Balanceamento de Carga com HAProxy e Keepalived

Disponibilidade e Desempenho

2023 - 2024

Fábio Capobianchi de Souza

Rodrigo de Oliveira Nascimento

a2018015299@isec.pt

a2021141973@isec.pt

Projeto B

Licenciatura em Engenharia Informática

Ramo de Redes e Administração de Sistemas

Instituto Politécnico de Coimbra

Instituto Superior de Engenharia de Coimbra

Agradecimentos

Gostariamos de agradecer ao Professor Luís Santos, por oferecer todo suporte necessário para realização deste Projeto, bem como o incentivo e orientação aos alunos, de forma que possam obter sucesso tanto na vida acadêmica, quanto no futuro percurso profissional.

Também gostaria de agradecer ao Eng. Ricárdo Tomás, por ter disponibilizado seu conhecimento e seu tempo, para que os alunos pudessem ter uma base de trabalho execional.

Resumo

Atualmente, a Disponibilidade e Desempenho são fatores cruciais para o êxito de qualquer sistema ou aplicação. Situações de baixa disponibilidade, resultantes de falhas ou gestão inadequada, podem gerar danos catastróficos, incluindo perdas financeiras. Este documento apresenta diversos métodos e tecnologias para prevenção, mitigação e recuperação de sistemas e aplicações. Serão explorados tópicos relacionados com a configuração de servidores, o uso de balanceadores de carga como o HAProxy, estratégias para alta disponibilidade com ferramentas como o Keepalived, e a implementação de clusters de bases de dados, exemplificado através do Galera Cluster para MariaDB. O objetivo é fornecer uma visão abrangente sobre as práticas e ferramentas que contribuem para a robustez e fiabilidade dos sistemas, minimizando riscos e maximizando a disponibilidade.

Palavras-chave: Disponibilidade, Desempenho.

Abstract

Currently, Availability and Performance are critical factors for the success of any system or application. Instances of low availability, stemming from failures or inadequate management, can lead to catastrophic damages, including financial losses. This document introduces various methods and technologies for the prevention, mitigation, and recovery of systems and applications. Topics related to server configuration, the use of load balancers such as HAProxy, high availability strategies with tools like Keepalived, and the implementation of database clusters, illustrated through the Galera Cluster for MariaDB, will be explored. The goal is to provide a comprehensive overview of practices and tools that contribute to the robustness and reliability of systems, minimizing risks and maximizing availability.

Keywords: Availability, Performance.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

HTTP *Hypertext Transfer Protocol*.

PHP *Hypertext Preprocessor*

VIP *Virtual IP Adress*

VRRP *Virtual Router Redundancy Protocol*

NIF *Número de Identificação Fiscal*

IP *Internet Protocol*

.

Capítulo 1

Introdução

Este relatório tem como objetivo documentar o trabalho desenvolvido ao longo do Primeiro semestre do ano letivo de 2023/2024, no âmbito da unidade curricular Disponibilidade e Desempenho, da Licenciatura em Engenharia Informática - ramo de Redes e Administração de Sistemas do Instituto Superior de Engenharia de Coimbra.

Capítulo 2

Objetivos

Contents

2.1 Objetivos Gerais

O presente projeto tem como principal desígnio implementar uma infraestrutura de alta disponibilidade e desempenho para hospedar uma aplicação web dinâmica. Os objetivos gerais abrangem:

- Configurar e otimizar um ambiente de balanceamento de carga através do uso do HAProxy.
- Garantir a alta disponibilidade do sistema mediante a implementação do Keepalived.
- Desenvolver um website dinâmico utilizando o Apache e *Hypertext Preprocessor* (PHP) para interagir com bases de dados MariaDB.
- Configurar um cluster MariaDB (Galera Cluster) para assegurar a replicação eficiente de dados entre os servidores de base de dados.
- Realizar testes de desempenho e disponibilidade para validar a eficácia da infraestrutura implementada.

2.2 Objetivos Específicos

Para além dos objetivos gerais, o projeto visa alcançar metas específicas que contribuem para a realização dos objetivos gerais. Os objetivos específicos incluem:

- Configurar o HAProxy para efetuar o balanceamento de carga de forma eficiente entre os servidores web.
- Implementar scripts de inicialização e monitorização para o Keepalived, assegurando uma resposta célere a falhas nos servidores HAProxy.
- Desenvolver páginas web dinâmicas que interajam com a base de dados MariaDB para permitir a adição e remoção de utilizadores através de formulários.

- Estabelecer e configurar devidamente o Galera Cluster para MariaDB, garantindo uma replicação fiável dos dados.
- Conduzir testes de carga para avaliar a capacidade do sistema em lidar com múltiplas requisições simultâneas e garantir a sua disponibilidade contínua.

Estes objetivos foram estabelecidos para criar uma infraestrutura robusta, capaz de lidar com a carga variável, garantindo alta disponibilidade e desempenho para os utilizadores finais.

Capítulo 3

Arquitetura do Sistema

Contents

3.1 Visão Geral da Arquitetura

Neste capítulo, será apresentada uma visão geral da arquitetura do sistema. Será fornecido um diagrama esquemático para visualizar a disposição e interconexão dos principais componentes da arquitetura.

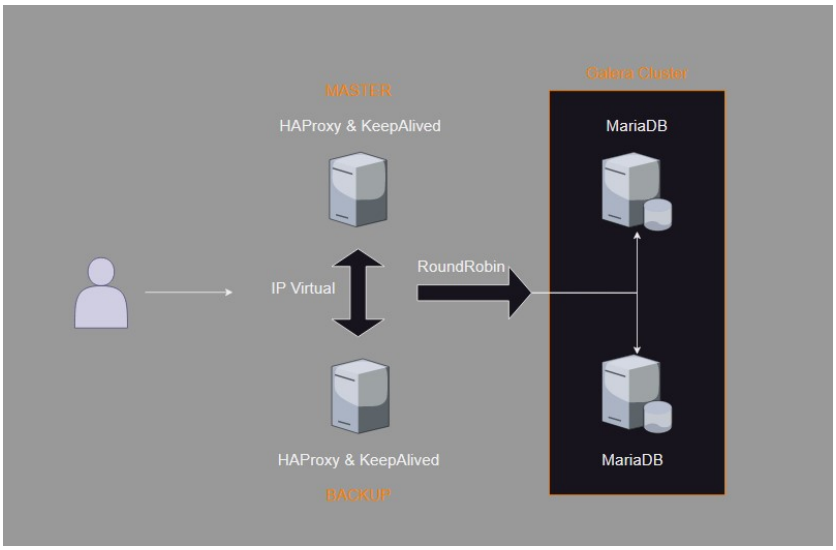

Figura 3.1: Diagrama da Arquitetura do Sistema.

A Figura 3.1 ilustra a arquitetura em termos de componentes e fluxos de dados. A seguir, será fornecida uma explicação detalhada da função de cada componente:

- **HAProxy:** Componente responsável pelo balanceamento de carga entre os servidores web. Distribui de forma eficiente os pedidos dos clientes pelos servidores disponíveis, contribuindo para a otimização do desempenho.
- **Keepalived:** Ferramenta essencial para garantir a alta disponibilidade do sistema. Fornece mecanismos de deteção de falhas e failover, assegurando que, em caso de falha em um dos servidores HAProxy, o outro assume sem interrupção.
- **Apache e PHP:** Servidor web e linguagem de programação, respectivamente, utilizados para desenvolver páginas web dinâmicas. O Apache

serve como o ponto de entrada para as requisições dos clientes, enquanto o PHP processa dinamicamente os conteúdos.

- **MariaDB:** Sistema de gestão de bases de dados relacionais, configurado como um cluster (Galera Cluster). Garante a replicação síncrona dos dados entre os servidores, contribuindo para a consistência e disponibilidade das bases de dados.

Capítulo 4

Configuração dos Servidores HAProxy e KeepAlived

Contents

4.1 HAProxy

O HAProxy é um software de balanceamento de carga que atua como um ponto de entrada para as solicitações dos clientes, distribuindo-as entre vários servidores para garantir um desempenho otimizado e uma alta disponibilidade do sistema. [7]

É importante destacar algumas considerações fundamentais sobre o HAProxy:

- **Balanceamento de Carga:** O HAProxy utiliza algoritmos de balanceamento, sendo o round-robin o padrão. Este capítulo irá explicar como essa lógica é aplicada.

- **Configuração Flexível:** A configuração do HAProxy é altamente flexível, permitindo ajustes precisos conforme as necessidades do sistema.

- **Monitorização de Servidores:** O HAProxy pode monitorizar a saúde dos servidores e ajustar dinamicamente o balanceamento para evitar sobrecargaas nos servidores.

4.1.1 Algoritmos de Balanceamento

Os algoritmos de balanceamento de carga utilizados desempenham um papel fundamental na determinação de qual servidor, dentro de um backend, será selecionado durante o processo de balanceamento. O HAProxy oferece várias opções para algoritmos, permitindo uma adaptação flexível às necessidades específicas do sistema. Além do algoritmo de balanceamento, os servidores podem ser associados a um parâmetro de peso para influenciar a frequência com que são selecionados em comparação com outros servidores.[2]

Alguns dos algoritmos mais utilizados são:

- **Round Robin:** O algoritmo Round Robin seleciona servidores em turnos. Este é o algoritmo padrão, distribuindo as solicitações de maneira equitativa entre os servidores disponíveis.

- **Least Connections:** Este algoritmo seleciona o servidor com o menor número de conexões ativas. Recomenda-se para sessões mais longas, e os servidores no mesmo backend também são rotacionados em uma forma de round-robin.

- **Source:** O algoritmo Source seleciona qual servidor usar com base em um hash do endereço IP de origem dos utilizadores que se encontram a fazer solicitações. Este método garante que os mesmos utilizadores conectam-se aos mesmos servidores, proporcionando consistência.

Esta variedade de algoritmos oferece opções para atender a diferentes cenários e requisitos de aplicação, permitindo a configuração personalizada para otimizar o desempenho e a eficiência do sistema.

4.1.2 Configuração

O arquivo de configuração do HAProxy, localizado em `/etc/haproxy /haproxy.cfg`, é estruturado em cinco secções distintas, cada uma desempenhando um papel específico na definição do comportamento do servidor, configurações padrão e interação com os clientes.[1]

- **Global:** Nesta secção inicial, são especificadas as configurações que afetam o funcionamento do processo em um nível mais baixo, relacionado diretamente com o sistema operativo.

- **Defaults:** Embora não seja obrigatória, a secção `defaults` permite reduzir a duplicação de comandos. As configurações aqui definidas são aplicadas nas secções `frontend` e `backend`, proporcionando uma maneira eficiente de centralizar configurações comuns.

- **Listen:** A secção `listen` possibilita a combinação simultânea de configurações `frontend` e `backend`. Essa abordagem é útil para redirecionamentos, especialmente para o `endpoint` de estatísticas.

- **Frontend:** Aqui, definimos como as solicitações dos utilizadores serão encaminhadas para o backend. Esta secção trata da interação inicial

com os clientes, direcionando as solicitações para o processamento apropriado.

- **Backend:** A secção `backend` concentra-se na definição dos servidores web que irão operar na infraestrutura. Além disso, é aqui que é especificado o algoritmo de balanceamento de carga a ser utilizado, moldando como as solicitações serão distribuídas entre os servidores disponíveis.

Essa estrutura modular proporciona uma abordagem organizada e escalável para configurar o HAProxy, garantindo uma gestão eficaz das diversas facetas do seu funcionamento. As Figuras 4.1 e 4.2 ilustram a configuração utilizada no projeto sendo semelhante para os dois servidores HAProxy.

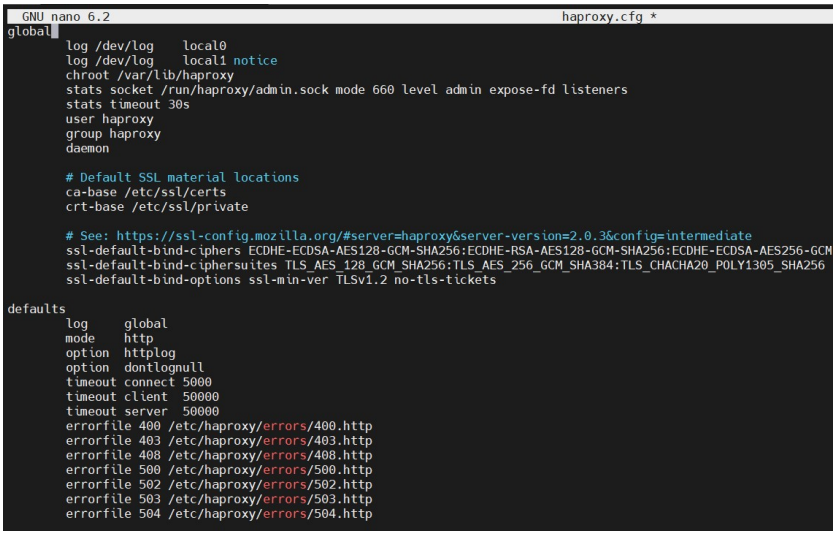

Figura 4.1: Configuração HAProxy golbal e defaults.

Conforme configurado no HAProxy, ao aceder a página web disponível em `http://192.168.1.222:8080/stats`, tem-se acesso a uma interface que exibe diversas estatísticas detalhadas sobre o desempenho dos servidores web.

A Figura 4.3 apresenta uma captura de tela dessa interface, proporcionando uma representação visual das estatísticas disponíveis.

```
#stats
listen stats

        bind 192.168.1.222:8080
        mode http
        option forwardfor
        option httpclose
        stats enable
        stats show-legends
        stats refresh 5s
        stats uri /stats
        stats realm Haproxy\ Statistics
        stats auth bduser:1qazZAQ!            #Login User and Password for the monitoring
        stats admin if TRUE
        default_backend galera_cluster_backend


# Galera Cluster Frontend configuration
frontend galera_cluster_frontend
    bind 192.168.1.222:80
    mode http
    default_backend galera_cluster_backend

# Galera Cluster Backend configuration
backend galera_cluster_backend
    balance roundrobin
    server database01 192.168.1.212:80 check inter 500 fall 3
    server database02 192.168.1.213:80 check inter 500 fall 3
```

Figura 4.2: Configuração HAProxy listen stats, frontend e backend.

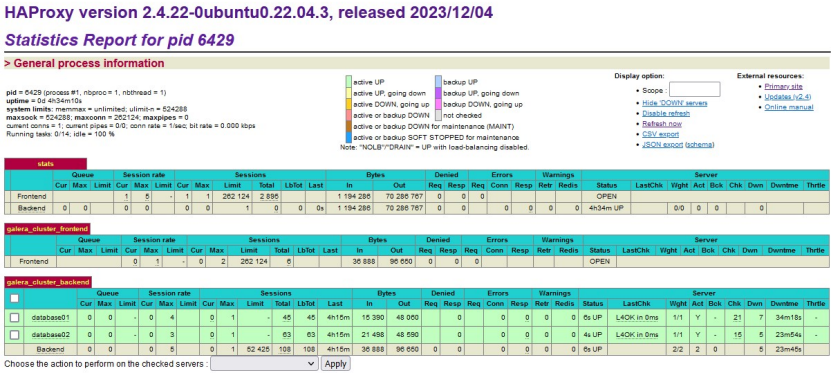

Figura 4.3: Interface de Estatísticas do HAProxy.

4.2 Keepalived

O Keepalived é uma ferramenta essencial para garantir alta disponibilidade e resiliência em ambientes de servidores. Atuando em conjunto com o HAProxy, o Keepalived desempenha o papel crucial de manter um endereço IP virtual, assegurando a continuidade dos serviços mesmo em situações de falha de um dos servidores. [4]

É relevante destacar algumas características fundamentais do Keepalived:

- ***Virtual IP Adress* (VIP):** O Keepalived é responsável por manter um VIP que é associado ao serviço. Em caso de falha de um dos

servidores, o Keepalived transfere automaticamente o VIP para o servidor operacional, garantindo a continuidade do serviço sem interrupções perceptíveis pelos utilizadores.

- **Deteção de Falhas:** O Keepalived monitoriza constantemente o estado dos servidores. Se uma falha for detetada, o VIP é transferido para um servidor operacional, garantindo a continuidade do serviço.
- **Configuração Simples:** Assim como o HAProxy, o Keepalived oferece uma configuração flexível e simples. As opções de configuração permitem ajustar o comportamento do Keepalived de acordo com as necessidades específicas do ambiente.

4.2.1 Configuração do Keepalived

No arquivo de configuração do Keepalived (localizado em `/etc/keepalived/keepalived.conf`), estabeleceu-se um `vrrp_script` com a finalidade de verificar a operação eficaz do HAProxy, com intervalos de 2 segundos, como mostra a Figura 4.4. Em caso de falha do HAProxy, entra em ação uma estratégia dinâmica, reduzindo o seu peso em 2 e, consequentemente, ajustando sua prioridade.[5]

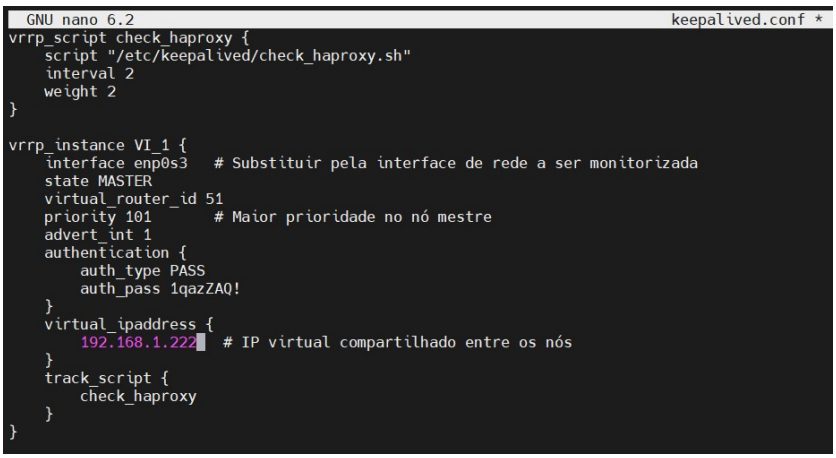

Figura 4.4: Interface de Estatísticas do HAProxy.

Após essa configuração, deu-se origem a um `vrrp_instance` que define

uma instância exclusiva do protocolo *Virtual Router Redundancy Protocol* (VRRP), incorporando diversos atributos para a gestão eficiente do sistema.

A configuração foi semelhante nos dois servidores, sendo que no segundo se alterou a prioridade e o estado para definir o mesmo como BACKUP

Para além disso foi desenvolvido o script `/etc/keepalived/check_haproxy.sh` para verificar regularmente o estado operacional do HAProxy. Este script, em Bash, retorna 0 se o HAProxy estiver a funcionar corretamente e 1 em caso de falha.

A execução periódica deste script é crucial para que o Keepalived ajuste dinamicamente as configurações, assegurando alta disponibilidade e confiabilidade na distribuição de tráfego. Posteriormente, concedeu-se permissão de execução ao script com o comando `sudo chmod +x /etc/keepalived/check_haproxy.sh`. Essa permissão é essencial para a execução adequada do script pelo Keepalived.

```
#!/bin/bash

# Este script deve retornar 0 se o HAProxy estiver funcionando corretamente e 1 em caso de falha

if systemctl is-active --quiet haproxy; then
    exit 0
else
    exit 1
fi
```

Figura 4.5: Script para verificar o estado do HAProxy.

4.2.2 Integração com o HAProxy

A integração entre o Keepalived e o HAProxy é crucial para garantir uma solução completa de alta disponibilidade. Enquanto o HAProxy distribui o tráfego entre os servidores web, o Keepalived assegura que o serviço seja ininterrupto, mesmo em situações de falha de um dos servidores.

Capítulo 5

Desenvolvimento do Website

Contents

5.1 Tecnologias Utilizadas

O website foi criado utilizando tecnologias web convencionais:

- **PHP:** Utilizado para desenvolver o backend, para a interação com a base de dados.
- **HTML e CSS:** Utilizados na construção da interface do utilizador no frontend.

5.1.1 Funcionalidade do Website

O website é uma aplicação simples que permite a criação e remoção de utilizadores. Os utilizadores podem inserir informações como nome, idade e *Número de Identificação Fiscal* (NIF). Esses dados são então processados pelo *Hypertext Preprocessor* (PHP), que interage com a base de dados.

A interface amigável, desenvolvida em HTML e estilizada com CSS, proporciona uma experiência intuitiva aos utilizadores. A principal funcionalidade consiste na gestão de dados de utilizadores, destacando-se pela simplicidade e eficácia como ilustra a Figura 5.1.

Figura 5.1: Website Utilizado.

Capítulo 6

Base de Dados MariaDB

Contents

6.1 Configuração do Galera Cluster

A infraestrutura de base de dados do projeto é sustentada pela tecnologia MariaDB, configurada como um Galera Cluster. Este capítulo explora as principais configurações e estratégias de replicação utilizadas para assegurar a resiliência e alta disponibilidade do sistema. [6]

6.1.1 Replicação de Estado

A implementação do Galera Cluster permite a replicação síncrona de estado entre os nós. Isto significa que cada nó da base de dados mantém um estado consistente em tempo real com os outros nós, garantindo integridade e consistência dos dados em toda a infraestrutura. Esta abordagem favorece a resiliência contra falhas individuais, contribuindo para um ambiente robusto e de alta disponibilidade.[3]

Neste contexto, a replicação de estado é uma peça fundamental para manter a consistência dos dados em todos os nós do cluster. O uso desta estratégia proporciona não apenas redundância, mas também a capacidade de continuar a operar, mesmo em situações de potenciais falhas de hardware ou indisponibilidade temporária de um nó específico.

No servidor primário da base de dados (`192.168.1.212`), foi elaborado o arquivo `galera.cnf` presente em `/etc/mysql/mariadb.conf.d/galera.cnf`. Este arquivo engloba todas as configurações essenciais para a formação e ajuste do cluster. (Figura 6.1).

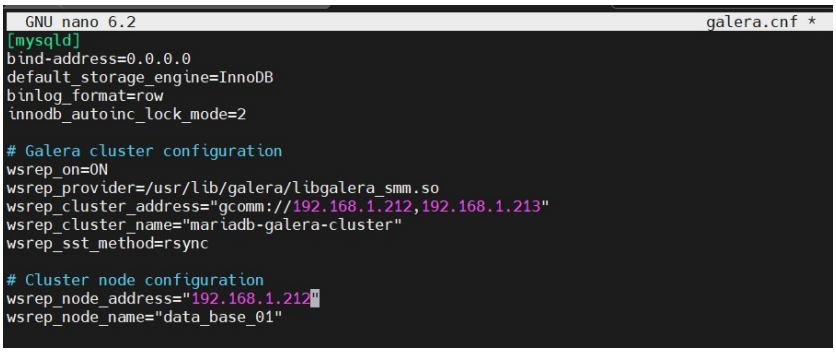

Figura 6.1: Configuração do Cluster na Primeira Base Dados.

No segundo servidor de base de dados (`192.168.1.213`), também foi elaborado o arquivo `galera.cnf`, contendo as configurações necessárias para a segunda base dados integrar-se ao cluster.

```
  GNU nano 6.2                                                    galera.cnf *
[mysqld]
bind-address=0.0.0.0
default_storage_engine=InnoDB
binlog_format=row
innodb_autoinc_lock_mode=2

# Galera cluster configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
wsrep_cluster_address="gcomm://192.168.1.212,192.168.1.213"
wsrep_cluster_name="mariadb-galera-cluster"
wsrep_sst_method=rsync

# Cluster node configuration
wsrep_node_address="192.168.1.213"
wsrep_node_name="data_base_02"
```

Figura 6.2: Configuração do Cluster na Segunda Base Dados.

Para verificar que estava tudo operacional, foi visto que o tamanho do cluster era de dois sendo que isso assegura que as duas base de dados estão integradas no cluster e prontas para fazer replicação de estado (Figura 6.3).

```
rodrigo@database02:/etc/mysql/mariadb.conf.d$ sudo mysql -u bduser -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 303
Server version: 10.6.12-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show status like 'wsrep_cluster_size';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 2     |
+--------------------+-------+
1 row in set (0.002 sec)

MariaDB [(none)]>
```

Figura 6.3: Tamanho do Cluster

Capítulo 7

Testes e Resultados

Contents

7.1 Testes de Disponibilidade e Desempenho

Neste capítulo, serão apresentados os testes realizados para verificar a disponibilidade e o desempenho do sistema desenvolvido. Serão abordados diversos cenários de operação, destacando a robustez e eficácia das soluções implementadas, como o HAProxy e o Galera Cluster, garantindo assim a integridade e eficiência do ambiente distribuído.

Para serem feitas algumas experiências foi criado um ambiente com várias máquinas virtuais.

- **node1** - 192.168.1.212
- **node2** - 192.168.1.213
- **HAProxy1 (MASTER)** - 192.168.1.214
- **HAProxy2 (BACKUP)** - 192.168.1.215
- **IP-Virtual** - 192.168.1.222
- **Cliente** - 192.168.1.233

7.2 Teste de Conectividade com HAProxy (Round Robin)

Ao aceder ao endereço *Virtual IP Adress* (VIP) 192.168.1.222, foram realizadas capturas de tela (Figura 7.1 e Figura 7.2) que evidenciam a alternância eficaz entre os servidores **node1** (192.168.1.212) e **node2** (192.168.1.213).

Na primeira experiência, onde todos os elementos do sistema estavam interligados, observou-se a seguinte interação. O cliente, identificado pelo endereço *Internet Protocol* (IP) 192.168.1.233, iniciou um pedido *Hypertext Transfer Protocol* (HTTP) para o VIP 192.168.1.222. Consequentemente, o servidor HAProxy1 (192.168.1.214), a operar como mestre (*MASTER*), encaminhou este pedido HTTP para o node1, cujo endereço IP é 192.168.1.212.

Figura 7.1: Acesso ao Node 1 pelo ip-virtual

Figura 7.2: Acesso ao Node 2 pelo ip-virtual

É importante notar que, devido ao protocolo de balanceamento *Round Robin* configurado, quando o cliente realizou um segundo pedido HTTP, o servidor HAProxy1 redirecionou a solicitação para o node2 (192.168.1.213). Este comportamento demonstra a eficácia do balanceamento de carga, alternando entre os servidores web disponíveis, otimizando assim a distribuição do tráfego (Figura 7.3).

Este cenário reflete a funcionalidade adequada do sistema quando todos os componentes estão ativos e o HAProxy1 assume o papel de mestre, enquanto o HAProxy2 permanece em modo de espera (*BACKUP*).

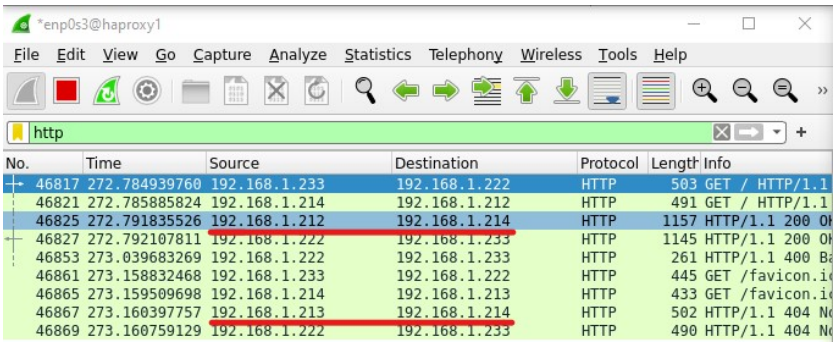

Figura 7.3: Experiência 1 - Round Robin

7.3 HAProxy (MASTER) Failover

Para avaliar a robustez do sistema, retirou-se o servidor HAProxy configurado como mestre (MASTER) de operação. O objetivo foi verificar como o sistema reagiria à falha do servidor principal. O resultado foi um processo de Failover eficaz, onde o servidor HAProxy configurado como backup (BACKUP) assumiu automaticamente as responsabilidades de mestre (MASTER) como mostra a Figura 7.4.

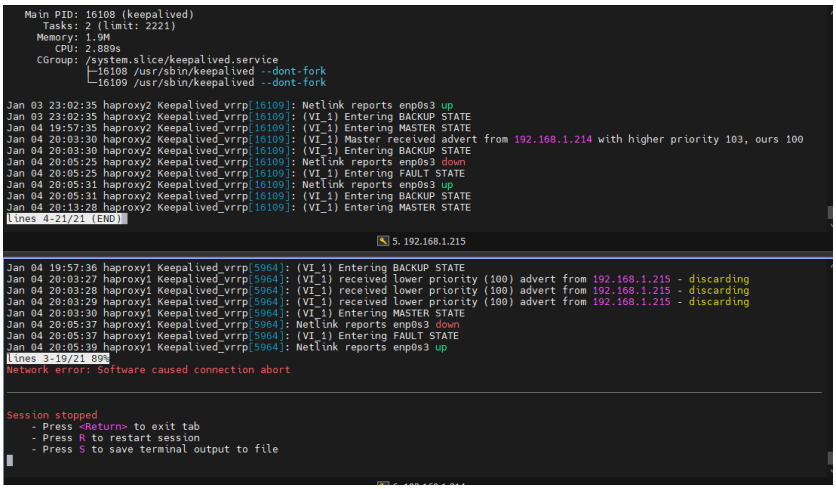

Figura 7.4: HAProxy02 assume o papel de (MASTER)

Durante esse processo, o VIP permaneceu acessível, realizando ainda a distribuição de tráfego entre os servidores web de acordo com o protocolo Round Robin previamente configurado (Figuras 7.5 e 7.6). Isso evidencia a

resiliência do sistema, assegurando a continuidade do serviço, mesmo em face de falhas inesperadas.

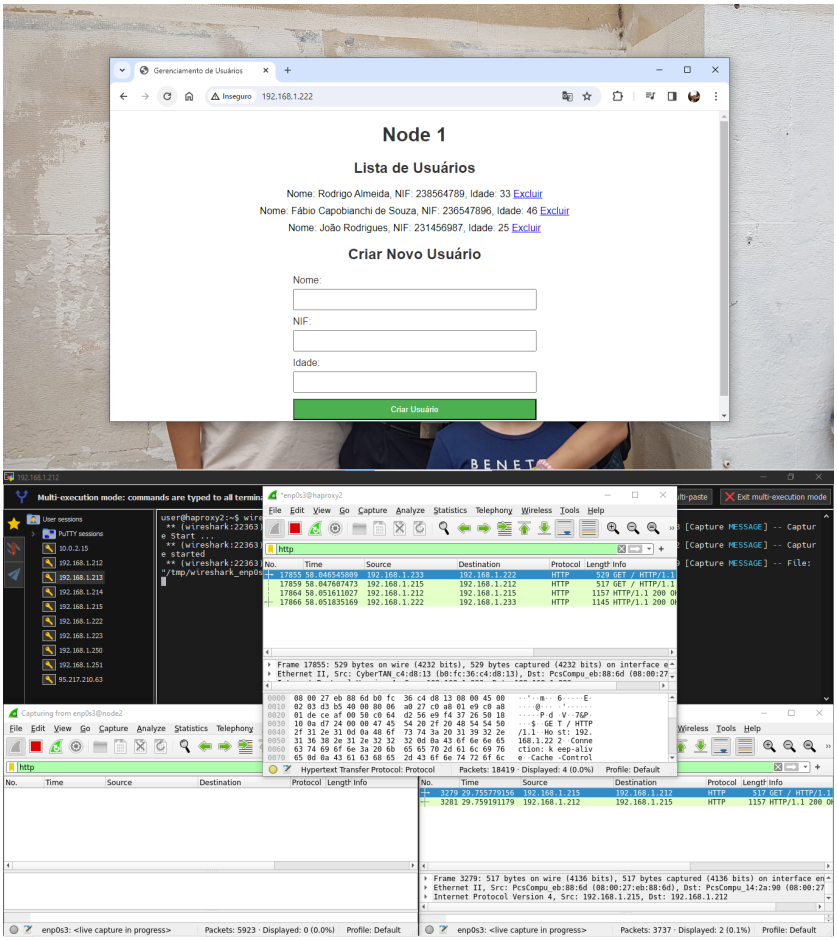

Figura 7.5: HAProxy02 para Node1

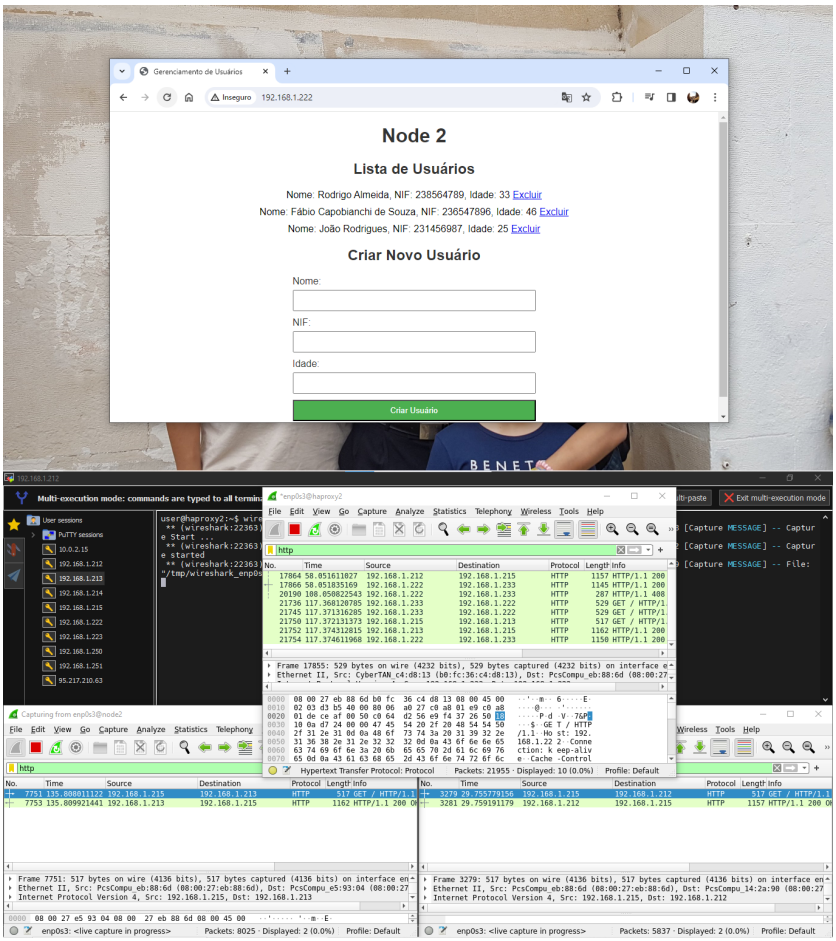

Figura 7.6: HAProxy02 para Node2

7.4 Retoma do HAProxy (MASTER)

Após testar que o servidor HAProxy2 consegue operar como mestre (*MASTER*) tendo ocorrido o failover do HAProxy1 , é crucial testar o que acontece após o mesmo voltar ao Ativo como mostra a Figura 7.7.

Como esperado, o HAProxy1 retoma o papel de mestre (*MASTER*) sendo que o HAProxy2 retoma o papel original de repouso (*BACKUP*). Ao longo desse cenário, o website permaneceu constantemente disponível, destacando a eficácia da configuração de failover.

Esse teste não apenas valida a robustez da configuração de failover, mas

também destaca a importância de se ter um sistema que se possa adaptar dinamicamente a falhas inesperadas, garantindo uma experiência contínua para os utilizadores finais.

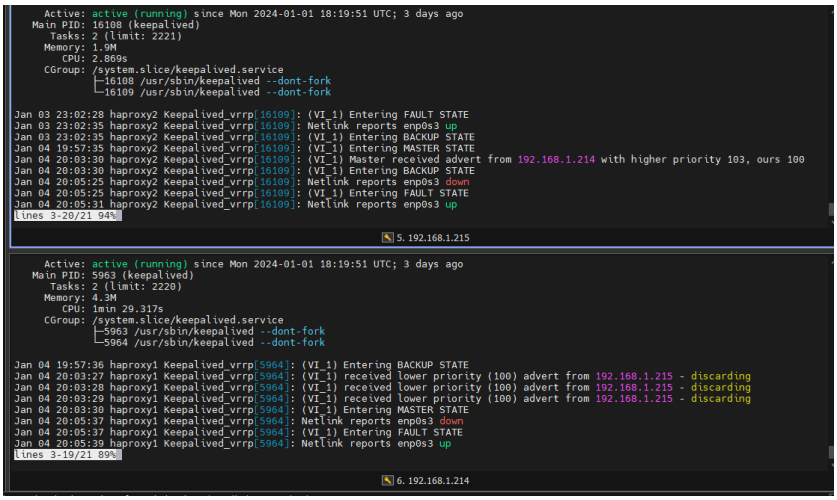

Figura 7.7: HAProxy1 volta a assumir o papel de (MASTER)

Capítulo 8

Conclusões e Lições Aprendidas

Contents

8.1 Conclusões Gerais

O desenvolvimento e implementação do sistema utilizando HAProxy para balanceamento de carga revelou-se fundamental para a garantia da alta disponibilidade e fiabilidade do serviço. A capacidade de distribuir o tráfego de maneira eficiente entre os servidores web proporcionou uma experiência de utilizador consistente e otimizada.

A utilização de uma arquitetura de cluster com o MariaDB Galera Cluster também se revelou eficaz, garantindo a replicação de estado entre os servidores de base de dados. Estas configurações asseguraram a integridade dos dados.

A implementação do mecanismo de failover no HAProxy com o KeepAlived demonstrou uma grande importância na manutenção da continuidade do serviço em situações de falha. A transição automática entre os servidores MASTER e BACKUP garantiu uma operação ininterrupta, mesmo diante de eventos imprevistos.

8.2 Conceitos Aprendidos

Ao longo do desenvolvimento e dos testes realizados, foram extraídas diversas aprendizagens fundamentais. A configuração precisa e equilibrada do sistema revelou-se crucial para garantir não apenas o desempenho, mas também a estabilidade operacional. A realização de testes abrangentes e simulações de cenários de falha destacou-se como uma prática essencial, possibilitando a identificação e correção de potenciais fragilidades.

A compreensão aprofundada das tecnologias empregadas, nomeadamente o HAProxy e o MariaDB Galera Cluster, mostrou-se indispensável para uma implementação bem-sucedida de um sistema redundante tolerante a falhas.

Resumidamente, a execução deste projeto proporcionou insights valiosos sobre a complexidade e as nuances associadas à criação de infraestruturas robustas e altamente disponíveis. As lições aprendidas constituem um conhecimento crucial para projetos futuros que busquem implementar soluções similares.

Referências

[1] *A.4. Install and Configure HAProxy Red Hat Enterprise Linux 7 | Red Hat Customer Portal*. [Online; accessed 4. Jan. 2024]. Jan. de 2024. URL: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/load_balancer_administration/install_haproxy_example1 (acedido em 03/01/2023).

[2] Mitchell Anicas. «An Introduction to HAProxy and Load Balancing Concepts». Em: *DigitalOcean* (mar. de 2022). URL: https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts (acedido em 24/09/2023).

[3] bsder. «How To Configure a Galera Cluster with MariaDB on CentOS 7 Servers». Em: *DigitalOcean* (jul. de 2019). URL: https://www.digitalocean.com/community/tutorials/how-to-configure-a-galera-cluster-with-mariadb-on-centos-7-servers (acedido em 04/01/2024).

[4] Anthony Critelli. «Keepalived and high availability: Advanced topics». Em: *Enable Sysadmin* (jan. de 2023). URL: https://www.redhat.com/sysadmin/advanced-keepalived (acedido em 03/01/2024).

[5] Anthony Critelli. «Setting up a Linux cluster with Keepalived: Basic configuration». Em: *Enable Sysadmin* (jan. de 2023). URL: https://www.redhat.com/sysadmin/keepalived-basics (acedido em 03/01/2024).

[6] *Getting Started with MariaDB Galera Cluster*. [Online; accessed 4. Jan. 2024]. Jan. de 2024. URL: https://mariadb.com/kb/en/getting-started-with-mariadb-galera-cluster (acedido em 04/01/2024).

[7] Alexander S. Gillis. *Haproxy.* URL: `https://www.techtarget.com/searchnetworking/definition/HAProxy` (acedido em 24/09/2023).

Instituto Superior de Engenharia
Politécnico de Coimbra