SCALING RAILS: A SYSTEM-WIDE APPROACH TO PERFORMANCE OPTIMIZATION

Gonçalo Santarém da Silva

Dissertação realizada sob a orientação do *Prof. Ademar Aguiar*
no *Escolinhas.pt*

1. Motivação e Objectivos

A importância da Web na vida das pessoas tem aumentado progressivamente, sendo esta utilizada tanto para fins profissionais como pessoais. A qualidade das experiências de utilização tem cada vez mais relevância e a *Web 2.0* contribuiu para o aumento das expectativas dos utilizadores quanto às suas experiências de interacção. Para ajudar ao desenvolvimento de plataformas que cumpram com as expectativas, foram criadas várias ferramentas e, entre elas, está o Ruby on Rails. Esta *framework* é bastante famosa por ser ágil, robusta e oferecer métodos e funcionalidades que melhoram significativamente a qualidade do produto final sem que o tempo de desenvolvimento aumente significativamente. No entanto, esta é frequentemente criticada por sofrer de problemas de escalabilidade.

É urgente que se criem linhas de guia gerais orientadas ao desenvolvimento de aplicações com boa escalabilidade e desempenho. As equipas de desenvolvimento procuram configurações adequadas para todos os componentes envolvidos na aplicação para que possam melhorar a sua escalabilidade e os seus tempos de resposta. É importante que se abordem todos os componentes dos quais o Rails depende e que se determine quais são as melhores alternativas para determinadas situações, bem como optimizá-los especificamente para o Rails.

Melhorar a qualidade das ferramentas de *profiling* é, também, imperativo. Esta actividade é crucial durante a optimização de aplicações mas as ferramentas necessárias não estão funcionais nas últimas versões do Rails e as versões anteriores apenas apresentavam os resultados em texto. É importante que estas ferramentas voltem a estar funcionais, sejam menos verbosas e mais intuitivas.

Por fim, é crucial que a comunidade esteja ciente da importância de desenvolver aplicações de alto desempenho. Este aspecto deve ser considerado desde o inicio do desenvolvimento e o acesso a informação que ajude ao planeamento e desenvolvimento de aplicações escaláveis deve ser facilitado.

2. Descrição do Problema

Existem comparativos entre alternativas para componentes e também existem análises de configurações. No entanto, esta informação encontra-se dispersa e muitas vezes concentra-se em componentes, arquitecturas ou objectivos específicos. A criação de um guia geral que cubra todos os componentes da perspectiva do Rails torna-se, portanto, imperativa.

As ferramentas de *profiling* do Rails nunca foram actualizadas para funcionar com o Ruby 1.9. Esta situação dificulta a análise das aplicações, bem como dificulta a adoção desta versão que tem um desempenho superior à versão anterior. A esta situação acresce ainda o facto das ferramentas nativas de análise do Ruby apenas apresentarem resultados em texto. O processo de análise de uma aplicação é essencial para optimizar o seu desempenho e, como tal, torna-se imperativo que as ferramentas de *profiling* do Ruby e do Rails sejam corrigidas, melhoradas, e que interajam sem falhas entre si, bem como é importante que se integrem novos formatos para a apresentação de resultados.

Várias plataformas baseadas nesta *framework* apresentaram problemas de escalabilidade e alguns destes tornaram-se bastante famosos. No entanto, a consciência da importância de desenvolver aplicações com bom desempenho é reduzida, dentro da comunidade. A maioria dos *plugins* e *gems* não suportam o Ruby 1.9 e o Rails 3, embora tenha sido lançado em 2008 e a API esteja disponível desde o inicio de 2009, respectivamente. A maior parte das aplicações famosas baseadas em Rails não suportam Ruby 1.9 e não começaram a actualizar para o Rails 3. Sabendo-se dos benefícios inerentes às últimas versões de ambos os componentes, é importante que se actualizem alguns *plugins* e aplicações famosas. Assim, as melhorias podem ser observadas pela comunidade, motivando a actualização das restantes aplicações, *plugins* e *gems*.

Os objectivos deste trabalho são os de criar um guia geral com a informação referida anteriormente, melhorar as ferramentas de *profiling* actuais e aumentar a sensibilidade da comunidade no que toca a este assunto. Para alcançar estes objectivos, é importante que sejam abordadas várias questões especificas a cada componente de uma perspectiva centrada no Rails.

3. Abordagem ao Problema e Resultados

Para cumprir os objectivos supracitados, todos os componentes de sistema habitualmente presentes em aplicações Rails foram abordados. As próximas secções expõe o trabalho feito e resultados obtidos em cada um destes componentes.

3.1. Sistemas Operativos

Relativamente a desenvolvimento, criou-se um *script* de análise de desempenho geral para sistemas UNIX. Na análise propriamente dita, usou-se este *script* no teste de quatro distribuições de Linux, no qual se obteve um desempenho inferior por parte do CentOS. Este foi, assim, descartado de análises futuras, seguindo-se um teste orientado ao desempenho de servidores web. Neste teste, o Gentoo obteve os melhores resultados e mostrou uma estabilidade bastante acentuada. Este foi, de seguida, comparado ao FreeBSD através de um teste de desempenho do Ruby. Voltando a mostrar os melhores resultados, o Gentoo foi escolhido como base para o trabalho futuro. Por fim, analizaram-se algumas configurações importantes do *kernel*.

3.2. Ruby

Determinaram-se, inicialmente, as diferenças de desempenho entre o YARV e o seu predecessor. Como era esperado, o desempenho do YARV mostrou-se geralmente superior ao do MRI. Quanto a desenvolvimento, fizeram-se várias actividades. Em primeiro lugar, portou-se o *Escolinhas.pt* para Ruby 1.9. Contabilizou-se e analisou-se o esforço necessário, concluíndo-se que foi bastante reduzido. Melhorou-se, ainda, a flexibilidade do GC do YARV através da introdução de cinco parâmetros de configuração. Testes iniciais mostraram que o desempenho aumenta e a utilização de memória diminui se forem utilizadas configurações diferentes. Melhorou-se o *profiler* nativo do YARV, que agora guarda mais dados, devolve informação no formato *hash* e introduziu-se um visualizador gráfico de resultados em HTML.

3.3. Servidores Web para Rails

Quanto a desenvolvimento, desenvolveu-se um *script* de monitorização de memória. Por outro lado, as análises de desempenho tiveram várias fases. Em primeiro lugar, comparou-se o desempenho de várias *proxies*, de onde se concluiu que embora obtendo resultados semelhantes, o Nginx consome menos memória. Posto isto, analisou-se o desempenho do Passenger em Apache e Nginx, onde se observou um desempenho similar mas, no entanto, um menor consumo de memória e mais estabilidade quando emparelhado com o Nginx. De seguida comparou-se o Thin ao Unicorn e o primeiro obteve resultados ligeiramente melhores. Por fim, fez-se um teste completo que envolveu o Thin, Unicorn, Passenger, Nginx e diferentes versões do Ruby. Todas as alternativas obtiveram resultados semelhantes, com o Unicorn a mostrar um desempenho ligeiramente superior e o Thin a consumir ligeiramente menos memória. No entanto, alternar as versões de Ruby teve um impacto muito grande nos resultados, onde se pode observar as melhorias de desempenho presentes no YARV. Por fim, analizaram-se algumas configurações de todos os servidores web, das quais se destacou a activação de *threads* no Thin. Fez-se um teste especial a esta opção, de onde se concluiu que não existem vantagens significativas.

3.4. Bases de Dados

Melhorou-se uma promissora biblioteca Ruby para MySQL—*mysql2*—através da introdução de *lazy type casting*. Anteriormente, a conversão era feita em simultâneo com o carregamento dos dados da base de dados.

3.5. Ruby on Rails

Em primeiro lugar, abordaram-se várias falhas comuns e porpuseram-se soluções para essas falhas. Quanto a desenvolvimento, melhoraram-se as ferramentas de análise de desempenho do Rails. Posteriormente, portaram-se alguns *plugins* e a Redmine para Rails 3. Criou-se um plugin para adicionar suporte para o cabeçalho *X-Accel-Redirect* do Nginx. Por fim, iniciou-se um projecto de integração contínua para o desempenho do Rails em si ao abrigo do *Ruby Summer of Code*. Iniciou-se, ainda, um blogue público e uma série de artigos na *Rails Magazine* sobre este assunto.

4. Conclusões

Criou-se o guia geral de optimização de desempenho de aplicações Ruby on Rails. As conclusões obtidas das várias análises aos componentes e a sua aplicação ao *Escolinhas.pt* colmataram as falhas da pesquisa inicial, culminando numa base sólida de conhecimento para a elaboração do guia referido.

Melhoraram-se e actualizaram-se as ferramentas nativas de *profiling* do Ruby e do Rails. As do Rails passaram a suportar o YARV. As do Ruby, por outro lado, passaram a guardar mais dados e a devolver e apresentar resultados em novos formatos. Assim, ambas foram melhoradas.

Houve, ainda, um contributo para o aumento da consciência da importância em desenvolver aplicações com bom desempenho e escalabilidade. Melhorar as ferramentas de *profiling*, dar flexibilidade ao GC do YARV, criar um blogue público sobre este assunto, iniciar uma série de artigos orientados ao desempenho de aplicações na *Rails Magazine* e desenvolver uma bancada oficial de testes de desempenho para o Rails são actividades que genéricamente aumentam a consciência sobre a importância deste assunto dentro da comunidade. Portar a Redmine para Rails 3 expôs as vantagens desta versão, dado que os utilizadores podem utilizar esta versão e comparar. Por fim, o facto de alguns plugins estarem, agora, compatíveis com o Ruby 1.9 e o Rails 3 diminui o esforço de porte de algumas aplicações existentes.