



Universidade do Minho

Mestrado Integrado em Engenharia Informática

Infraestruturas de Centros de Dados

Escalabilidade e Telemetria de Infraestruturas

Graphite, Grafana e collectd

Trabalho Prático

Hugo Oliveira (A78565)

João Vieira (A78468)

Raphael Oliveira (A78848)

28 de Janeiro de 2019

Resumo

No contexto da unidade curricular de *Infraestruturas de Centros de Dados* pertencente ao perfil de *Engenharia de Aplicações*, é realizado um projeto onde se pretende planear e implementar um serviço escalável de telemetria de infraestruturas através da utilização de ferramentas como o **Graphite**, **Grafana** e **collectd**. Todo o processo de implementação, envolve o provisionamento de máquinas virtuais e *deployment* de serviços que permitem a escalabilidade da infraestrutura e a disponibilidade dos mais variados serviços, com vista à inexistência de pontos de falha únicos.

A utilização destas ferramentas permitirá, a um administrador de sistemas, ter uma visão macro e microscópica do estado de cada uma das máquinas da infraestrutura e dos serviços por ela disponibilizada.

Ao longo deste relatório, serão analisados os componentes da arquitetura e ainda as diferentes relações e conexões entre eles. Por fim, será abordada toda a implementação da infraestrutura.

Conteúdo

1	Introdução	1
2	Análise dos componentes da infraestrutura	2
3	Estrutura adotada	3
4	Implementação do sistema	5
4.1	Sincronização do tempo nas máquinas	6
4.2	Armazenamento e replicação dos dados	6
4.2.1	Instalação e configuração dos <i>DRBD</i>	7
4.2.2	Criação de um <i>cluster</i>	8
4.2.3	Adição e configuração de recursos do <i>cluster</i>	9
4.3	Sistema de Gestão base de dados	11
4.3.1	Instalação e configuração do <i>PostgreSQL</i>	11
4.3.2	Adição e configuração de recursos do <i>cluster</i>	12
4.4	Sistema de balanceamento de carga do serviço <i>web</i>	14
4.4.1	Instalação e configuração do <i>HAproxy</i>	14
4.4.2	Adição e configuração de recursos do <i>cluster</i>	15
4.5	<i>Graphite</i>	16
4.5.1	Instalação e configuração do <i>Graphite</i>	16
4.5.2	Adição e configuração de recursos do <i>cluster</i>	19
4.6	<i>Grafana</i>	20
4.6.1	Instalação e configuração do <i>Grafana</i>	21
5	Escalabilidade e disponibilidade da infraestrutura	23
6	Análise do sistema	25
6.1	Exemplos de <i>Dashboards</i> no <i>Grafana</i>	27
7	Conclusão	30
8	Bibliografia	31

Lista de Figuras

1	Arquitetura da infraestrutura.	4
2	Processo de sincronização entre discos.	8
3	Ambas as máquinas sincronizadas.	8
4	Estado do <i>cluster</i> num dado momento.	10
5	Estado do <i>cluster</i> num dado momento.	11
6	<i>Dashboard</i> para monitorização da utilização de <i>CPU</i>	28
7	<i>Dashboard</i> para monitorização da utilização de <i>RAM</i>	28
8	<i>Dashboard</i> para monitorização da utilização do <i>haproxycluster</i>	29
9	<i>Dashboard</i> para monitorização do disco <i>/dev/drbd1</i> nos <i>DRBD's</i>	29

1 Introdução

Numa infraestrutura que se pretenda a alta disponibilidade de um serviço, é necessário o estudo e a análise do mesmo, de modo a eliminarmos pontos de falha únicos que possam negar o serviço aos utilizadores e, ainda, permitir escalar o serviço em caso de alteração da carga do sistema.

O **Graphite** é uma ferramenta de monitorização de máquinas e serviços, ideal num contexto empresarial. Pensada e implementada de forma a funcionar corretamente em máquinas físicas de uma infraestrutura e ou em máquinas providenciadas na *cloud*. Esta aplicação tende a ser usada por várias equipas de desenvolvimento de *software*, como forma de medir a utilização de recursos e medir a *performance* das aplicações, *websites*, serviços em rede, etc. Esta ferramenta é constituída por vários componentes, como: **carbon**, **whisper** e **graphite-web**, descritos com pormenor mais à frente. Ferramentas como o **collectd** e o **StatsD** podem ser utilizados para recolher dados estatísticos de um sistema, como percentagem de uso do *cpu*, *ram* e disco e, ainda, informações de serviços que estejam a correr no mesmo.

O **Grafana** é uma aplicação *web*, que possibilita a criação de *dashboards* e análise de métricas para o sistema de monitorização. No contexto deste projecto, é utilizado para suporte e visualização dos dados fornecidos pelo **Graphite**, funcionando como o *frontend* principal deste serviço de telemetria da infraestrutura.

Para descrever todo o processo de desenvolvimento do sistema, irá ser analisado o problema em questão e, ainda, todos os seus componentes. De seguida, é necessário descrever o processo de implementação da infraestrutura, de forma a fornecer um serviço escalável, não possuindo pontos únicos de falhas e, por fim, permitir a telemetria de infraestruturas computacionais, que dada a exiguidade de recursos existentes, serão utilizadas as próprias máquinas da infraestrutura para a respetiva monitorização.

2 Análise dos componentes da infraestrutura

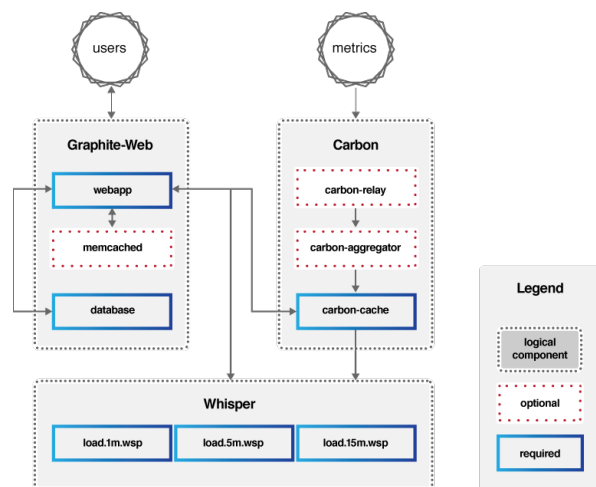
Tendo por base a criação de uma estrutura com o objetivo de monitorizar as máquinas pertencentes à mesma, com posterior apresentação dos resultados através de gráficos, foram estudadas várias ferramentas que permitem a resolução deste problema.

Por forma a recolher os dados de cada máquina, foi decidido, a utilização da ferramenta `collectd`, que desta forma será responsável por recolher as métricas escolhidas em cada uma das máquinas na qual se encontra instalado.

É utilizada a aplicação **Graphite** tendo em conta as características que a mesma apresenta: armazenar dados numéricos (métricas escolhidas) de séries temporais, efetuar renderização de gráficos com estes mesmos dados.

Esta aplicação é constituída por três componentes:

- **carbon** - serviço de alto desempenho que recolhe dados de séries temporais;
- **whisper** - biblioteca simples de base de dados para armazenar dados de séries temporais;
- **graphite-web** - interface *web* do **Graphite** que permite a renderização de gráficos com as métricas fornecidas.



Apesar do **Graphite** fornecer uma interface *web*, foi decidido utilizarmos uma interface melhorada e mais apelativa ao utilizador, que permite não só utilizar o **graphite-web** (ou **graphite-api**) como o **Elasticsearch** (por exemplo), para recolha de dados. Esta interface é fornecida pelo **Grafana**, que possibilita a criação de *dashboards* e a monitorização e análise de métricas das diferentes instâncias que, no contexto do trabalho, funcionará como o *frontend* principal do mesmo.

É ainda importante referir que, todos os dados associados à autenticação de utilizadores e gestão de *dashboards*, são tratados pelo Sistema de Gestão de Base de Dados **PostgreSQL**. A utilização de outros sistema é possível, como é o caso do **MySQL**.

Com vista à disponibilidade e escalabilidade do serviço de monitorização, foram adicionadas várias máquinas responsáveis por fornecer o serviço de telemetria, de modo a eliminarmos SPOFs (*Single Point Of Failures*). Na secção em baixo, apresentamos e esclarecemos a estrutura adotada, que permite a disponibilidade e escalabilidade do serviço.

3 Estrutura adotada

No que concerne à disponibilidade e escalabilidade do serviço, foram tomadas as seguintes decisões:

- implementação de um *cluster* constituído por duas máquinas a servir o **HAProxy**. Esta ferramenta fornece um serviço de balanceamento de carga entre vários serviços. Neste contexto, é utilizado para balancear carga entre os servidores responsáveis pelo serviço **Grafana**. A utilização de apenas um balanceador, aumenta a probabilidade de negação do serviço, daí a criação do *cluster*. Para isso utilizamos o **Pacemaker** e o **Corosync**. Foi disponibilizado um IP virtual (10.10.10.68) ao qual os clientes acedem;
- utilização de duas máquinas para aplicação do **Grafana**. Em caso de necessidade de diminuir a carga das máquinas existentes, basta adicionarmos mais máquinas a servir o **Grafana** à estrutura. Por ser um serviço *stateless*, não há necessidade de criação de um *cluster*, dado que o **HAProxy** reconhece a falha deste;
- implementação de um *cluster* que fornece o serviço **Graphite**. Este *cluster* contém também duas máquinas, cada uma com as ferramentas **Pacemaker** e **Corosync**, ao qual é disponibilizado um IP virtual (10.10.10.96). É através deste IP que devem ser enviadas as métricas recolhidas pelo **collectd** e, também, requisitados os dados pelo **Grafana**;
- implementação de um *cluster* para o **PostgreSQL**, novamente com duas máquinas e com o uso das ferramentas **Pacemaker** e **Corosync**, ao qual é disponibilizado um IP virtual (10.10.10.39). A base de dados é acedida tanto pelo **Graphite** como pelo **Grafana**;
- implementação de um *cluster* responsável pela replicação dos dados, permitindo uma maior disponibilidade dos mesmos. O *cluster* conta com duas máquinas que têm instalado o serviço **DRDB** (*Distributed Replicated Block Device*), onde é efetuada a replicação dos dados nos discos **/dev/sdb**. São também utilizadas as ferramentas **Pacemaker** e **Corosync**, ao qual é disponibilizado um IP virtual (10.10.10.28) a utilizar pelos outros componentes do sistema, como é o caso do **PostgreSQL** e **Graphite**. Os dados são fornecidos pela rede através de **NFS**.

A utilização de um IP virtual em cada um dos *clusters* mencionados em cima, permite que os outros componentes da infraestrutura reconheçam o *cluster* como uma única máquina, conferindo uma transparência total no caso de uma falha da máquina do *cluster* e, deste modo, quando existe a falha de uma máquina do *cluster*, outra é imediatamente assumida como a principal ficando encarregue desse IP. De notar que, para uma maior disponibilidade do serviço, poderão ser adicionadas máquinas a cada um dos *clusters* a fornecerem serviços semelhantes.

Assim sendo, a disponibilidade e escalabilidade do sistema, é garantida através da utilização de *clusters* e de balanceadores de carga. Para ser possível uma maior escalabilidade do sistema, recomendaríamos a utilização de um balanceador de carga imediatamente antes do *cluster* do **Graphite**, sendo necessária a adição de mais um *cluster* para o mesmo efeito. Assim a carga seria distribuída entre estes dois *clusters*.

A arquitetura da infraestrutura adotada é representada na imagem em baixo.

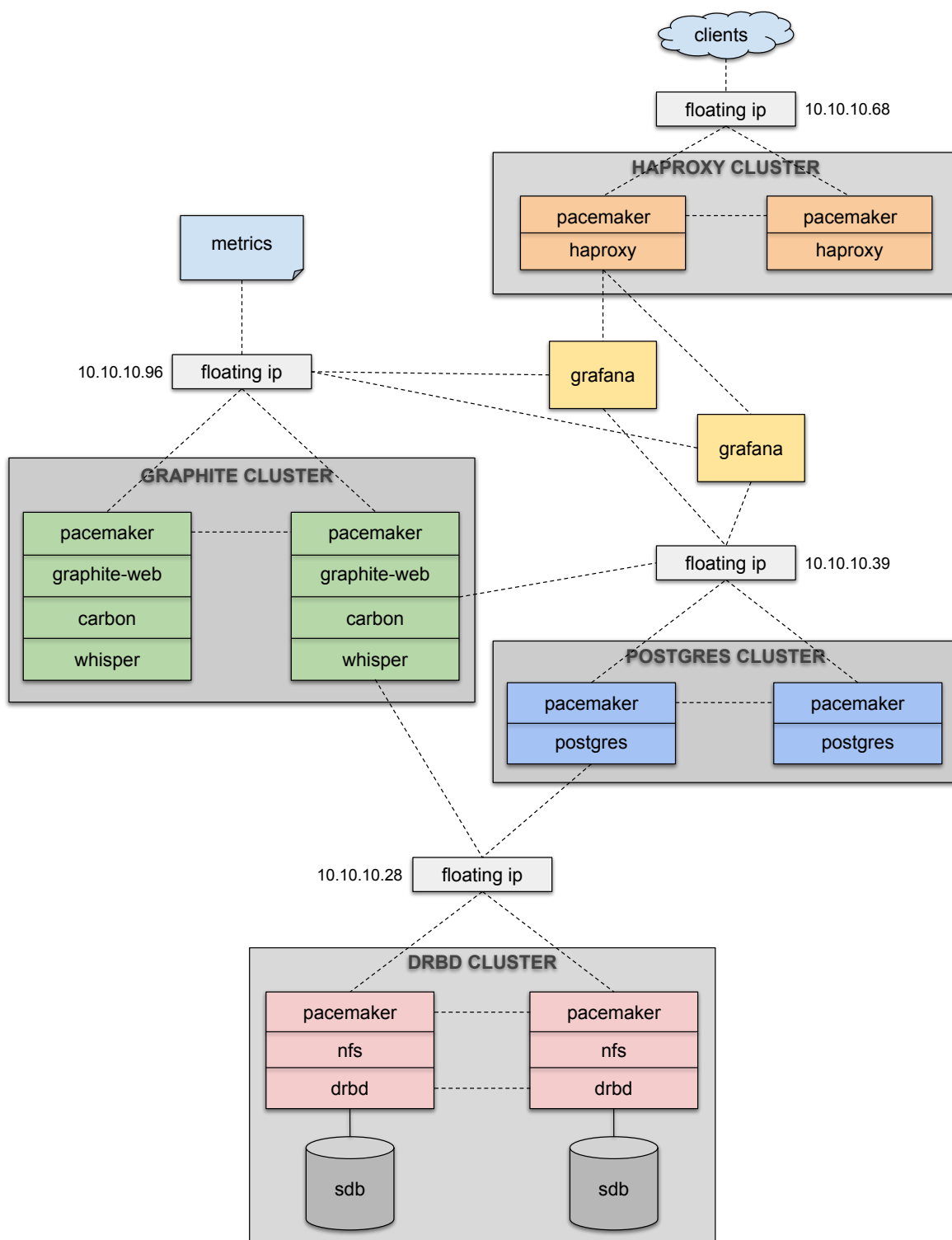


Figura 1: Arquitetura da infraestrutura.

4 Implementação do sistema

O sistema foi desenvolvido com recurso a máquinas virtuais. Foi utilizado o **VMware** para a virtualização das máquinas.

O sistema operativo utilizado foi o **CentOS 7** na versão 7.6.1810.

Para uma melhor gestão da rede, foram criadas duas redes com usos distintos:

- **Host Only:**

- Endereço da rede: 192.168.132.0;
- Utilizada para o acesso SSH entre o *host* e as máquinas virtuais;

- **NAT:**

- Endereço da rede: 10.10.10.0;
- Utilizada na comunicação entre as máquinas;

Para uma melhor organização das diferentes redes, foram atribuídos os seguintes *ips* às máquinas:

	Machine	Host Only	NAT	Floating IP
Cluster	drbd-1	192.168.132.10	10.10.10.10	10.10.10.28
	drbd-2	192.168.132.11	10.10.10.11	
Cluster	postgres-1	192.168.132.20	10.10.10.20	10.10.10.39
	postgres-2	192.168.132.21	10.10.10.21	
Cluster	haproxy-1	192.168.132.30	10.10.10.30	10.10.10.68
	haproxy-2	192.168.132.31	10.10.10.31	
Non Cluster	grafana-1	192.168.132.40	10.10.10.40	-
	grafana-2	192.168.132.41	10.10.10.41	
Cluster	graphite-1	192.168.132.50	10.10.10.50	10.10.10.96
	graphite-2	192.168.132.51	10.10.10.51	

Tabela 1: Inventário do sistema.

Caso fosse pretendido um isolamento das diferentes máquinas, seria interessante a utilização de diferentes redes. Desse modo, garantia-se que, apenas as máquinas da rede, conseguiram comunicar entre si. Uma vez que todo o sistema está virtualizado, optamos pela utilização das redes mencionadas anteriormente.

Vamos de seguida, explicar todo o procedimento efetuado para a implementação do sistema. Todo este procedimento visa a **alta disponibilidade** e **alta escalabilidade** do sistema, de modo a tornar o sistema 100% disponível e suportar picos de carga.

4.1 Sincronização do tempo nas máquinas

Um dos aspetos a ter em conta numa infraestrutura como a que é pretendida, passa pela sincronização do tempo em todas as máquinas, ou seja, garantir que o tempo é o mesmo em todas elas.

Este problema é resolvido através da utilização do protocolo *NTP* (*Network Time Protocol*).

Sendo assim, e em todas as máquinas, procedemos à instalação do *software* necessário para o efeito.

```
yum install -y ntp ntpdate
```

De seguida, procedemos à inicialização do serviço e à ativação dessa mesma inicialização no arranque da máquina. Deve ainda ser consultado o estado do serviço.

```
systemctl start ntpd
systemctl enable ntpd
systemctl status ntpd
```

Define-se ainda a *timezone* pretendida para a sincronização dos tempos, no nosso caso, *Europe/Lisbon*.

```
timedatectl set-timezone Europe/Lisbon
```

4.2 Armazenamento e replicação dos dados

Para garantirmos a disponibilidade do armazenamento dos dados, tanto para as base de dados, como para as métricas recolhidas, foi adotado um mecanismo de replicação de dados, que envolve pelo menos duas máquinas. No sistema implementado, decidiu-se utilizar duas máquinas: *drbd-1* e *drbd-2*. No caso da primeira máquina falhar, conseguimos garantir que os dados estão disponíveis e prontos a serem utilizados.

Num cenário real, seria interessante colocar as máquinas em regiões geográficas distintas, para prevenir problemas de disponibilidade, como catástrofes, falta de *internet*, etc.

Alguns aspetos importantes para a implementação deste sistema de armazenamento e replicação dos dados:

- Foi adicionado um disco de 4GB a cada máquina, *drbd-1* e *drbd-2* respetivamente;
- Utilizou-se o *software* DRBD para ser realizada a replicação dos dados. Neste caso, decidiu-se que a máquina *drbd-1* será a máquina primária e a máquina *drbd-2* a secundária;
- Foram utilizados o *Pacemaker* e o *Corosync*, em ambas máquinas, que permitem a gestão de um *cluster*, garantindo que pelo menos uma das máquinas está a efetuar um determinado serviço;
- Para podermos tornar o serviço de armazenamento dos dados disponível pela rede, foi utilizado o *software* NFS.

Em baixo, seguem um conjunto de passos necessários à implementação do *cluster*.

4.2.1 Instalação e configuração dos *DRBD*

Em ambas as máquinas, procede-se à modificação do ficheiro `/etc/hosts` de forma a resolvermos os nomes das máquinas no *cluster*.

```
10.10.10.10    drbd-1
10.10.10.11    drbd-2
```

Em ambas as máquinas, procede-se à instalação do software *DRBD* e *NFS*:

```
rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org
rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm
yum update -y
yum install -y kmod-drbd84 drbd84-utils
yum install -y nfs-utils
systemctl enable nfs-server.service
```

Procede-se ainda a criação do ficheiro de configuração `/etc/drbd.d/d1.res`, tendo especial cuidado com o nome dos discos inseridos e os endereços IP das duas máquinas. A porta 7789 deverá estar aberta em ambas as máquinas para a comunicação entre elas.

O disco final está na seguinte diretoria com o seguinte nome: `/dev/drbd1`.

```
resource d1 {
    meta-disk internal;
    device /dev/drbd1;
    net {
        protocol C;
    }
    on drbd-1 {
        disk /dev/sdb;
        address 10.10.10.10:7789;
    }
    on drbd-2 {
        disk /dev/sdb;
        address 10.10.10.11:7789;
    }
}
```

Tendo em conta a configuração em cima, é iniciado em ambas as máquinas, o recurso `d1`:

```
drbdadm create-md d1
```

Depois de iniciado, deve-se proceder ao arranque do serviço *DRBD* em simultâneo. Note-se que não deve ser ativado o serviço no arranque do sistema, uma vez que este vai ser controlado pelo *Pacemaker*.

```
systemctl start drbd.service
```

A máquina `drbd-1` deve agora ser considerada como a primária, e o processo de sincronização entre os discos deve arrancar de imediato. Efetuar o seguinte comando na máquina `drbd-1`.

```
drbdadm primary --force d1
```

Como podemos verificar pelo comando `drbdadm status` na figura em baixo, o processo de sincronização iniciou.

```
[root@drbd-2 ~]# drbdadm status
d1 role:Secondary
  disk:Inconsistent
  peer role:Primary
    replication:SyncTarget
    peer-disk:UpToDate done:8.25
```

Figura 2: Processo de sincronização entre discos.

```
[root@drbd-2 ~]# drbdadm status
d1 role:Secondary
  disk:UpToDate
  peer role:Primary
    replication:Established peer-disk:UpToDate
```

Figura 3: Ambas as máquinas sincronizadas.

Na máquina `drbd-1`, deve-se proceder à formatação do disco `/dev/drbd1` para o formato desejado, neste caso, `xf`s.

```
mkfs.xfs /dev/drbd1 -f
```

O processo de configuração dos *DRBD*'s, está agora terminado. Para caso de teste, poder-se-á proceder ao *mount* do disco para uma diretoria.

4.2.2 Criação de um *cluster*

Para a configuração de um *cluster*, começa-se por instalar as ferramentas necessárias em ambas as máquinas:

```
yum install -y pacemaker corosync pcs
```

De seguida inicia-se os serviços e ativa-se os mesmos para o arranque do sistema:

```
systemctl start pcsd.service
systemctl enable pcsd.service
systemctl enable pacemaker.service
systemctl enable corosync.service
```

Em ambas as máquinas é necessário definir a palavra passe do utilizador *hacluster*, sendo que esta deverá ser a mesma em ambas:

```
passwd hacluster
```

Esta palavra passe permite a autenticação no *cluster*, via interface *web* e/ou via terminal.

Neste momento, uma das máquinas deverá ficar responsável por se autenticar no *cluster* em ambas as máquinas, e proceder à configuração do mesmo. Neste caso, foi selecionada a máquina *drbd-1*. De referir a importância da definição dos *hosts* no ficheiro */etc/hosts*, que permitem a simplificação deste processo com a utilização dos nomes atribuídos aos endereços.

```
pcs cluster auth drbd-1 drbd-2
pcs cluster setup --name drbdcluster drbd-1 drbd-2 # ou outro nome
pcs cluster start --all # arranque do serviço de clustering nas máquinas
```

Uma vez que estão a ser utilizados apenas dois nós, o conceito de *quorum* deixa de fazer sentido neste cenário, uma vez que uma máquina é suficiente para decidir se uma máquina deixa de estar ativa. Foi desativada a funcionalidade *stonith*, uma vez que não pretendemos que uma máquina desligue a outra, visto que os dados estão a ser replicados. Numa das máquinas, efetue os seguintes comandos:

```
pcs property set no-quorum-policy=ignore
pcs property set stonith-enabled=false
```

Neste momento, o *cluster* está preparado a ser configurado através da adição de recursos, que poderá ser efetuado numa das máquinas.

De notar, que o *cluster* pode ser acedido e configurado via interface *web*, através de um *browser* com o *IP* de uma das máquinas na porta 2224.

4.2.3 Adição e configuração de recursos do *cluster*

No que concerne à escolha dos recursos do *cluster* *drbdcluster*, foi necessária a adição de recursos capazes de gerir os *drbd*, o *filesystem* e, por fim, um *IP* virtual.

Relativamente ao recurso dos *drbd*, foi criado um ficheiro temporário (*drbd_cfg*) de configuração, permitindo enviar para o *cluster* várias configurações de uma só vez.

```
pcs cluster cib drbd_cfg

pcs -f drbd_cfg resource create DrbdData ocf:linbit:drbd \
    drbd_resource=d1 op monitor interval=60s

pcs -f drbd_cfg resource master DrbdDataClone DrbdData \
    master-max=1 master-node-max=1 clone-max=2 clone-node-max=1 \
    notify=true

pcs cluster cib-push drbd_cfg --config
```

Relativamente ao recurso responsável por montar o disco na diretoria */data*, foi utilizado, uma vez mais, um ficheiro de configuração. Neste ficheiro são adicionadas algumas restrições de ordem, ou seja, o *filesystem* só é montado caso os *drbd* estejam a funcionar e exista um disco, neste caso, o *drbd1*.

```
pcs cluster cib fs_cfg

pcs -f fs_cfg resource create DataFS Filesystem \
    device="/dev/drbd1" directory="/data" fstype="xfs"
```

```
pcs -f fs_cfg constraint colocation add \
    DataFS with DrbdDataClone INFINITY with-rsc-role=Master

pcs -f fs_cfg constraint order \
    promote DrbdDataClone then start DataFS

pcs cluster cib-push fs_cfg --config
```

Foi criado o grupo *DrbdGroup* para permitir que todos os recursos desse grupo estejam ativos numa única máquina. Deste modo, temos que a máquina com o *drbd* primário, irá conter ativos os recursos do *filesystem* e *IP* virtual.

```
pcs resource group add DrbdGroup DataFS
```

Para exportarmos o *filesystem* pela rede foi também adicionado um recurso que permite exportar uma diretoria pela rede. Uma vez mais, só estará ativo se o recurso do *filesystem* também estiver.

```
pcs resource create nfs-root exportfs \
    clientspec="10.10.10.0/24" \
    options="rw, sync, no_root_squash" \
    directory="/data" \
    fsid=0 \
    --group DrbdGroup

pcs constraint colocation add \
    nfs-root with DataFS INFINITY
```

Por fim, foi adicionado um recurso do *IP* virtual, permitindo que outras máquinas vejam o *cluster* como uma máquina só.

```
pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=10.10.10.28 \
    cidr_netmask=24 nic=ens33 op monitor interval=30s \
    --group DrbdGroup

pcs constraint colocation add \
    VirtualIP with nfs-root INFINITY
```

Como podemos observar na figura em baixo, depois dos 5 recursos terem sido adicionados, existe uma máquina que está a correr todos eles, neste caso, a máquina *drbd-1*.

```
2 nodes configured
5 resources configured

Online: [ drbd-1 drbd-2 ]

Full list of resources:

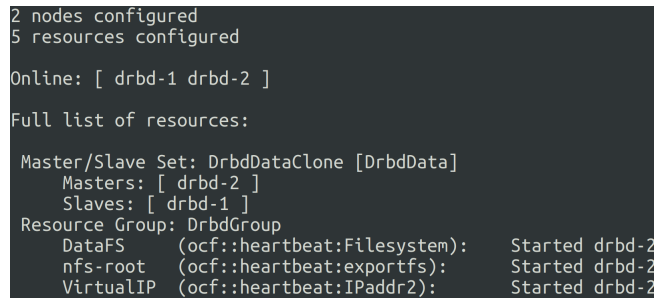
Master/Slave Set: DrbdDataClone [DrbdData]
Masters: [ drbd-1 ]
Slaves: [ drbd-2 ]
Resource Group: DrbdGroup
DataFS (ocf::heartbeat:Filesystem): Started drbd-1
nfs-root (ocf::heartbeat:exportfs): Started drbd-1
VirtualIP (ocf::heartbeat:IPaddr2): Started drbd-1
```

Figura 4: Estado do *cluster* num dado momento.

Para motivos de teste, é possível "migrarmos" os serviços para a outra máquina. Para isso basta deixarmos, temporariamente, a máquina *drbd-1* em *standby*.

```
pcs cluster standby drbd-1
pcs cluster unstandby drbd-1
```

Como podemos observar na figura em baixo, os recursos estão ativos na máquina *drbd-2*.



```
2 nodes configured
5 resources configured

Online: [ drbd-1 drbd-2 ]

Full list of resources:

Master/Slave Set: DrbdDataClone [DrbdData]
  Masters: [ drbd-2 ]
  Slaves: [ drbd-1 ]
Resource Group: DrbdGroup
  DataFS      (ocf::heartbeat:Filesystem): Started drbd-2
  nfs-root    (ocf::heartbeat:exportfs):   Started drbd-2
  VirtualIP   (ocf::heartbeat:IPaddr2):    Started drbd-2
```

Figura 5: Estado do *cluster* num dado momento.

4.3 Sistema de Gestão base de dados

Dada a necessidade da existência de um Sistema de Gestão base de dados, procedemos à criação de um *cluster* com duas máquinas: *postgres-1* e *postgres-2*. Este *cluster* é pensado de forma a garantir a disponibilidade do serviço *postgres*, garantindo que em caso de falha de uma das duas máquinas, a outra toma o lugar e o papel desta.

4.3.1 Instalação e configuração do *PostgreSQL*

Em ambas as máquinas, procede-se à instalação do *software* de base de dados *PostgreSQL* e *NFS* (ativando o mesmo a partir do arranque da máquina):

```
yum install -y postgresql-server postgresql-contrib
yum install -y nfs-utils
systemctl enable nfs-server.service
```

Uma vez que é necessária a criação de duas bases de dados (*graphite* e *grafana*), é necessário que uma das máquinas monte o sistema de ficheiros fornecido pela rede pelo *cluster* dos *DRBD*.

Todos os passos seguintes são efetuados em apenas uma máquina.

```
mkdir /data
mount -t nfs 10.10.10.28:/data /data
mkdir /data/pgdata
```

Depois de montado o sistema de ficheiros, é necessário editar o ficheiro do serviço do *postgres* (*/etc/systemd/system/postgresql.service*), de forma a iniciar com a variável *PGDATA* na diretoria anteriormente montada.

```
.include /lib/systemd/system/postgresql.service
[Service]
Environment=PGDATA=/data/pgdata
```

De seguida, é alterada a permissão de utilização da pasta *pgdata*, permissão esta fornecida ao utilizador *postgres*. De seguida são inicializados todos os ficheiros do *PostgreSQL* da diretoria e inicia-se o serviço.

```
chown postgres /data/pgdata
postgresql-setup initdb
systemctl start postgresql
```

São então criadas as duas bases de dados necessárias à estrutura, uma para o serviço prestado pelo *graphite*, e outra para o *grafana*.

```
su - postgres -c "createdb graphite"
su - postgres -c "createdb grafana"
```

De seguida, é editada a configuração do *PostgreSQL* para que a base de dados receba conexões de todos os endereços de interfaces *IP* disponíveis. Sendo assim, é editado o seguinte parâmetro em */data/pgdata/postgresql.conf*:

```
listen_addresses = '*'
```

No seguimento da intenção de receber conexões de várias máquinas é ainda editado o ficheiro */data/pgdata/pg_hba.conf*, de forma permitir a conexão de qualquer *host* à base de dados. Por questões de segurança isto deve ser descartado.

TYPE	DATABASE	USER	ADDRESS	METHOD
host	all	all	0.0.0.0/0	trust

Uma vez que os serviços vão ser tratados pelo *Pacemaker*, devemos proceder à paragem do serviço *postgres* e desabilitar o ser arranque no início da máquina.

```
systemctl stop postgresql
systemctl disable postgresql
```

4.3.2 Adição e configuração de recursos do *cluster*

O processo de criação do *cluster*, deve seguir as indicações mencionadas na secção 4.2.2. Neste processo, foi criado o *cluster postgrescluster*.

No que diz respeito à escolha de recursos a adicionar a este *cluster*, foram adicionados recursos capazes de gerir o *PostgreSQL*, o *filesystem* e um *IP* virtual.

Primeiramente é efetuado o *download* de um recurso que permite a gestão do serviço *PostgreSQL*.

```
wget -O /usr/lib/ocf/resource.d/heartbeat/postgresql \
  https://raw.githubusercontent.com/ClusterLabs/ \
  resource-agents/master/heartbeat/pgsql
chmod 755 /usr/lib/ocf/resource.d/heartbeat/postgresql
```

Foi criado um ficheiro temporário (*psql_cfg*) de configuração, permitindo enviar para o *cluster* as várias configurações estabelecidas de uma só vez.


```
pcs cluster cib psql_cfg
```

Relativamente ao recurso utilizado para montar o sistema de ficheiros na diretoria `/data`, são adicionadas as seguintes configurações ao ficheiro, passando como argumento o endereço virtual do *cluster DRBD*. Isto acontece porque pretendemos que o sistema de ficheiros esteja unicamente montado numa única máquina num dado momento. Este recurso é adicionado ao grupo *PostgresGroup*.

```
pcs -f psql_cfg resource create DataFS Filesystem \  
  device="10.10.10.28:/data" \  
  directory="/data" \  
  fstype="none" \  
  --group PostgresGroup
```

Quanto ao recurso utilizado para o *PostgreSQL*, é utilizado `ocf:heartbeat:postgresql`, um *script* para o *PostgreSQL*, que faz a gestão do mesmo como um recurso de *High Availability*. O parâmetro `pgctl` especifica o caminho para o comando `pg_ctl`, `psql` é o caminho para o comando `psql`, `pgdba` especifica o utilizador que "detém" o *PostgreSQL*, ao passo que `pgdata` especifica o caminho para a diretoria de dados da base de dados. Este mesmo recurso é também adicionado ao grupo *PostgresGroup*.

```
pcs -f psql_cfg resource create psql ocf:heartbeat:postgresql \  
  pgctl=/usr/bin/pg_ctl \  
  psql=/usr/bin/psql \  
  pgdba=postgres \  
  pgdata=/data/pgdata \  
  op monitor interval="30s" timeout="60s" \  
  op start interval="0" timeout="60s" \  
  op stop interval="0" timeout="60s" \  
  --group PostgresGroup  
  
pcs -f psql_cfg constraint colocation add \  
  psql with DataFS INFINITY
```

É ainda adicionado o recurso que configura um *IP* virtual, que permite que as máquinas vejam o *cluster* como um só. Desta forma, designado o IP 10.10.10.39 ao *cluster*.

```
pcs -f psql_cfg resource create VirtualIP ocf:heartbeat:IPaddr2 ip=10.10.10.39 \  
  cidr_netmask=24 nic=ens33 op monitor interval="30s" \  
  --group PostgresGroup  
  
pcs -f psql_cfg constraint colocation add \  
  VirtualIP with psql INFINITY
```

Para configurar a ordem pela qual os recursos devem ser inicializados é utilizado o seguinte comando:

```
pcs -f psql_cfg constraint order DataFS then psql  
pcs -f psql_cfg constraint order psql then VirtualIP
```

Desta forma, a ordem dos recursos deverá ser:

Filesystem → *PostgreSQL* → *Virtual IP*

É depois então realizado o *push* do ficheiro de configuração para o *cluster*.

```
pcs cluster cib-push psql_cfg --config
```

4.4 Sistema de balanceamento de carga do serviço *web*

Para interagir com os utilizadores da infraestrutura e balancear a carga pelas máquinas *grafana*, procedeu-se à criação de um *cluster* com duas máquinas: *haproxy-1* e *haproxy-2*. Novamente são utilizadas duas máquinas neste *cluster* de forma a garantir a disponibilidade do serviço para, em caso de falha da máquina *master*, a outra tomar o seu papel e lugar. Foi ainda novamente utilizado em ambas as máquinas os serviços Pacemaker e Corosync.

4.4.1 Instalação e configuração do *HAProxy*

Em ambas as máquinas, procede-se à instalação do *HAProxy* e ao arranque do serviço.

```
yum install -y haproxy
systemctl start haproxy
```

Já no que diz respeito à configuração do *HAProxy*, é necessário alterar o *default* do *HAProxy*, localizado em */etc/haproxy/haproxy.cfg*.

```
global
    user haproxy
    group haproxy
    daemon
    maxconn 256
    stats socket /var/lib/haproxy/stats user haproxy group haproxy mode 666
    level user

defaults
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms
    option forwardfor # app servers will know which IP sent a request
    option http-server-close

listen stats
    bind *:9000
    mode http
    stats enable
    stats hide-version
    stats realm Haproxy\ Statistics
    stats uri /stats
    stats auth root:root
```

```

frontend grafanaHttp
    bind *:80
    mode http
    default_backend grafanaFrontend

backend grafanaFrontend
    mode http
    balance roundrobin
    server grafana-1 10.10.10.40:3000 check fall 2 rise 1
    server grafana-2 10.10.10.41:3000 check fall 2 rise 1

```

No caso de se pretendido saber quais as máquinas ativas no *HAProxy*, neste caso duas (*grafana-1* e *grafana-2*), foi adicionada a secção **listen stats**, que permite também a obtenção de estatísticas como a transferência de dados, o número de conexões, , etc. Estes dados podem ser acedidos via *web* através do IP virtual do *cluster* na porta 9000.

Na secção **backend** é então onde devem ser configurados os servidores pelos quais será distribuído o tráfego da aplicação. É definido que as solicitações *HTTP* serão passadas para os servidores listados e que a estratégia de balanceamento pelos servidores é através do algoritmo *Round Robin*.

Após todas estas configurações serem estabelecidas é verificado se as mesmas são corretas/válidas, sendo depois desativado o serviço *HAProxy* no arranque da máquina, uma vez que irá ser tratado pelo *cluster*.

```

haproxy -c -V -f /etc/haproxy/haproxy.cfg
systemctl stop haproxy.service
systemctl disable haproxy

```

4.4.2 Adição e configuração de recursos do *cluster*

O processo de criação do *cluster* deve seguir as indicações mencionadas na secção 4.2.2. Neste processo, foi criado o *cluster haproxycluster*.

No que concerne à escolha de recursos a adicionar a este *cluster*, são adicionados recursos capazes de gerir o *HAProxy* e a definição de um *IP* virtual.

Em ambas as máquinas, é efetuado o *download* do recurso que permite ao *cluster* gerir o *HAProxy*. São ainda alteradas as permissões do ficheiro obtido.

```

wget -O /usr/lib/ocf/resource.d/heartbeat/haproxy \
    http://github.com/ruscki/cluster-agents/raw/master/haproxy

chmod 755 /usr/lib/ocf/resource.d/heartbeat/haproxy

```

Após isto, em apenas uma das máquinas é criado o recurso no *cluster*. É indicado o caminho para o ficheiro binário e para o ficheiro de configuração do *HAProxy*, sendo ainda definido que uma operação de monitorização deve ser realizada a cada 10 segundos..

```

pcs resource create haproxy ocf:heartbeat:haproxy \
    binpath=/usr/sbin/haproxy conffile=/etc/haproxy/haproxy.cfg \
    op monitor interval=10s

```

É ainda adicionado o recurso que configura um IP virtual, que permite que as máquinas vejam o *cluster* como um só. Neste caso, foi designado o IP 10.10.10.68.

```
pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=10.10.10.68 \
cidr_netmask=24 nic=ens33 op monitor interval=30s
```

Após isto, os dois recursos configurados são adicionados ao grupo *HAProxyGroup*, e é definida a ordem pela qual os dois recursos devem ser inicializados.

```
pcs resource group add HAProxyGroup haproxy VirtualIP
pcs constraint order haproxy then VirtualIP
```

Desta forma, a ordem dos recursos deverá ser:

$$HAProxy \rightarrow Virtual\ IP$$

4.5 *Graphite*

Tal como ilustrado na Figura 1, este *cluster* irá receber as métricas por nós definidas com origem nas diferentes máquinas da infraestrutura estudada. Para tal, a escolha recai novamente na implementação de duas máquinas no *cluster*, **graphite-1** e **graphite-2**. A ideia passa por cada uma das máquinas possuir os três *software's* do *Graphite*: *graphite-web*, *carbon* e *whisper*. Para além disso é também utilizado o *Pacemaker* que, como já referido, permite a gestão do *cluster*, garantindo que pelo menos uma das máquinas está ativa e em serviço.

4.5.1 Instalação e configuração do *Graphite*

De forma a configurar cada uma das máquinas do *cluster*, inicialmente faz-se a instalação dos programas que serão necessários para a implementação do sistema, como o *Python* e algumas bibliotecas, *gcc* ou *nfs-utils*.

```
yum install -y epel-release

yum install -y python-devel cairo-devel libffi-devel
yum install -y python-pip
yum install -y python-django
yum install -y gcc
yum install -y nfs-utils

pip install --upgrade pip setuptools
pip install django-tagging
pip install m2r
pip install pycpg2-binary
pip install gunicorn
pip install pytz
```

É realizada a instalação do *Graphite* e seus componentes com recurso ao **pip**, como é de seguida apresentado.

```
export PYTHONPATH="/opt/graphite/lib:/opt/graphite/webapp/"

pip install --no-binary=:all: \
    https://github.com/graphite-project/whisper/tarball/master
pip install --no-binary=:all: \
    https://github.com/graphite-project/carbon/tarball/master
pip install --no-binary=:all: \
    https://github.com/graphite-project/graphite-web/tarball/master
```

Após esta instalação, os ficheiros de configuração de exemplo do *Graphite* passam a ser usados como a configuração base do programa.

```
cp /opt/graphite/conf/storage-schemas.conf.example \
    /opt/graphite/conf/storage-schemas.conf
cp /opt/graphite/conf/storage-aggregation.conf.example \
    /opt/graphite/conf/storage-aggregation.conf
cp /opt/graphite/conf/graphTemplates.conf.example \
    /opt/graphite/conf/graphTemplates.conf
cp /opt/graphite/conf/graphite.wsgi.example \
    /opt/graphite/conf/graphite.wsgi
cp /opt/graphite/webapp/graphite/local_settings.py.example \
    /opt/graphite/webapp/graphite/local_settings.py
cp /opt/graphite/conf/carbon.conf.example \
    /opt/graphite/conf/carbon.conf
```

De seguida, é necessário editar as definições da base de dados do *Graphite* de acordo com a base de dados de *PostgreSQL* montada no *cluster postgrescluster*. Neste seguimento, é alterado o ficheiro `/opt/graphite/webapp/graphite/local_settings.py` no qual é colocado na configuração da base de dados o nome da mesma (`graphite`), o utilizador (`postgres`) e a *password* (que neste caso não foi definida). Também é declarado o endereço da máquina com a base de dados, utilizando o *Virtual IP* do *cluster postgrescluster* (10.10.10.39) através da porta 5432.

```
TIME_ZONE = "Europe/Lisbon"
SECRET_KEY = '571b8234-994f-4c43-961f-6b051cd29b00'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'graphite',
        'USER': 'postgres',
        'PASSWORD': '',
        'HOST': '10.10.10.39',
        'PORT': '5432',
    }
}
```

De seguida, é iniciado o processo de sincronização do estado da base de dados com o conjunto atual de modelos e migrações. Desta forma, é iniciada a gestão da base de dados *Graphite*.

```
PYTHONPATH=/opt/graphite/webapp django-admin migrate \
    --settings=graphite.settings \
    --run-syncdb
```

É também criado um *Graphite superuser* para a gestão da aplicação *web* através do comando:

```
PYTHONPATH=/opt/graphite/webapp django-admin createsuperuser \
    --settings=graphite.settings
```

Na primeira configuração do *Graphite*, é necessário montar o sistema de ficheiros fornecido pelo *cluster* DRBD e criar a diretoria para o armazenamento dos dados do *whisper*.

```
mkdir /data
mount -t nfs 10.10.10.28:/data /data
mkdir /data/graphite-storage
mkdir /data/graphite-storage/whisper
```

De seguida, é necessário alterar a diretoria de *storage* da base de dados no *Graphite* tendo em conta as pastas criadas anteriormente. Sendo assim, é modificado o ficheiro `/opt/graphite/conf/carbon.conf`:

```
LOCAL_DATA_DIR = /data/graphite-storage/whisper/
```

Assim, o ficheiro `/opt/graphite/webapp/graphite/local_settings.py` também precisa de ser alterado com base na configuração anterior referenciada:

```
STORAGE_DIR = '/data/graphite-storage'/
```

De seguida são realizadas as alterações no que diz respeito ao *daemon Carbon*.

Para isso, é criada uma unidade de ficheiro do tipo `system.d`, que consiste na criação do serviço *carbon* (`/etc/systemd/system/carbon.service`). O conteúdo do mesmo é apresentado de seguida.

```
[Unit]
Description = Carbon Metrics store

[Service]
Type = forking
GuessMainPID = false
PIDFile = /opt/graphite/storage/carbon-cache-a.pid
ExecStart = /opt/graphite/bin/carbon-cache.py start

[Install]
WantedBy = multi-user.target
```

Sendo assim, para que as alterações efetuadas causem efeito, deve-se proceder à re-inicialização dos serviços, assim como uma possível utilização de `status` para verificação do estado do mesmo.

```
systemctl daemon-reload
systemctl start carbon
systemctl status carbon
```

É criada também uma nova unidade de ficheiro do tipo `system.d` localizado em `/etc/systemd/system/graphite-web.service` para o componente *graphite-web*. O conteúdo do mesmo é apresentado de seguida.

De notar que neste ficheiro, o serviço *graphite-web* é iniciado com os seguintes parâmetros: `0.0.0.0:8080`, indicando que a interface *web* estará à escuta de todos os endereços na porta 8080.

```
[Unit]

[Service]
Environment=PYTHONPATH="/opt/graphite/lib:/opt/graphite/webapp/"
WorkingDirectory=/opt/graphite/webapp
ExecStart=/usr/bin/gunicorn -u graphite -g graphite -b 0.0.0.0:8080 \
  --log-file=/opt/graphite/storage/log/webapp/gunicorn.log wsgi:application
Restart=on-failure
User=graphite
Group=graphite
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

Tal como realizado anteriormente, é novamente necessário reinicializar os serviços, de forma a que as alterações dos ficheiros de configuração tenham efeito.

```
systemctl daemon-reload
systemctl start graphite-web
systemctl status graphite-web
```

Procede-se à criação de um conta de utilizador para o *Graphite* (utilizador com o nome `graphite`). De seguida, é dado ao mesmo utilizador o papel de proprietário da diretoria `/opt/graphite`.

```
useradd -d /opt/graphite graphite
chown graphite /opt/graphite -R
```

4.5.2 Adição e configuração de recursos do *cluster*

O processo de criação do *cluster*, deve seguir as indicações mencionadas na secção 4.2.2. Neste processo, foi criado o *cluster* `graphitecluster`.

No que diz respeito à escolha de recursos a adicionar a este *cluster*, foram adicionados recursos capazes de gerir o *filesystem*, o *Carbon*, o *Graphite-Web* e um *IP* virtual.

Primeiro, é adicionado o recurso utilizado para montar o sistema de ficheiros na diretoria `/data`, tendo em conta o endereço virtual do *cluster* *DRBD*. Isto acontece porque pretendemos que o sistema de ficheiros esteja unicamente montado numa única máquina num dado momento. Todos os recursos são adicionados ao grupo `GraphiteGroup`, garantindo que todos os recursos do mesmo estão todos a "correr" numa das máquinas do *cluster*.

```
pcs resource create DataFS Filesystem \  
    device="10.10.10.28:/data" \  
    directory="/data" \  
    fstype="none" \  
    --group GraphiteGroup
```

Depois, é adicionado o recurso para a gestão do *Carbon* pelo *cluster*.

```
pcs resource create Carbon systemd:carbon \  
    --group GraphiteGroup
```

É adicionado de seguida ao *cluster* o recurso para a gestão do *Graphite-Web*, sendo colocado no mesmo grupo mencionado anteriormente.

```
pcs resource create GraphiteWeb systemd:graphite-web \  
    --group GraphiteGroup
```

É ainda adicionado o recurso que configura um *IP* virtual, que permite que as máquinas vejam o *cluster* como um só. Desta forma, é designado o *IP* 10.10.10.39 ao *cluster*.

```
pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=10.10.10.96 \  
    cidr_netmask=24 nic=ens33 op monitor interval=30s \  
    --group GraphiteGroup
```

Para configurar a ordem pela qual os recursos devem ser inicializados são utilizados os seguintes comandos:

```
pcs constraint order DataFS then Carbon  
pcs constraint order Carbon then GraphiteWeb  
pcs constraint order GraphiteWeb then VirtualIP
```

Desta forma, a ordem dos recursos deverá ser:

Filesystem → *Carbon* → *Graphite-Web* → *Virtual IP*

4.6 Grafana

Tal como referido anteriormente, o **Grafana** é uma aplicação *web*, sendo que a mesma permite a criação de *dashboards* e análise de métricas para o sistema de monitorização.

Desta forma, são implementadas duas máquinas, **grafana-1** e **grafana-2**. Por se tratar de um serviço *stateless* (sem estado), não existe a necessidade de criação de um *cluster*, dado que o *HAProxy* reconhece a falha deste. Desta forma, cada uma das máquinas é utilizada para suporte e visualização dos dados recolhidos e apresentados através deste serviço aos utilizadores da infraestrutura, sendo que as duas máquinas efetuam exatamente o mesmo serviço e acedem aos mesmos recursos, sendo que o balanceamento do tráfego de cada uma das máquinas é feito pelo *HAProxy* (encaminhamento dos pedidos).

4.6.1 Instalação e configuração do *Grafana*

O processo de configuração começa por instalar em ambas as máquinas o *software* necessário para o efeito desejado.

```
yum install fontconfig freetype* urw-fonts -y
```

Após isto, procede-se à instalação do *Grafana* nas duas máquinas. É criado o ficheiro `/etc/yum.repos.d/grafana.repo` com o seguinte conteúdo:

```
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

De seguida, e tendo em conta este ficheiro criado, pode então proceder-se à instalação do serviço pretendido.

```
yum install grafana -y
```

Procede-se à ativação do serviço de servidor *Grafana* e à inicialização do mesmo serviço. Deve-se proceder à verificação desta inicialização através do `netstat`.

```
systemctl enable grafana-server.service
systemctl start grafana-server

netstat -plntu
```

Após este processo ser bem procedido, podemos passar à modificação das configurações do *Grafana*. O mesmo é conseguido através da edição do ficheiro `/etc/grafana/grafana.ini`.

O *Grafana* necessita de uma base de dados na qual possa armazenar utilizadores e *dashboards* (entre outros dados), sendo assim, a secção `[database]` no ficheiro é onde são estabelecidas as definições da mesma.

```
[database]
type = postgres
host = 10.10.10.39:5432
name = grafana
user = postgres
```

É definido que o tipo da base de dados utilizada é o `postgres` (poderia ser utilizado como alternativas `mysql` ou `sqlite3`). É definido o *host* onde se deve proceder ao acesso à base de dados, passando o *IP* virtual do `postgrescluster`. Para além disto é estabelecido o nome da base de dados e o utilizador da mesma.

Na secção `[security]` são definidas como configurações de segurança:

- `admin_user` → o nome do utilizador administrador por padrão do *Grafana*, que detem totais permissões.
- `admin_password` → a *password* do administrador definido em cima. É definido uma vez na primeira execução.

Ambos os valores são deixados com os valores pré-definidos (`admin`).

```
[security]
admin_user = admin
admin_password = admin
```

Por último, a secção `[session]` define as configurações das sessões que são estabelecidas com os utilizadores do sistema implementado.

```
[session]
provider = postgres
provider_config = user=postgres host=10.10.10.39 port=5432 dbname=grafana \
sslmode=disable
```

As sessões devem ser armazenadas num local diferente da própria aplicação *Grafana*, uma vez que estas devem estar disponíveis para todas as aplicações, para, no caso de falha, um cliente continuar autenticado no sistema.

Uma boa solução seria a utilização de um servidor *Redis*. Uma vez que o *Grafana* permite a possibilidade de armazenar as sessões na base de dados principal (neste caso *Postgres*), foi decidido não implementar este servidor. De notar que o controlo de sessões via base de dados pode trazer um impacto significativo no caso de existirem múltiplos utilizadores ao mesmo tempo, sobrecarregando a base de dados.

5 Escalabilidade e disponibilidade da infraestrutura

Por forma a concluirmos sobre a escolha da estrutura adotada, é importante resumirmos alguns aspetos que são importantes que garantem a escalabilidade e disponibilidade do sistema.

Tendo em conta todo o processo de implementação da infraestrutura anteriormente demonstrado, podemos tirar as seguintes conclusões:

- quanto à **disponibilidade** da infraestrutura planeada, a mesma é conseguida através da colocação de duas máquinas por cada um dos *clusters* desenhados, através da replicação dos dados e, ainda, pela utilização de duas máquinas *Grafana* para responder às solicitações dos clientes. Desta forma, é garantida a existência de uma máquina responsável para fornecer um determinado serviço. Pretendemos salientar que a utilização de apenas duas máquinas por *cluster* pode não ser necessário, assim como a replicação dos dados ser efetuada em apenas duas máquinas e, consequentemente, dois discos. A adição de mais máquinas pode influenciar a diminuição de SPOFs - *Single Point of Failure*. O mesmo acontece com as máquinas a servir o *Grafana*, as duas estão simultaneamente ativas e a disponibilizar exatamente o mesmo tipo de serviço, mas em caso de falha de uma delas, os pedidos são garantidos pela outra máquina disponibilizada. A adição de mais máquinas, para além de permitir uma diminuição da carga, permite reduzir pontos de falha únicos e, consequentemente, a indisponibilidade do sistema. Ainda como medida de disponibilidade, foi garantido que as sessões dos utilizadores no *Grafana* estão a ser armazenadas na base de dados, permitindo que caso ocorra uma falha de um servidor *Grafana*, o utilizador não se aperceba dessa falha;
- quanto à **escalabilidade** da infraestrutura, com a arquitetura adotada, é esperado que através da monitorização da infraestrutura, e face às necessidades que se possam surgir, esta possa facilmente ser alterada de forma a dar resposta às crescentes necessidades. Por exemplo, se o número de utilizadores a solicitar conexões às máquinas *Grafana* aumentar (o que pode ser verificado através da monitorização do sistema), é possível ser adicionada uma ou mais máquinas a servir o *Grafana*, permitindo reduzir **picos de carga** nas máquinas existentes. Uma solução para a escalabilidade dos balanceadores de carga, passa por ser adicionado mais um *cluster* de balanceadores, e ser utilizado o *DNS Round Robin*, diminuindo significativamente a carga dos balanceadores. A escalabilidade dos servidores *Graphite* é também um ponto a considerar, com a adição de um novo *cluster* a servir o *Graphite* com um balanceador imediatamente na camada acima, distribuindo a carga entre os *clusters*. Todas estas recomendações visam a escalabilidade do sistema, de modo a responder eficazmente aos utilizadores e a resolver casos em que os tempos de resposta são altos.
- relativamente à **política de balanceamento** e, como já foi falado nos pontos anteriores, foi um aspeto tido em conta durante a fase de estruturação do sistema: a utilização do *HAProxy* permitiu a distribuição de carga pelos diferentes servidores *Grafana*, através de um algoritmo *Round-Robin*. No entanto, a utilização deste algoritmo pode não ser eficaz na distribuição de carga, uma vez que não toma

como partido que todos os servidores apresentam *hardware* heterogéneo. Neste caso, deveria ser utilizado um algoritmo que tivesse em conta vários fatores de balanceamento (*hardware, carga do sistema, pedidos ativos, etc*).

6 Análise do sistema

De forma a poder agir pro-ativamente e determinar antecipadamente algumas falhas que possam haver na infraestrutura, decidiu-se analisar as seguintes métricas, em cada uma das máquinas:

- **CPU:** Deste modo, pretende-se avaliar se a gestão desta métrica está a ser efetuada com sucesso, isto é, se todas as máquinas estão a utilizar a maior parte do CPU destinado a elas, ou então, se as mesmas necessitam de menos CPU dedicado.
- **RAM:** Através da análise desta métrica, pretende-se avaliar se a gestão desta métrica está a ser efetuada com sucesso, isto é, se a RAM dedicada a cada máquina é a suficiente e, por outro lado, se a maior parte desta é utilizada por parte dos diferentes serviços.
- **Rede:** Através da análise do tráfego da rede, aliada a interfaces presentes na rede (neste caso, utilizaram-se as interfaces `ens33` e `ens36`), pacotes transmitidos por segundo e erros de interfaces, consegue-se, assim, determinar se existe congestão na rede, verificando se a carga do sistema se encontra corretamente balanceada entre todas as máquinas e se todos os pedidos dos clientes são efetuados com sucesso e rapidamente. Assim, pode ser avaliado se serão necessárias mais máquinas na infraestrutura ou outros métodos de balanceamento de carga.

Esta monitorização é efetuada em cada máquina, através da ferramenta, já anteriormente referenciada, o `collectd`. Esta ferramenta recolhe as métricas, já referidas, sendo os seus resultados disponibilizados ao *cluster* das máquinas `graphite`. O componente `carbon` da máquina ativa desse *cluster* recolhe esses resultados, sendo armazenados pelo componente `whisper`, no *filesystem* distribuído pela rede. Os resultados são evidenciados por uma das máquinas `grafana` (a que foi direcionada pelo `HAProxy` e se encontra a servir o cliente naquele momento), que se encontram ligadas ao *cluster* das máquinas `graphite`.

Deste modo, o ficheiro `/etc/collectd.conf`, ficheiro de configuração do serviço `collectd`, em cada máquina foi configurado da seguinte maneira:

```
Hostname      "<nome_da_máquina>"

LoadPlugin syslog

LoadPlugin cpu
LoadPlugin memory
LoadPlugin interface
LoadPlugin write_graphite

<Plugin "syslog">
  LogLevel "info"
  NotifyLevel "OKAY"
</Plugin>

<Plugin cpu>
  ValuesPercentage true
```

```

    ReportByCpu true
</Plugin cpu>

<Plugin memory>
    ValuesAbsolute true
    ValuesPercentage true
</Plugin memory>

<Plugin "interface">
    Interface "ens33"
    Interface "ens36"
    IgnoreSelected true
</Plugin>

<Plugin write_graphite>
    <Node "graphite">
        Host "10.10.10.96"
        Port "2003"
        Protocol "tcp"
        LogSendErrors true
        Prefix "collectd."
        StoreRates true
        AlwaysAppendDS true
        EscapeCharacter "_"
    </Node>
</Plugin>

```

De entre as métricas já referenciadas, importa destacar neste ficheiro o *plugin write_graphite*, responsável por transmitir cada resultado da recolha das métricas de cada máquina para o *graphitecluster*, identificado no campo *Host* pelo IP virtual 10.10.10.96. Será transmitido através da porta 2003.

Para visualizar as métricas no *Grafana*, terá de ser escolhido o campo *collectd* primeiramente, determinado pelo campo *Prefix* deste *plugin*, seguido do *hostname*, determinado através do *hostname* inserido neste ficheiro.

Além desta configuração comum, decidiu-se recolher outras métricas em algumas máquinas:

- **HAProxy**: De forma a recolher métricas do *HAProxy* e verificar o seu correto funcionamento, é utilizado o *plugin collectd-haproxy* na versão 1.2.1, construído com foco numa fácil instalação e configuração. O projeto encontra-se no *PyPI* e, por isso deve ser instalado com o comando:

```
pip install collectd-haproxy
```

Para que funcione corretamente, a instância *HAProxy* necessita de ter o *socket* para estatísticas ativo (*stats socket*). Sendo assim, adiciona-se a seguinte configuração ao *collectd* das máquinas deste *cluster*:

```

LoadPlugin "python"

<Plugin python>
    Import "collectd_haproxy"

```

```

    <Module haproxy>
      Socket "/var/lib/haproxy/stats"
    </Module>
  </Plugin>

```

- **DRBD**: No *cluster* das máquinas *drbd*, decidiu-se verificar a utilização do disco */dev/drbd1*. Com esta monitorização é possível determinar se o espaço em disco está a ser bem gerido, de forma a que não se percam dados importantes, de modo a não comprometer a escalabilidade do sistema. Através desta decisão, utilizou-se o *plugin* *df*, determinante para a recolha dos resultados das métricas definidas anteriormente.

```

LoadPlugin df

<Plugin "df">
  Device "/dev/drbd1"
  MountPoint "/data"
  FSType "xfs"
  IgnoreSelected false
</Plugin>

```

6.1 Exemplos de *Dashboards* no *Grafana*

Através destas métricas que decidimos levantar em cada uma das máquinas da infraestrutura, são vários os dados que podemos consultar e várias as *dashboards* que podem ser criadas através da interface gráfica do *Grafana*.

Um exemplo concreto desta aplicação passa pela criação de um *dashboard* que permita monitorizar a utilização de *CPU* por cada uma das máquinas. O *output* devolvido é apresentado de seguida que mostra a percentagem da utilização de *CPU* das máquinas organizadas em 5 gráficos diferentes: *haproxycluster*, *graphitecluster*, *postgrescluster*, *drbdcluster* e os dois servidores *grafana-1* e *grafana-2*.

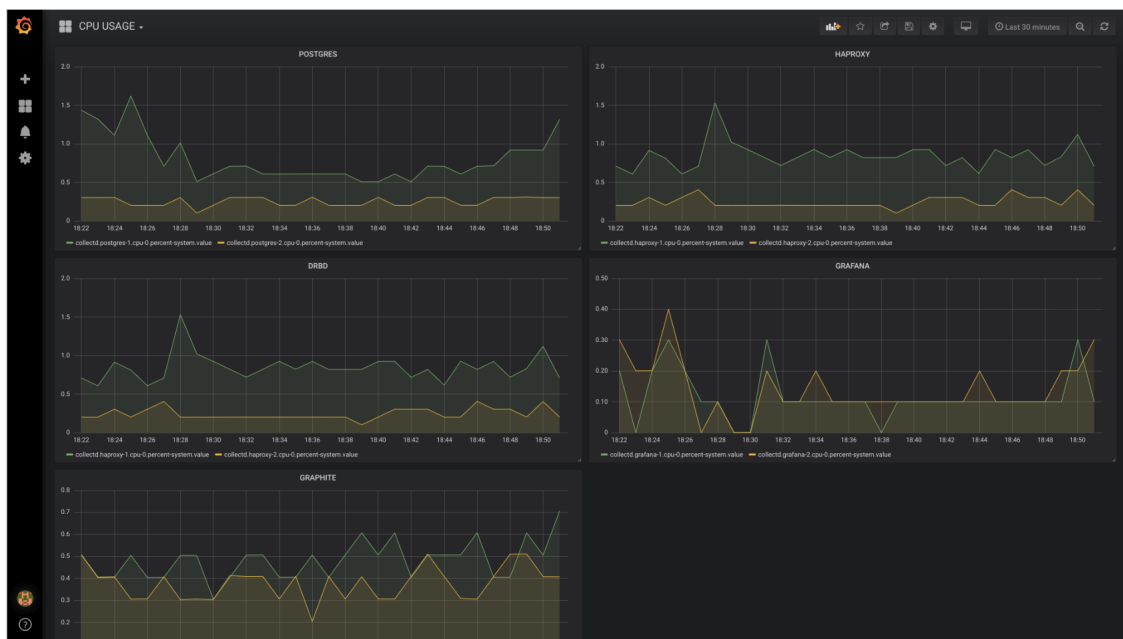


Figura 6: *Dashboard* para monitorização da utilização de *CPU*.

Outro *dashboard* criado tendo em conta os mesmos grupos de gráficos é para a monitorização da *RAM* em uso. Sendo assim, neste caso é possível consultar dados sobre a quantidade de memória livre de cada máquina ao longo do tempo.



Figura 7: *Dashboard* para monitorização da utilização de *RAM*.

Atendendo às duas imagens em cima, é possível constatar a diferença nas métricas de um *cluster*. Há maior utilização de recursos na máquina dita ativa do que na máquina em *standby*. No caso das máquinas **grafana-1** e **grafana-2**, ambas apresentam a utilização de recursos muito semelhantes, por estarem em uso simultâneo.

Foram ainda realizadas outras *dashboards* que nos permitem ter uma visão geral do sistema, como por exemplo, saber se os servidores *frontend* estão ativos e qual o estado do balanceador.

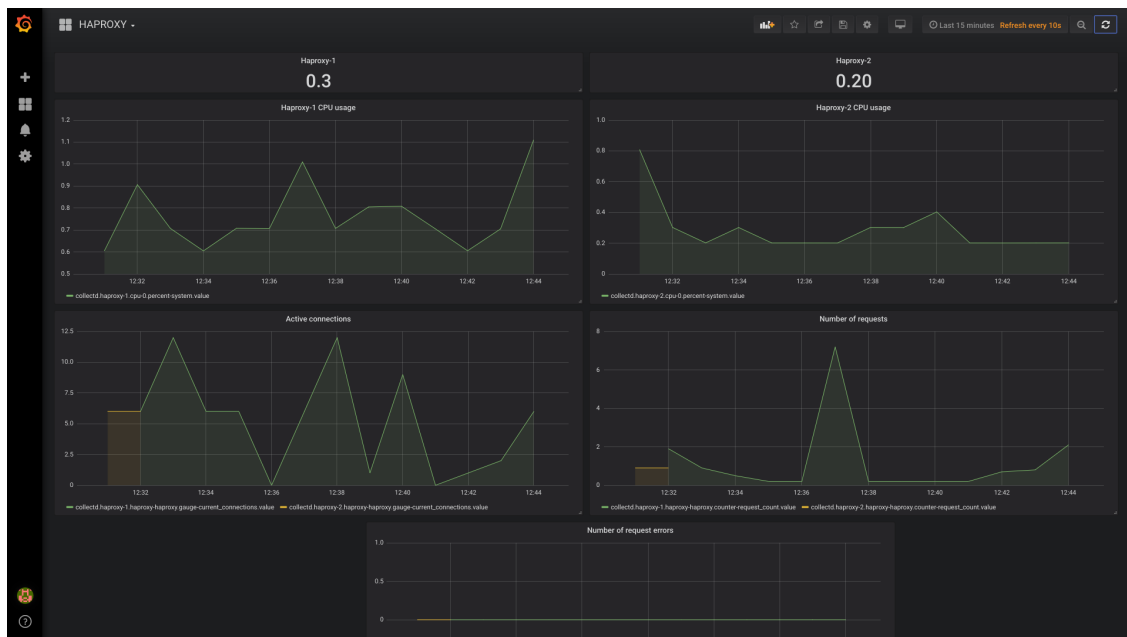


Figura 8: *Dashboard* para monitorização da utilização do haproxycluster.

Como dito anteriormente, para analisar o estado do balanceador, foram monitorizadas as seguintes métricas: utilização de CPU, por cada máquina desse *cluster*, número de conexões ativas e número de pedidos em cada máquina, número de erros resultantes de pedidos efetuados e o último registo do número de conexões. Neste seguimento, verificou-se que a maior utilização de CPU (neste caso, por parte da máquina haproxy-1) deve-se a uma maior disponibilidade para atender os pedidos dos clientes, bem como efetuar várias conexões ao mesmo tempo. É de notar que em ambas as máquinas não ocorreram erros resultantes dos pedidos dos vários clientes.



Figura 9: *Dashboard* para monitorização do disco /dev/drbd1 nos DRBD's.

Foi criada outra *dashboard* que permite monitorizar o disco /dev/drbd1 do *cluster* DRBD. Desta forma, os dois gráficos apresentados em cima, mostram o espaço utilizado e livre, respetivamente.

É importante referir que a criação das *dashboards* vai um bocado ao encontro do que o administrador do sistema pretende monitorizar.

7 Conclusão

Através da realização deste trabalho, podemos concluir que as noções sobre alta disponibilidade de infraestruturas e escalabilidade das mesmas foram aprofundadas e consolidadas.

Para além disso, este projeto foi ainda uma forma de tomarmos conhecimento de aplicações como o *Graphite* e respetivos componentes, o *Grafana* e o *Collectd* para a recolha de métricas que permitam uma melhor monitorização de máquinas de uma infraestrutura.

Possíveis melhorias a este trabalho poderiam passar por utilizar uma melhor forma de *load balancing* para os pedidos que os servidores *Grafana* recebem dos clientes, assim como da criação de volumes lógicos, *LVM* (*Logical Volume Manager*), em cima do disco *DRBD*, permitindo uma expansão de armazenamento no futuro.

No que diz respeito a alterações ao sistema em caso de necessidade de maior resposta aos utilizadores do sistema, o mesmo pode ser conseguido como referido anteriormente através da adição de mais máquinas ao sistema ou da criação de novos *clusters*, permitindo a alta escalabilidade da infraestrutura.

Quanto à disponibilidade do sistema desenvolvido, a mesma já é conseguida com a colocação de duas máquinas por cada um dos *clusters* desenvolvidos (uma máquina ativa e uma em *standby*), através da replicação dos dados e também pela utilização de duas máquinas *Grafana* para responder aos pedidos provenientes dos clientes da aplicação.

Como forma de avaliar o desempenho da aplicação e infraestrutura implementada, seria interessante recorrer a uma ferramenta como o *Apache JMeter*, que realiza a geração de carga e medição de desempenho, para realizar testes de *stress* à aplicação. Com recurso a esta ferramenta, podem ser geradas tarefas automaticamente, de forma a testar vários tipos de serviços na aplicação e analisar e avaliar o desempenho dos mesmos, até mesmo como forma de perceber se a infraestrutura desenhada vai de encontro ao efeito pretendido.

8 Bibliografia

HIGH AVAILABILITY ADD-ON ADMINISTRATION. Acedido em 21 de Janeiro de 2019, em <https://goo.gl/G2dn3f>.

Installing Graphite. Acedido em 21 de Janeiro de 2019, em <https://goo.gl/9Dmny6>.

Thomas, Shaun. (Setembro, 2012). High Availability with PostgreSQL and Pacemaker. Acedido em 21 de Janeiro de 2019, em <https://goo.gl/cZN31d>.

How to setup Grafana for high availability. Acedido em 21 de Janeiro de 2019, em <https://goo.gl/LxyvnW>.