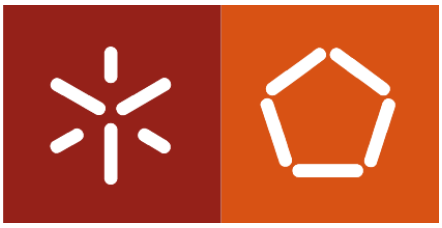

UNIVERSIDADE DO MINHO

SYSTEM DEPLOYMENT AND BENCHMARKING

OPENPROJECT

Grupo:
João Nunes - A82300
Tiago Pinheiro - A82491
Vítor Gomes - A75362
Joel Gama - A82202

Braga, Portugal
31 de Outubro de 2019

Conteúdo

1 Introdução

No âmbito da UC de *Systems Deployment and Benchmarking* foi proposta a análise e caraterização de uma aplicação distribuída *opensource*.

Como tal, nesta primeira fase do projeto foi necessário identificar a arquitetura da aplicação, os padrões de distribuição, os pontos de configuração e por fim a identificação das operações cujo o desempenho é crítico.

Como era necessário escolher uma aplicação *multi-tier*, com elasticidade em pelo menos uma das camadas e com um sistema de gestão de base de dados, foi escolhido portanto o *OpenProject*, um software de gestão de projetos.

2 Arquitetura da aplicação

A arquitetura da aplicação *OpenProject* é uma arquitetura *multitier*. Neste capitulo irão ser apresentados os principais componentes - *PostgreSQL*, *Apache2*, *phusion-passenger* e *memcached* - desta arquitetura onde assenta a aplicação.

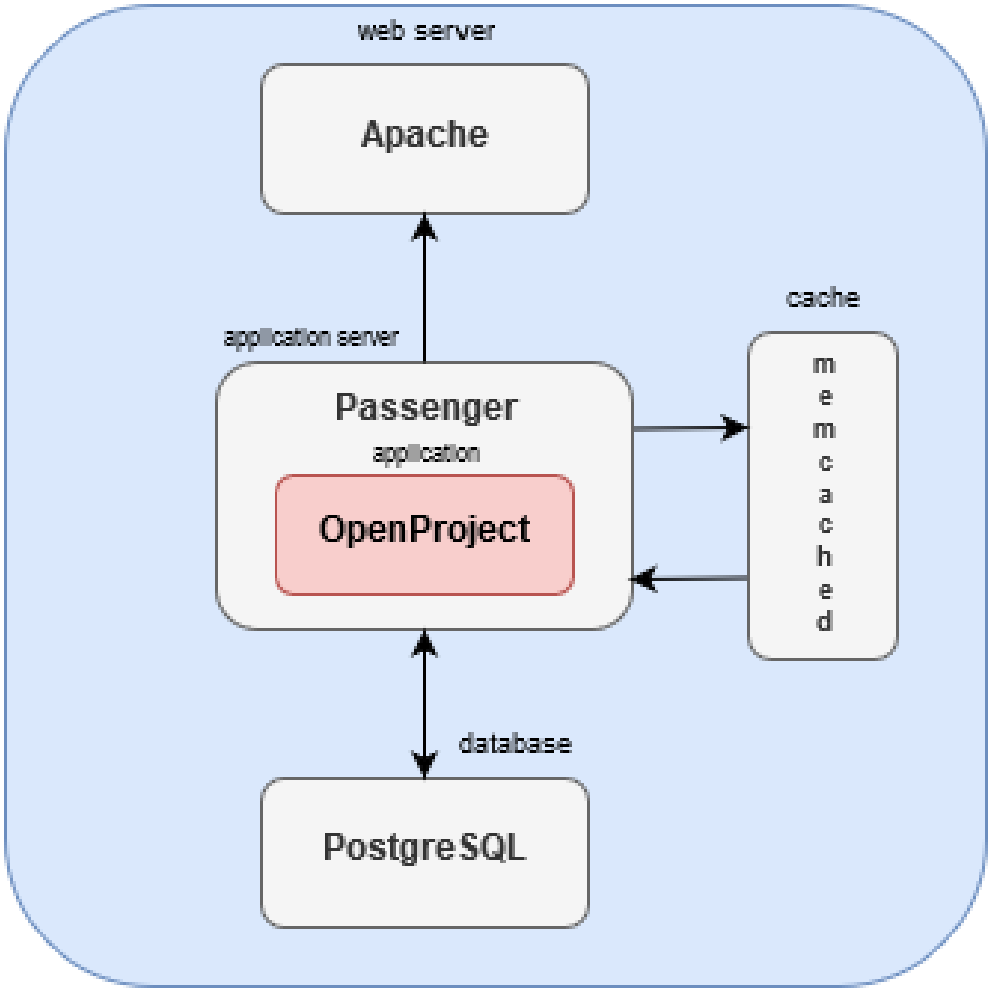

Figura 2.1: Arquitetura do *OpenProject*.

A base de dados usada, como foi dito anteriormente, é o *PostgreSQL*. Por sua vez, o *PSQL* encontra-se numa arquitetura *Master-Slave*, uma arquitetura que tem uma base de dados principal (*Master*) e um ou mais *servers* (*Slaves*) prontos para passarem a principais, caso a base de dados principal falhe.

O componente que faz a comunicação com a base de dados é o *AppServer* neste caso, o *Phusion-Passenger*. O *passenger* faz a conexão do *WebServer* com a base de dados. Para além desta função, os *AppServers* também tem a função de tratar de todas as conexões, configurações e gestão da aplicação.

A aplicação aproveita-se do sistema distribuído de *caching* em memória, o *memcached*, para aumentar a velocidade e reduzir a quantidade de acessos necessários à base de dados principal. Ele consegue isso ao manter dados e objetos em RAM. Desta forma, o sistema consegue reduzir alguns dos gastos computacionais e diminuir o tempo de resposta da aplicação a um pedido.

Para que a aplicação possa comunicar com o cliente através da rede, é necessária a existência de um *WebServer*. Uma forma de o implementar é com o *Apache*. O *Apache* tem a função de comunicar com o *AppServer* e com um cliente através do protocolo *HTTP*. A cada pedido de um cliente para o *WebServer*, este aloca uma *thread* para processar do pedido. Por sua vez, o *AppServer*, que está sempre à escuta na porta do apache, processa o pedido reencaminhado do *WebServer*. Quando o *WebServer* recebe a resposta do *AppServer*, este fecha a *thread* alocada ao pedido já foi processado, e responde ao cliente através de uma mensagem HTTP.

3 Padrões de Distribuição e Pontos de Configuração

Cada componente da aplicação será isolado dos restantes componentes implementados em máquinas diferentes. Este isolamento permite tratar do escalamento e implementação de cada componente de forma independente.

O *PostgreSQL* permite implementar uma arquitetura *master-slave* onde o nodo *master* trata de todas as escritas para a base de dados. Estas escritas são depois replicadas para os *slaves*. A replicação da base de dados para os *slaves* é realizado através de *streaming* síncrono. Os nodos *slave* podem ser definidos em *hot standby* o que significa que estão prontos para substituir o *master* em caso de falha e disponíveis para realizar queries de leitura. Este padrão de distribuição oferece alta disponibilidade, graças aos *slaves* prontos a substituir o *master* em qualquer momento, e um aumento de performance para operações de leitura pelos vários nodos disponíveis para leitura. Como a forma de replicação é síncrona, o *master* espera que os *slaves* em *hot standby* efetuem as suas escritas, isto garante que os dados não serão perdidos em caso de falha no *master*, mas incorre num aumento do tempo de escrita na base de dados.

Este tipo de distribuição é visto como o mais adequado para a aplicação *OpenProject* porque retira a possibilidade da perda de dados durante, por exemplo, a definição da *deadline* de um projeto e oferece alta disponibilidade permitindo assim o desenvolvimento de um projeto em qualquer momento.

4 Operações críticas

Operações críticas destacam-se como sendo operações que podem causar elevada lentidão ou até mesmo falha no sistema, como tal foi necessário analisar a arquitetura e o funcionamento geral do sistema de modo a identificar em que situações o desempenho e a disponibilidade do sistema podem ser postos em causa.

Uma operação crítica sobre o sistema é na base de dados. Caso esta falhe o serviço fica indisponível devido à impossibilidade de aceder aos dados da aplicação que nela estão guardados. Uma situação que causa sobrecarga na base de dados é um elevado número de operações sobre desta.

Um outro ponto crítico do sistema é o *Web Server* visto que este é responsável por atender os pedidos *HTTP* do cliente. Mais uma vez, se existir sobrecarga de pedidos neste módulo o serviço pode ficar indisponível.

Por fim, outro ponto crítico do sistema é o *App Server*. Este é responsável por realizar os pedidos à base de dados, ao *memcached* e comunica também com o *Web Server*. Tal como nos casos anteriores um excesso de pedidos ao *App Server* pode resultar numa falha do mesmo falhar.

5 Conclusão

Em suma, após uma extensa análise da aplicação *OpenProject* foi possível definir a arquitetura geral da aplicação, onde foi possível identificar os principais módulos e componentes que compõem a aplicação, de seguida foi também possível identificar um possível padrão de distribuição para a aplicação em questão, bem como os respetivos pontos de distribuição. Por fim, foram identificadas as operações críticas da aplicação.

Neste processo descrito anteriormente, o grupo sentiu mais dificuldades a identificar os padrões de distribuição e a arquitetura da aplicação. Ambas as dificuldades foram causadas tanto pela falta de informação útil no site da aplicação como no processo de instalação, um processo quase totalmente automatizado e com poucas explicações sobre as escolhas feitas.

Em suma, esta primeira fase do projeto forneceu ao grupo conhecimentos acerca dos componentes de uma aplicação real e que permitiu ao grupo analisar uma aplicação na sua completa amplitude. Na seguinte fase terá que ser feito o *deployment* tendo por base o modelo anteriormente especificado.