Universidade do Minho
Mestrado Integrado em Engenharia Informática
System Deployment and Benchmarking
Relatório da Fase 1 do Projeto Prático
Grupo 7

novembro 2020

Bruno Carvalho
A83851

Lázaro Pinheiro
A86788

Luís Cunha
A84244

Gonçalo Almeida
A84610

Gonçalo Ferreira
A84073

Conteúdo

Lista de Figuras

1 Introdução

O presente relatório desenvolve-se no âmbito da Unidade Curricular *System Deployment and Benchmarking*, tendo como objectivo expor o trabalho realizado nesta fase (seleção, caracterização e análise de uma aplicação distribuída), perspetivando uma estrutura fundamentada para a fase seguinte (automatização do processo de *deployment* e de *benchmarking*).

Optou-se por seleccionar a aplicação *Mattermost*, que é uma plataforma de mensagens instantâneas segura *open-source*, direcionada ao trabalho de equipa, oferecendo a todos os utilizadores facilidade para conversar em *chats* (privados ou públicos). O ponto forte desta aplicação é a possibilidade de hospedagem e centralização da plataforma de comunicação, de forma privada, numa *NAS* local, permitindo o aumento do controlo, da disponibilidade, da confidencialidade e um alto potencial de armazenamento. Os utilizadores podem interagir com o sistema através das aplicações *desktop*, *mobile* e da *webapp*.

Como era requisito do enunciado, este relatório comporta a descrição da arquitectura e componentes da aplicação, padrões de distribuição usados e formas de comunicação, a descrição dos pontos de configuração e a identificação de operações críticas, que representam possíveis *bottlenecks* de desempenho.

Com a realização deste trabalho prático, o grupo objectiva uma melhor compreensão do processo de *benchmarking* e de *deployment*.

2 Arquitetura e Componentes da Aplicação

A *Mattermost*, na sua generalidade, consiste num servidor Go exposto como um servidor *Restful JSON* ligado a uma base de dados *SQL* e a um serviço de armazenamento de ficheiros, com clientes Javascript e Go. Um modelo desta arquitetura pode ser visto na Figura 2.

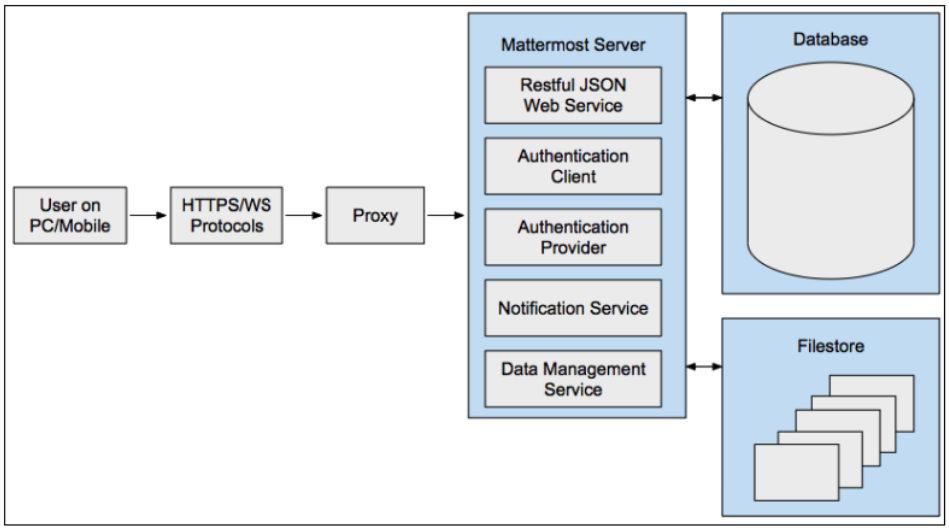

Figura 2: Arquitetura da Aplicação [1]

2.1 Servidor

O servidor é instalado através de um único ficheiro binário e pode ser configurado através do ficheiro *config/config.json*. Esta configuração pode ser feita diretamente no ficheiro, ou através de uma interface *web* (*System Console*). Este pode ser acedido através de uma *RESTful API*.

Tal como está representado na Figura 2, o servidor contém vários componentes que são acedidos a partir da API. O *authentication client* fornece serviços de autenticação aos utilizadores através de *e-mail* e palavra-passe (a *Enterprise Edition* tem mais soluções). O *authentication provider* permite que a autenticação seja feita a partir de outros serviços, tais como o GitLab. O *notification service* está encarregue de gerir as notificações e, por fim, o *data management service* está ligado às bases de dados e *filestores* de forma a controlar o acesso aos dados.

Se a aplicação estiver configurada com a *Enterprise Edition*, pode usufruir de servidores em modo *cluster*. Esta abordagem permite por um lado, que a latência de acesso seja minimizada por se colocarem servidores fisicamente mais perto dos clientes, e por outro, que seja possível fazer balanceamento de carga entre servidores, assim como o *handoff* de tráfego entre servidores em cenários de falha. [3]

2.2 Base de Dados

A base de dados (*MySQL* ou *PostgreSQL*) é responsável pelo armazenamento de dados do sistema e pela execução de pesquisas de texto.

Em ambientes empresariais (*Enterprise Edition)*, a base de dados pode possuir várias réplicas de leitura e consulta (*queries*). As réplicas de leitura podem, por exemplo, permitir que a base de dados *master* encaminhe operações para as réplicas em caso de falha de forma a que esta não tenha um impacto significativo no funcionamento da aplicação. As réplicas de consulta podem ser configuradas de forma a que, por exemplo, cada uma apenas lide com um determinado tipo de *query*. [4]

2.3 Armazenamento de Ficheiros

O armazenamento de imagens e ficheiros pode ser configurado diretamente no servidor *Mattermost*, num servidor *NAS* (*Network-Attached Storage*) ou num serviço externo de armazenamento de ficheiros, como o *Amazon S3*, dependendo das necessidades dos utilizadores e da escala da implementação. [5]

2.4 Servidor *Proxy*

É aconselhado o uso de um servidor *proxy* entre o cliente e o servidor principal de forma a fornecer:

- Melhor segurança, pois pode gerir as comunicações SSL com o servidor.
- Melhor desempenho através de técnicas de balanceamento de carga entre múltiplos servidores.
- Monitorização de tráfego de dados, tais como transferências de ficheiros.

Na Figura 3 encontra-se um exemplo de uma implementação em modo de alta disponibilidade, disponível na *Enterprise Edition*.

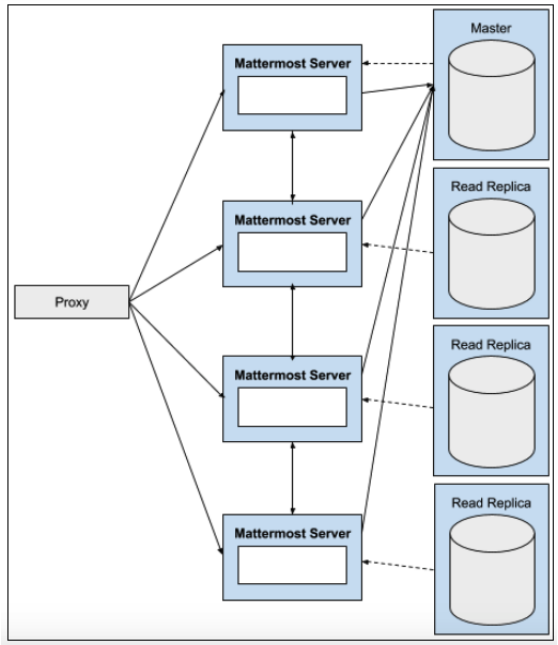

Figura 3: Exemplo de uma Organização de Alta Disponibilidade [2]

3 Distribuição

3.1 Padrões de Distribuição

Esta aplicação pode assumir vários padrões de distribuição, dependendo da forma como é configurada.

Como podemos observar pela Figura 2, caso se opte por não usar um servidor *proxy*, o cliente comunica diretamente com o servidor *Mattermost* através dos protocolos de comunicação. Neste caso, a aplicação segue uma arquitetura *client-server*.

Caso contrário, podemos utilizar uma arquitetura *proxy server*, em que os clientes comunicam com um servidor *proxy* e este comunica com os diversos servidores *Mattermost*. Neste caso, não há comunicação direta entre cliente e servidor.

Na *Enterprise Edition*, existe suporte a uma arquitetura de servidores *Mattermost* em *cluster*. Tal como demonstra a Figura 3, o servidor *proxy* pode distribuir a carga para servidores *Mattermost* que se encontrem com menos carga ou que se localizem mais perto do cliente.

Há também o suporte a uma arquitetura de réplicas de leitura do servidor da base de dados. Na Figura 3, podemos ver o servidor da base dados *master* e as várias réplicas. Desta forma a carga é distribuída e, espalhando as réplicas por diversos pontos geográficos, também conseguimos diminuir a latência das comunicações.

Em geral, a escalabilidade do sistema segue uma arquitetura *service oriented*, uma vez que escalonamos de forma separada os diversos tipos de entidades (*i.e.* armazenamento, bases de dados e servidores).

3.2 Formas de Comunicação

A aplicação é compatível tanto com HTTPS (*Secure HyperText Transfer Protocol*) como com WSS (*Secure WebSocket*). No entanto, existem diferenças na utilização dos dois protocolos.

Uma ligação HTTPS com o servidor fornece funcionalidades básicas e a capacidade de fazer *render* de páginas. No entanto, não suporta a interatividade em *real-time* disponibilizada pelo WSS. Caso não seja possível estabelecer uma ligação HTTPS, a aplicação não funcionará. Uma configuração HTTP pode ser utilizada para testes iniciais, mas não é aconselhada em produção.

Uma ligação WSS com o servidor fornece atualizações e notificações em *real-time*. Caso este tipo de ligação não esteja disponível, e o HTTPS esteja, o sistema continuará a funcionar sem as funcionalidades há pouco referidas, *e.g.*, as páginas só serão atualizadas com um *refresh* [6].

4 Pontos de Configuração

As configurações do servidor do *Mattermost* encontram-se definidas num ficheiro de configuração *mattermost/config/config.json*. Este ficheiro pode ser modificado através de um editor de texto ou da consola do sistema, desde que tenha permissões para tal, o que implica o *reload* deste.

O ficheiro anteriormente referido, é gerado através do código localizado em *mattermost-server/config/config_generator* que recorre ao modelo de configurações presente em *mattermost-server/model/model.go*

Para qualquer definição não definida em *config.json*, o servidor do Mattermost utiliza os valores default documentados em [7].

Algumas das principais configurações são:

- Web Server:
  - SiteURL: define o URL que os utilizadores utilizam para aceder ao Mattermost. É necessário indicar o número da porta caso esta não seja uma porta *standard* como 80 ou 433. Um exemplo de um valor desta configuração é "https://example.com/company/mattermost".
  - ListenAddress: define o endereço e porta para se conectar e ouvir. Ao atribuir o valor ":8056"vão ser conectadas todas as interfaces

de rede. Ao atribuir o valor "127.0.0.1:8065"vai apenas ser conectada a interface de rede com o IP descrito.

  - Outras configurações: Forward80To443, ConnectionSecurity, TLSCertFile, TLSKeyFile, UseLetsEncrypt, LetsEncryptCertificateCacheFile, ReadTimeout, WriteTimeout, IdleTimeout, EnableAPIv3, WebserverMode, EnableInsecureOutgoingConnections, ManagedResourcePaths.

- Base de Dados:

  - DriverName: define o *driver* da base de dados e pode assumir os valores "mysql"e "postgres".

  - DataSource: define a *connection string* da base de dados principal. Quando o *DriverName* assume o valor "postgres", a forma da *string* é "postgres://mmuser:password@localhost:5432/mattermost_test?sslmode=disable&connect_timeout=10". Quando o *DriverName* assume o valor "mysql", a forma da *string* é "mysql://mmuser:password@localhost:5432/mattermost_test?sslmode=disable&connect_timeout=10".

  - Outras configurações: MaxIdleConns, MaxOpenConns, QueryTimeout, DisableDatabaseSearch, ConnMaxLifetimeMilliseconds, MinimumHashtagLength, AtRestEncryptKey, Trace.

- Armazenamento de Ficheiros

  - DriverName: define o *driver* do sistema de armazenamento de ficheiros e pode assumir os valores "local"(valor *default*) e "amazons3".

  - Directory: define a diretoria em que os ficheiros são escritos. Quando o *DriverName* assume o valor "local", o valor desta configuração é relativo à diretoria onde o Mattermost está instalado. Quando o *DriverName* assume o valor "amazons3", o valor *default* desta configuração é "./data/".

  - MaxFileSize: define o tamanho máximo dos ficheiros (em megabytes) guardados na *System Console UI*.

- Servidor Proxy
  - Enable: define se um *image proxy* está ativo para imagens externas de modo a preveni-las de se conectarem diretamente aos servidores remotos e pode assumir os valores "true"e "false".
  - ImageProxyType: define o tipo do *image proxy* utilizado e pode assumir os valores "local"(o próprio servidor Mattermost serve de *image proxy*) e "atmos/camo"(é utilizado um *atmos/camo image proxy* externo).
  - RemoteImageProxyURL: define o URL do *atmos/camo proxy*.
  - RemoteImageProxyOptions: define a URL *signing key* passada a um *atmos/camo proxy*.

5 Operações Críticas

5.1 Comunicação cliente-servidor

Através da Figura 2 podemos observar que a comunicação cliente-servidor e vice-versa é balanceada através de um *proxy*. Ora, visto que toda a comunicação passa por este componente, torna-se um ponto único de falha, ou seja, uma falha compromete o funcionamento sistema em geral.

Os serviços de autenticação que o servidor fornece (*authentication client* e *provider*) devem também estar sempre disponíveis, uma vez que tratam da autenticação dos utilizadores. Sem estes serviços ativos, não lhes é possível acederem ao resto do sistema.

De modo a viabilizar as características anteriormente mencionadas, é proposto um modelo de *Cluster* para o *Proxy*. Seguindo o modelo Servidor Proxy (consultar 2.4) é possível usar diversas tecnologias como *NGINX proxy* ou *Apache 2*.

Esta solução oferece diversos mecanismos como subdivisão de tarefas, e para solucionar o problema de falha usa-se um método *Active-Passive*. Neste o agente *Passive* garante se houver alguma falha com o servidor *Active* encarregar-se de substituir a sua função sem que haja algum tipo de quebra no serviço.

A autenticação aproveita este modelo de comunicação para balancear a sua carga por um *Cluster* de Mattermost servers, como é possível observar na Figura 3, assim proporcionando uma alta disponibilidade do serviço de autenticação e não só.

5.2 Base de Dados e Armazenamento de Ficheiros

A disponibilidade dos dados é também um ponto crítico do sistema. Sem estes, o sistema torna-se praticamente inútil, uma vez que os utilizadores não têm acesso à informação que procuram.

Com uma arquitetura simples deste serviço, apenas um servidor base de dados, é difícil garantir a alta disponibilidade e redundância de dados como também origina ponto de falha único, tornando este inacessível. O exemplo remete também para a questão de *throughput*, que dada um súbito aumento de clientes reflete uma baixa de performance do serviço, provocando *overflow* de pedidos.

Com objetivo de solucionar este problema, foi proposto a utilização de um conjunto de bases de dados, gerido por um *master* que sincroniza a informação com as restantes replicas de leitura. Este mecanismo oferece replicação de informação, escalabilidade e reduz a latência com o consumidor final (consultar 2.2).

À semelhança do serviço base de dados, o serviço de armazenamento de ficheiros deparasse com os mesmos problemas. Face a este obstáculo propõem-se utilizar um mecanismo de copia de ficheiros entre servidores *master* e *readable*, com recurso a utensílios de sistemas de replicação dados é possível atingir a solução desejada.

6 Conclusão

Culminada a elaboração da primeira fase do trabalho prático, importa referir que a execução do mesmo permitiu aos elementos do grupo compreender a arquitetura e o modo de funcionamento da aplicação *Mattermost* que o grupo selecionou como alvo da sua análise.

A execução desta fase do trabalho prático permitiu uma melhor consolidação dos construtos teóricos implementados pelo *Mattermost*.

Ao nível de dificuldades sentidas, importa referir alguma complexidade sentida na adaptação à arquitetura da documentação fornecida pelos *developers*.

Esta fase tornou-se fulcral, dado que permitiu analisar e caraterizar a aplicação selecionada, tendo em vista, num futuro próximo, a fase de automatização do processo de *deployment* e de *benchmarking*.

No ímpeto geral o desenvolvimento desta fase do trabalho decorreu como planeado, alcançando os objetivos delineados pelo enunciado.

Referências

[1] https://docs.mattermost.com/overview/architecture.html#communication-protocols

[2] https://docs.mattermost.com/overview/architecture.html#high-availability-and-scalability

[3] https://docs.mattermost.com/deployment/deployment.html#mattermost-server

[4] https://docs.mattermost.com/deployment/deployment.html#data-stores

[5] https://docs.mattermost.com/deployment/deployment.html#file-store

[6] https://docs.mattermost.com/deployment/deployment.html#communication-protocols

[7] https://docs.mattermost.com/administration/config-settings.html