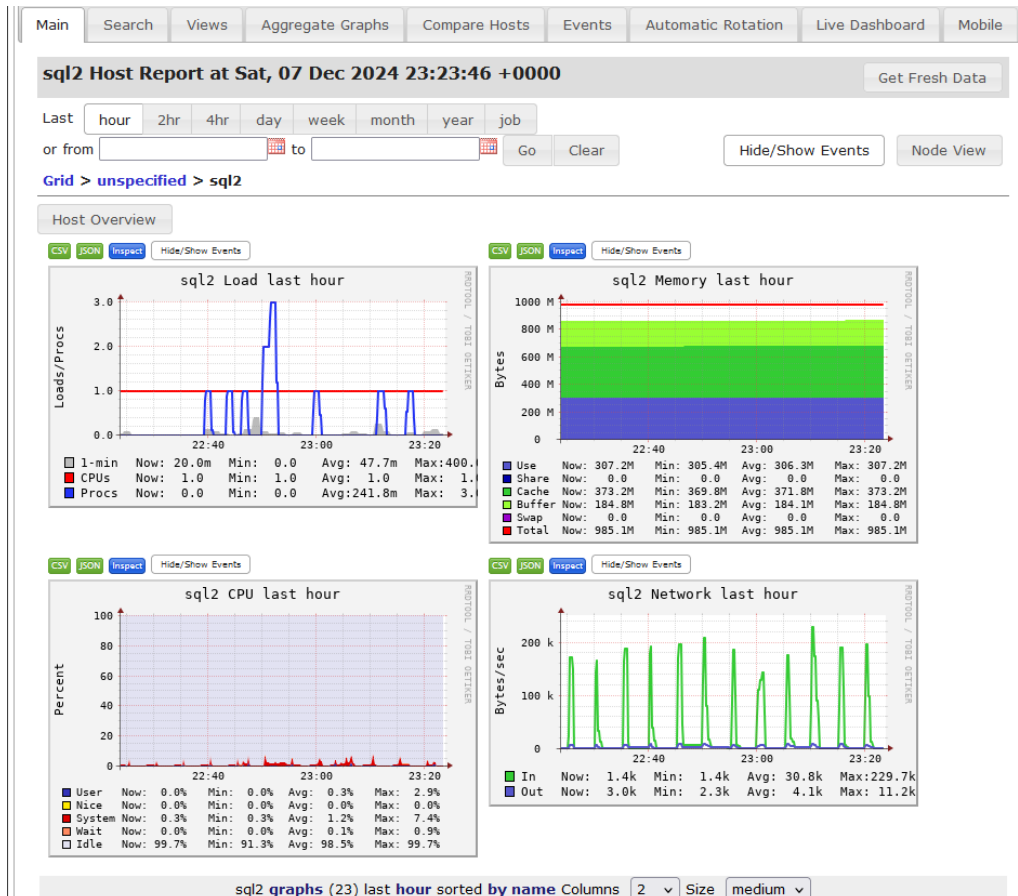
### 3. CPU (Processador):

- Uso intenso da CPU, com períodos de 100% de utilização.
- O sistema passa muito tempo processando requests de utilizadores, com pouca ociosidade.



#### 4. Network (Rede):

- Altíssima atividade de rede, com picos de 6.4 MB/s de saída, indicando a entrega de grandes volumes de dados, como imagens ou páginas completas.



### Servidor SQL2

#### 1. Load (Carga):

- Carga significativamente menor que SQL1, com uma média de 47.7% e picos de 400%.
- Indica que SQL2 está menos sobrecarregado, possivelmente devido a uma menor quantidade de consultas simultâneas.

#### 2. Memory (Memória):

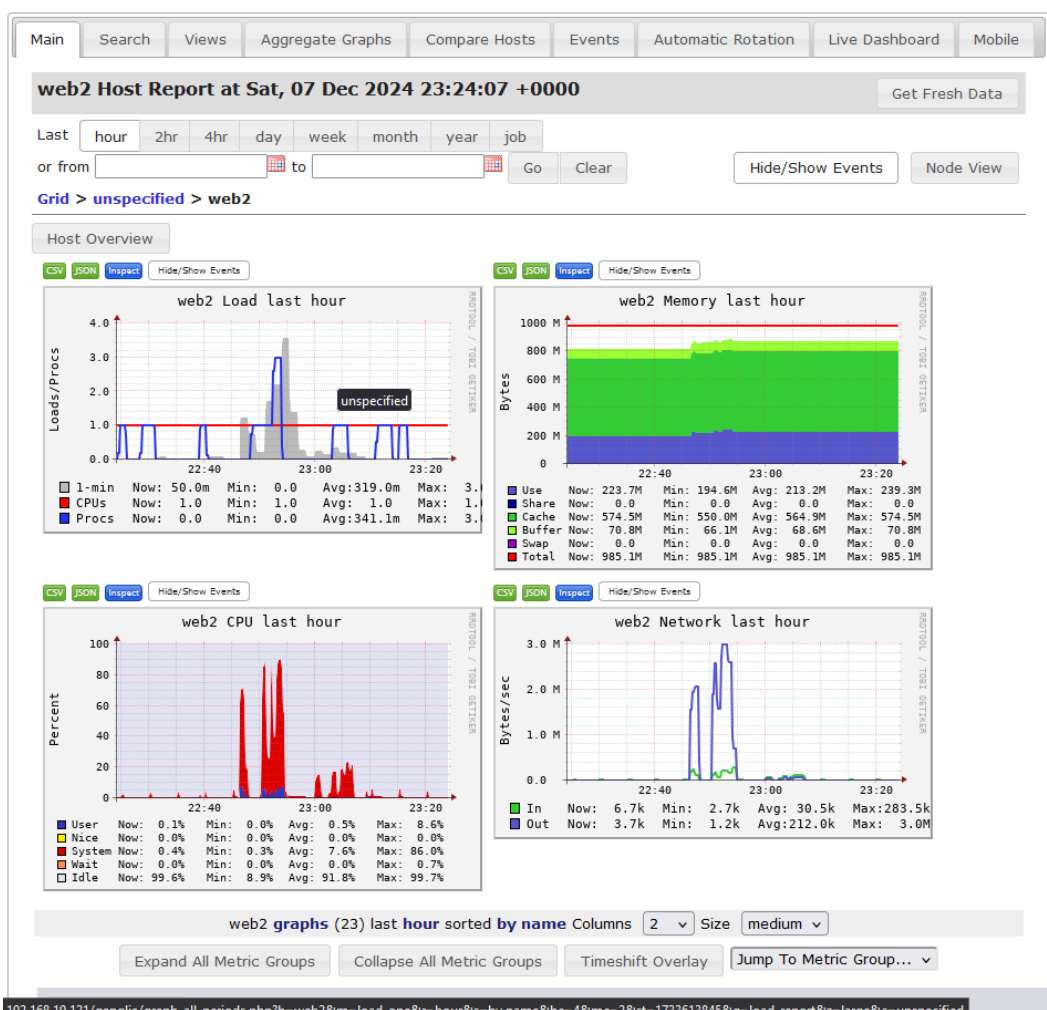
- Uso de memória semelhante ao SQL1, com 807.2 MB utilizados.
- Buffer e cache ocupam grande parte, sinal de bom aproveitamento de memória para melhorar desempenho.

### 3. CPU (Processador):

- CPU está praticamente ociosa (99.7% de idle), mostrando que o sistema não está a processar muitas operações complexas.

### 4. Network (Rede):

- Atividade de rede baixa, com picos de 229 KB/s de entrada.
- Reflete um número menor de interações com o servidor comparado ao SQL1.



## Servidor WEB2

### 1. Load (Carga):

- Carga significativamente baixa, com uma média de 0.30 e picos em torno de 2.0.

- Isso indica que o servidor está funcionando de maneira eficiente, com períodos curtos de maior utilização. A carga é bem administrada e parece estar longe de atingir qualquer limite crítico.

## 2. Memory (Memória):

- O uso de memória é elevado, com a maior parte da RAM ocupada, mas sem utilização de **swap**, o que demonstra que o sistema está gerenciando bem os recursos.
- A presença de buffers e cache ocupa boa parte da memória, o que é ideal para otimizar o desempenho.

## 3. CPU (Processador):

- A CPU está ociosa na maior parte do tempo, com **picos de utilização intensa** durante períodos curtos.
- Esses picos sugerem execução de tarefas específicas ou aumento temporário da carga de trabalho.

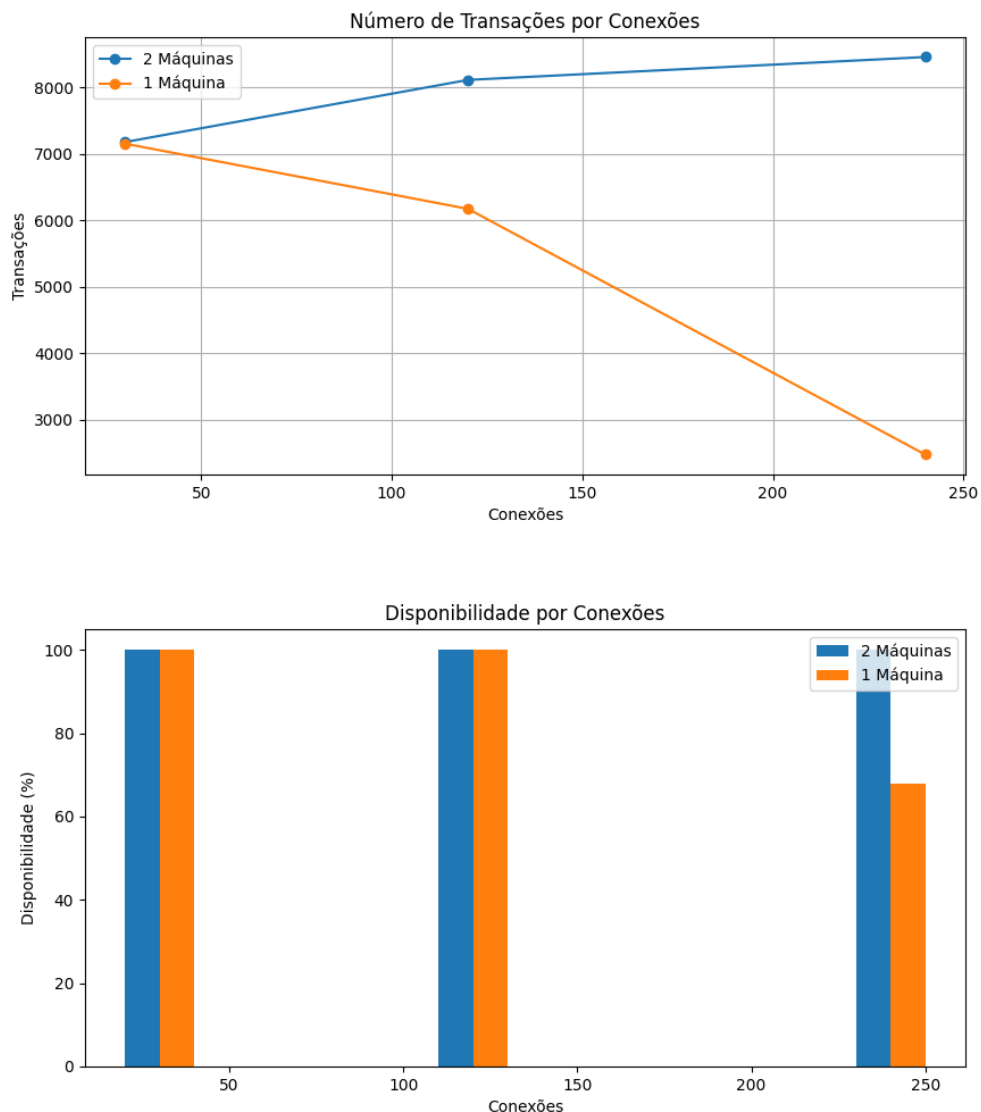
## 4. Network (Rede):

- A atividade de rede é baixa em geral, com picos de até **238.5 KB/s** de saída e **229 KB/s** de entrada.
- Esses valores indicam interações moderadas, provavelmente relacionadas a requisições pontuais, transferências de dados ou backups esporádicos.

Os servidores Web2 e SQL2 apresentam resultados mais moderados por não serem os nós principais isto tudo devido ao balanceamento de carga nos servidores Web e o modo ativo/passivo nos servidores SQL, ao contrário dos resultados nos servidores Web1 e Sql1 que são os nós principais e responsáveis pelas respostas às requisições feitas apartir do cliente.

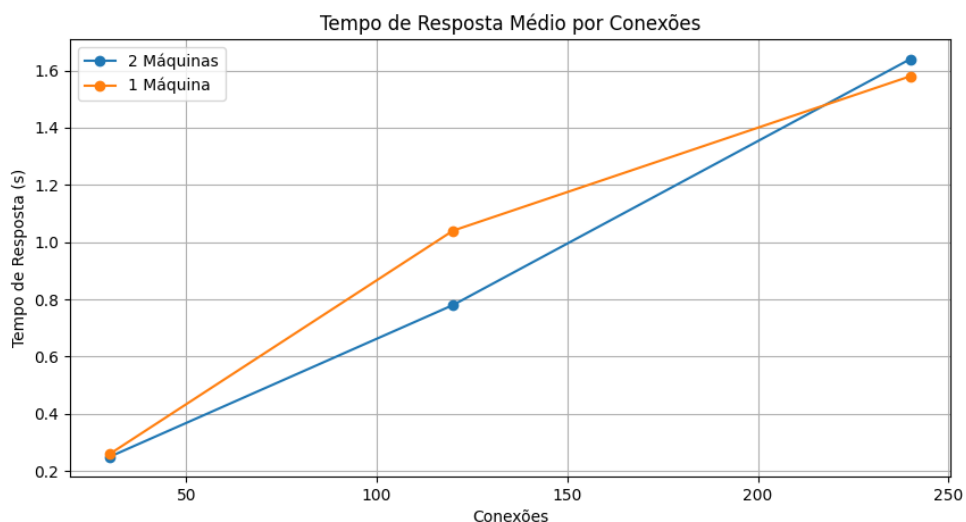
## Análise Qualitativa dos resultados

### Número de Transações e Disponibilidade



- **Duas Máquinas:** O número de transações aumenta proporcionalmente ao número de conexões, mantendo 100% de disponibilidade. Isso evidencia uma boa distribuição de carga e alta resiliência.
- **Uma Máquina:** A disponibilidade começa a diminuir drasticamente com 240 conexões, com uma redução significativa no número de transações bem-sucedidas (68,01% de disponibilidade e 2.475 transações). Este comportamento reflete a limitação de recursos em cenários de alta carga.

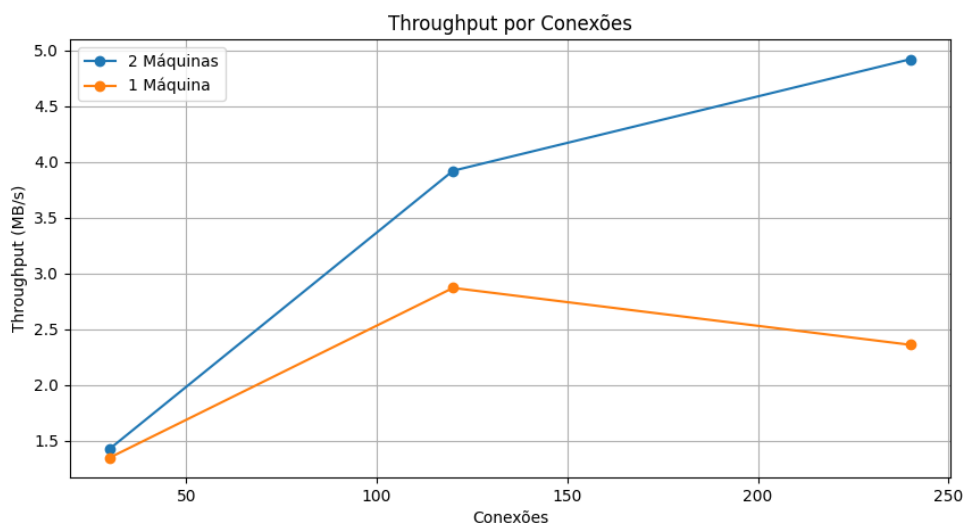
## Tempo de Resposta



© Duas **Máquinas**: Os tempos de resposta aumentam moderadamente com o crescimento da carga, mas permanecem abaixo de 2 segundos mesmo em condições extremas.

© Uma **Máquina**: Em cargas altas, os tempos de resposta são muito maiores e chegam a valores inaceitáveis (26,45 segundos no máximo), demonstrando uma sobrecarga crítica.

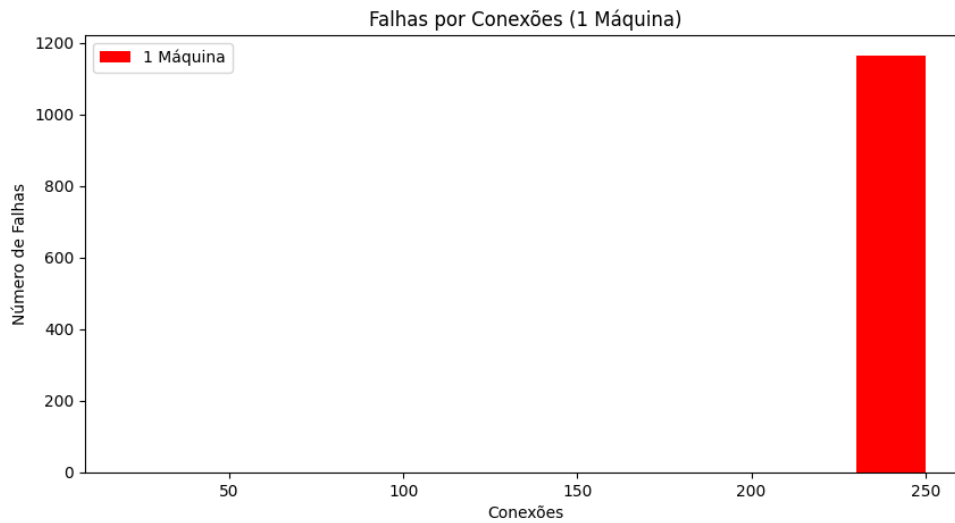
## Throughput



© Duas **Máquinas**: O throughput cresce proporcionalmente ao número de conexões, alcançando 4,92 MB/s no pico, mostrando eficiência no uso de banda.

© Uma **Máquina**: O throughput atinge um limite inferior (2,36 MB/s) em altas cargas, reforçando que a máquina não consegue lidar adequadamente com muitas solicitações simultâneas.

## Falhas



© Duas **Máquinas**: Nenhuma falha foi registada, mesmo com 240 conexões.

© Uma **Máquina**: O número de falhas cresce significativamente com 240 conexões (1.164 falhas), evidenciando a incapacidade do sistema de atender às solicitações.

A análise revela que a arquitetura com duas máquinas web é significativamente mais robusta, conseguindo lidar com cargas altas sem comprometer a disponibilidade, tempo de resposta ou throughput. Já a configuração com apenas uma máquina apresenta limitações evidentes em cenários de alta carga, resultando em falhas, tempos de resposta elevados e throughput reduzido.

## Anexo

Site do guia de instalação do cacti

<https://linux.how2shout.com/how-to-install-cacti-monitoring-on-ubuntu-22-04-20-04/>

Ficheiros de configuração configuradas durante o trabalho que estão na pasta do projeto

Proxy1 e Proxy2

Haproxy → haproxy

Corosync → proxy\_corosync

Hosts → hosts\_proxy

Web1 e Web2

Ganglia → ganglia

Nginx → nginx

Sql1 e Sql2

MariaDB → mariadb\_sql

Corosync → sql\_corosync

Hosts → hosts\_sql

