Um guia para Testes de Escalabilidade de Serviços Web

IME-USP

Desenvolvimento
Diego de Araújo Martinez Camarinha

Supervisores
Prof. Fabio Kon
Paulo Bittencourt Moura (Doutorando)

CCSL USP
CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE

Motivação

O Mezuro [1], plataforma de monitoramento de código-fonte livre e aberto na Internet, logo entrará em produção e, por isso, tornou-se necessário saber se o *Kalibro Web Service* [2], seu componente central, é capaz de tratar inúmeras requisições simultâneas mantendo o tempo de resposta praticamente inalterado, assim como descobrir se ele possui limitantes de desempenho. Para isso, é preciso analisar sua escalabilidade.

Por não haver consenso sobre quais métricas são relevantes nem sobre o que o resultado das medições significam, é pertinente investigar uma forma melhor de executar testes de escalabilidade e interpretar tais resultados.

Objetivos

- Realizar testes de escalabilidade no Kalibro;
- Encontrar limitantes de processamento;
- Propor e implementar soluções para os problemas encontrados;
- Refazer os testes de escalabilidade e comparar os resultados com os dos testes anteriores;
- Realizar testes de escalabilidade em uma composição de serviços web e,
- Escrever um guia para testes de escalabilidade que será resultante dos resultados obtidos.

Scalability Explorer

O Scalability Explorer [3] é um arcabouço escrito em Java, desenvolvido no contexto do projeto CHOReOS [4] para ser flexível e permitir ao usuário escrever testes automatizados de escalabilidade significativos utilizando poucas linhas de código.

As métricas que o Scalability Explorer coleta que utilizamos para avaliar a escalabilidade do Kalibro foram:

- **Degradação**: mede a performance que é perdida para uma dada configuração de sistema, conforme a carga de requisições cresce;
- ***Speedup***: diz respeito ao ganho relativo de performance quando mais recursos são adicionados ao sistema, mantendo fixa a carga de requisições recebidas e,
- **Performance agregada**: mede a performance de um sistema quando a carga de requisições cresce, enquanto os recursos do sistema também são escalados.

Estudo de Caso

Kalibro

Para poder realizar os testes de escalabilidade, primeiramente foi necessário ter o Kalibro instalado e rodando em várias máquinas. Para conseguir isso, utilizamos um computador do CCSL e, com a ajuda do OpenStack, criamos algumas máquinas virtuais de tamanhos diferentes.

Depois, usamos o Scalability Explorer para escrever testes para todas as operações do serviço. Com os resultados, identificamos três requisições cujo desempenho era muito ruim comparado às outras. Dentre elas, escolhemos a requisição de processamento de repositório para analisar, por essa ser a principal função do serviço.

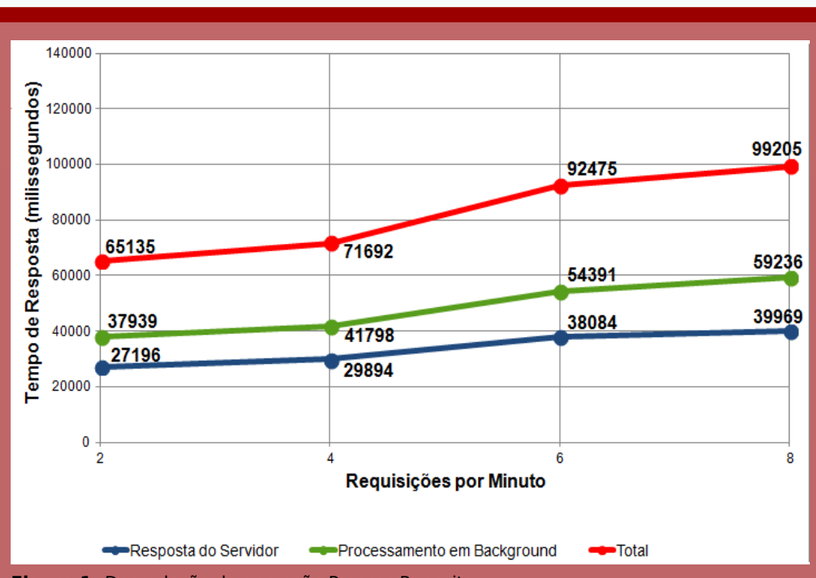


Figura 1: Degradação da operação Process Repository.

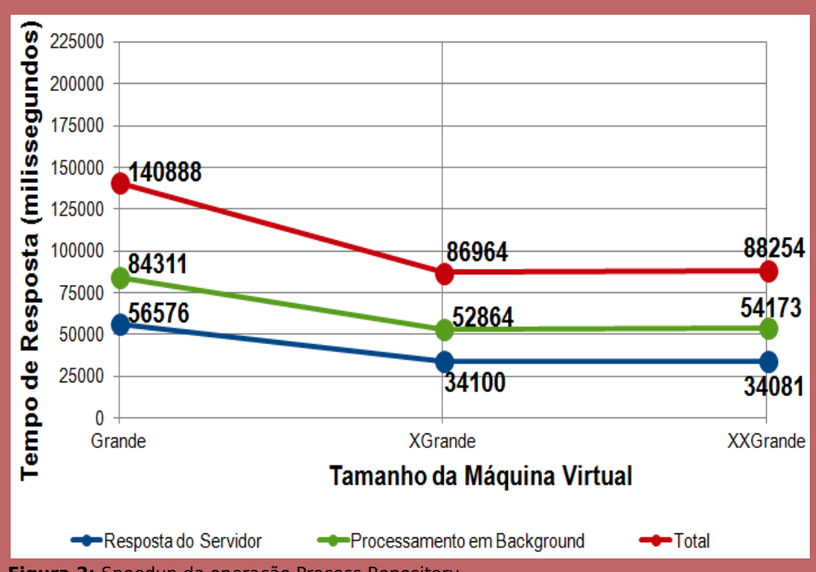


Figura 2: Speedup da operação Process Repository.

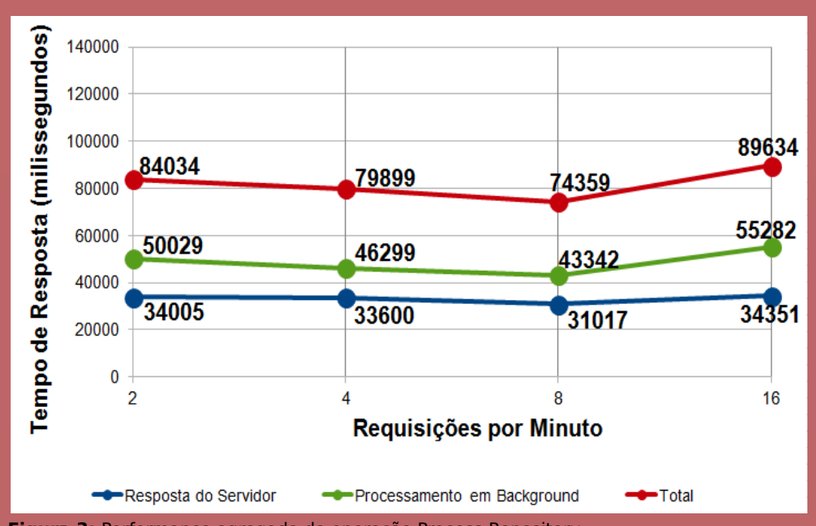


Figura 3: Performance agregada da operação Process Repository.

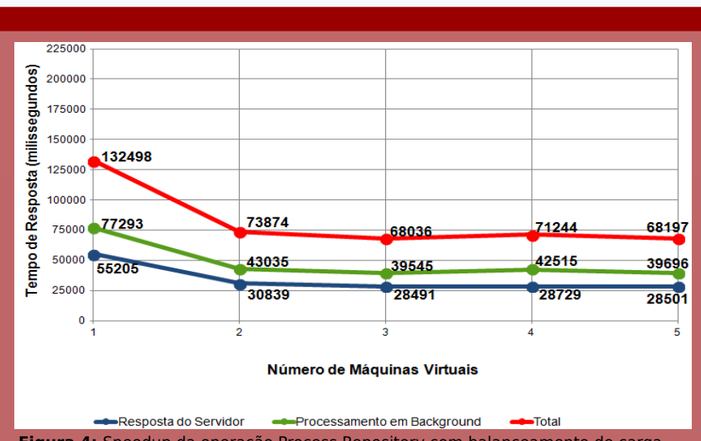


Figura 4: Speedup da operação Process Repository com balanceamento de carga.

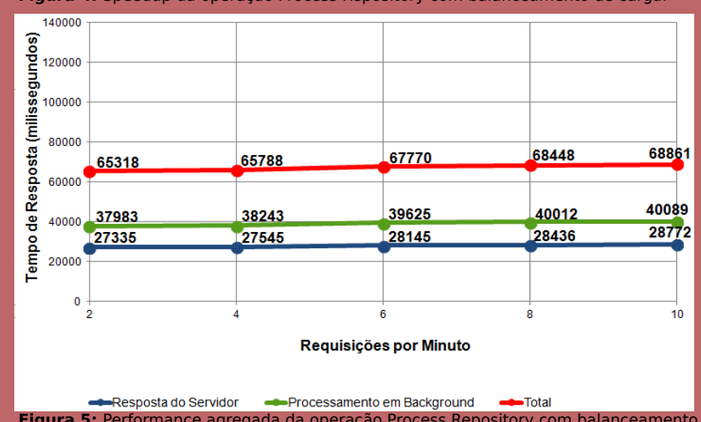


Figura 5: Performance agregada da operação Process Repository com balanceamento de carga.

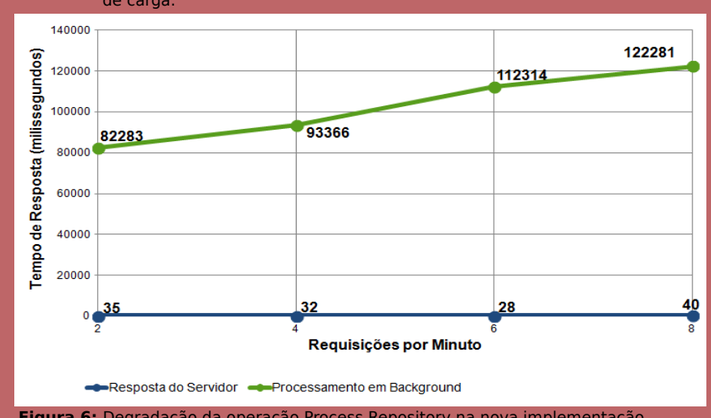


Figura 6: Degradação da operação Process Repository na nova implementação.

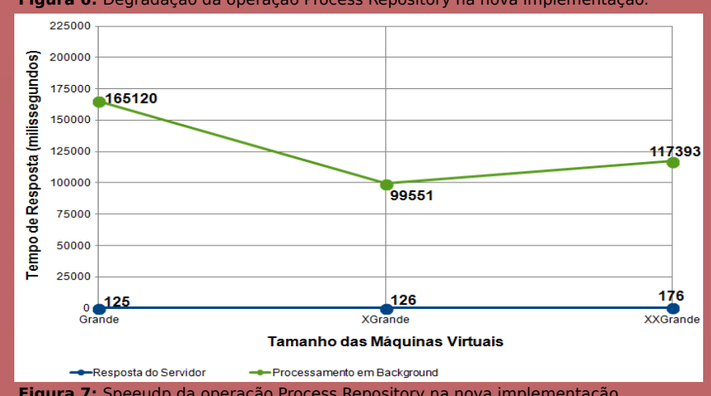


Figura 7: Speeudp da operação Process Repository na nova implementação.

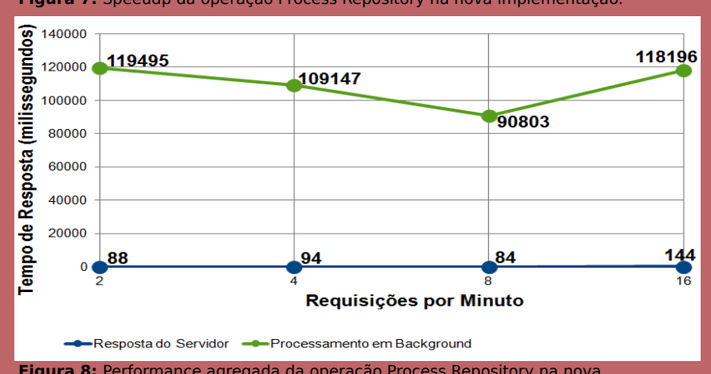


Figura 8: Performance agregada da operação Process Repository na nova implementação.

Solução 1:

Balanceamento de carga

Implementação de um balanceador de carga que possa (i) replicar o serviço em mais computadores; (ii) cadastrá-los em um computador que servirá como balanceador e, (iii) quando houverem requisições de processamento de repositório de código-fonte, distribuir a carga entre os serviços cadastrados.

Solução 2:

Mudança na infraestrutura

Melhorar o fluxo de processamento do Kalibro, de modo que o usuário do Mezuro, ao fazer requisições para o serviço, tenha um *feedback* mais rápido do estado do processamento.

Conclusão

O balanceamento de carga é uma ótima opção para manter o sistema estável em horários de picos de requisições, podendo utilizar a elasticidade da nuvem para economizar recursos.

Por sua vez, a mudança na infraestrutura, apesar de ter diminuído significativamente o *feedback* para o usuário, aumentou o tempo total de processamento da requisição.

Portanto, para alcançar maior escalabilidade no sistema, o melhor seria combinar as duas soluções, pois, dessa forma, o usuário continuaria recebendo *feedback* rapidamente, mesmo quando houverem muitas requisições para serem processadas simultaneamente.

Repositórios:

https://github.com/mezuro/mezuro.git
https://github.com/mezuro/kalibro.git
https://github.com/choreos/choreos_v-v.git

Referências bibliográficas:

[1] Meirelles, P.; Morais, C; Martins, R. M.; Kon, F.; Santos, C.; Maldonado, J. C.. "Mezuro Plataform: Source Code Tracking Network", Sessão de Ferramentas do *III Congresso Brasileiro de Software: Teoria e Prática*, 2012.
[2] Filho, C. M. de O.; Kon, F.. "Kalibro: Interpretação de métricas de código-fonte". Dissertação de mestrado, Instituto de Matemática e Estatística, Univesidade de São Paulo, 2013.
[3] Moura, P.; Kon, F.. "Automated Scalability Testing of Software as a Service", *8th International Workshop on Automation of Software Test*, 2013.
[4] CHOReOS: Large Scale Choreographies of the Future Internet. Disponível em http://www.choreos.eu. Acessado em novembro/2013.