XI Salão de Iniciação Científica PUCRS

Avaliando o desempenho de um software: motivação, metodologia e estudo de caso

Artur Freitas, Suzane Menon, Maicon da Silveira, Mariana Kolberg, Avelino Zorzo

CePES, Faculdade de Informática, PUCRS

Resumo

Conforme os *softwares* foram adquirindo maiores responsabilidades, a preocupação em avaliá-los também aumentou. É necessário garantir, de alguma forma, que estes *softwares* possuam qualidade, confiabilidade e desempenho. Embora não seja possível "testar a qualidade" de um *software*, também não é possível construir um *software* de qualidade sem teste e análise. Além disto, a qualidade não é um ingrediente que possa ser acrescentado em uma última etapa antes da entrega; portanto, é necessário realizar atividades de verificação e validação durante todo o desenvolvimento do produto. Outra vantagem do teste e análise é a diminuição dos custos para corrigir falhas, pois quanto antes um erro for detectado, menor será o custo para corrigi-lo.

Introdução

A indústria de *software* tem investido cada vez mais recursos na busca pela qualidade de seus produtos (DELAMARO e outros, 2007). Segundo o padrão ISO/IEC 9126 a qualidade é composta pela funcionalidade, eficiência, confiabilidade, portabilidade, usabilidade e manutenibilidade (PEZZÉ e YOUNG, 2008). O foco do teste de desempenho é avaliar exclusivamente a eficiência e a confiabilidade.

- **Eficiência:** habilidade de garantir o desempenho requerido sob certas condições.
  - **Comportamento temporal:** verifica requisitos de tempo de resposta.
  - **Utilização de recursos:** verifica recursos necessários para executar o *software*.
- **Confiabilidade:** habilidade de prover o nível requerido de serviço quando o *software* é usado sob as condições apropriadas.
  - **Maturidade:** habilidade de evitar erros resultantes de falhas de *software*.

- **Tolerância a falhas:** habilidade de manter um nível aceitável de funcionalidades mesmo na presença de falhas externas.
- **Recuperabilidade:** habilidade de recuperar dados após uma falha.

O teste de desempenho avalia o comportamento de uma aplicação quando submetida a uma carga de trabalho, ou seja, com um determinado número de usuários interagindo simultaneamente com a aplicação. Os resultados coletados no teste revelam os pontos onde os recursos da aplicação estão sendo desperdiçados ou utilizados de forma ineficiente. Nestes pontos, normalmente são verificados gargalos (*bottlenecks*) de *hardware* e *software*.

Metodologia

O *software* testado como estudo de caso é uma aplicação *web* para gerenciamento de perfis profissionais de funcionários de uma empresa. O sistema permite que os usuários realizem *login*, *logout*, cadastro de suas habilidades, certificações e experiências. A aplicação em teste utiliza o SGBD *MySQL* para persistência de dados, o *TomCat* como servidor web e *Java* como linguagem de programação.

A primeira etapa do teste de desempenho consistiu na criação de *scripts* para simular o comportamento do usuário. Em seguida, criamos o cenário de teste, onde definimos o tempo de duração do teste, a quantidade de usuários virtuais, os recursos a serem monitorados e outras configurações que dependem dos objetivos do teste. Para este teste, foi definido um cenário de uma hora de duração com 30 usuários virtuais realizando as transações de login, logout, cadastro de habilidades, certificações e experiências. Após a definição do cenário, foi utilizada uma ferramenta para automatizar a execução do teste e monitorar os recursos. O *software* usado foi o *HP LoadRunner*, que permite a criação e execução de *scripts* que simulam usuários interagindo concomitantemente com a aplicação em teste.

Após a execução dos testes, analisamos os resultados para avaliar o comportamento do sistema e a utilização dos recursos, na tentativa de encontrar problemas de desempenho.

Resultados

O *TomCat* e o *MySQL* estavam executando na mesma máquina e competindo pelos recursos disponíveis. Como podemos perceber na tabela 1, o *TomCat* usou, em média, três vezes mais tempo de processamento que o *MySQL*. A máquina monitorada possui quatro núcleos de processamento e essa métrica é a soma das porcentagens do tempo de processamento que o processo utilizou em cada um dos quatro núcleos da máquina.

Métrica	Significado	Média	Desvio padrão
% processor time – *MySQL*	Porcentagem de tempo que os processadores gastaram para executar instruções do *MySQL*	30,406	29,319
% processor time – *TomCat*	**Porcentagem de tempo que os processadores gastaram para executar instruções do *TomCat***	**90,125**	**90,291**

Tabela 1 Métricas dos processos *MySQL* e *TomCat*

Além disto, neste cenário com 30 usuários virtuais, obtivemos um tempo de resposta não satisfatório nas transações *Habilidade* e *Certificação*, como podemos ver na tabela 2:

Nome da transação	Média do tempo de resposta em segundos	Desvio padrão em segundos	Tempo máximo de 90% das transações em segundos
Login	0,197	0,472	0,44
Logout	0,016	0,018	0,021
Experiência	0,785	1,066	1,664
Habilidade	**17,198**	**5,17**	**13,086**
Certificação	**11,559**	**13,716**	**30,425**

Tabela 2 Tempo de resposta das transações

Conclusão

A construção de software de alta qualidade requer a combinação de atividades de projeto e construção com atividades de teste e análise. Assim sendo, podemos avaliar a qualidade, identificar defeitos e possibilitar melhorias no software. É importante lembrar que não podemos provar que um software está correto por meio da sua execução; portanto, um teste bem sucedido é o que revela a presença de um defeito. A importância de descobrir defeitos é que, segundo a engenharia de software, uma melhoria somente se torna possível quando um defeito é encontrado. Neste estudo de caso, conseguimos apontar as duas transações (*Habilidade* e *Certificação*) que estão deixando a desejar no tempo de resposta. O próximo passo seria investigar o código da aplicação que trata dessas transações para que a equipe de desenvolvimento possa aprimorá-lo.

Referências

PEZZÉ, M.; YOUNG, M., **Teste e Análise de Software – processos, princípios e técnicas**. Porto Alegre: Bookman. 2008.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M., **Introdução ao teste de software**. Rio de Janeiro: Elsevier. 2007.