8
Considerações finais

Meu objetivo era quantificar o peso de diferentes modelos de concorrência e modelos de *sandbox* em servidores web, especialmente no que se refere à linguagem Lua. Apesar do desempenho de servidores web já ter sido abordado em outros trabalhos, o uso da linguagem Lua para implementar servidores web é relativamente recente e a documentação a respeito praticamente inexistente. Também não encontramos referências ao uso de co-rotinas em servidores web.

Baseamos o trabalho na iniciativa do projeto Kepler de criar suporte web para aplicações Lua. Nos interessou especialmente o servidor web Xavante que implementa seu modelo de concorrência com co-rotinas. Para permitir uma comparação consistente do desempenho dos modelos de concorrência, estendemos o servidor implementando novos modelos de concorrência. Mantendo a idéia de criar comparações consistentes, implementamos um modelo de criação de processos com a semântica do Windows tanto no Windows como no Linux.

Identificamos que o modelo de *sandbox* era um dos itens que tinham grande influência no desempenho, porém não haviam referências sobre *sandboxes* em Lua, muito menos a respeito do seu desempenho. Não era possível afirmar com certeza como elas se comportariam em relação ao desempenho.

Comparamos os dois modelos que já existiam no projeto Kepler com a não utilização de *sandboxes* e com uma *sandbox* baseada em processos do SO que é um modelo mais tradicional de isolamento.

Criamos um modelo matemático para descrever as tendências de comportamento esperadas no sistema e fizemos a análise dos resultados baseados nas expectativa criadas por esse modelo.

8.1
Trabalhos relacionados

O trabalho relatado em (hu97measuring, hu99jaws) mediu o desempenho de diferentes modelos de concorrência em um *framework* para servidores web denominado JAWS. Esse framework foi escrito em C++. Além da linguagem de programação utilizada, os testes executados no JAWS foram para conteúdo estático. Aqui pensamos em aplicações, logo pensamos em conteúdo gerado

dinamicamente. Outra diferença é que no caso dos trabalhos baseados no JAWS procurou-se minimizar a sobrecarga da criação de novos *threads* e processos criando *pools* desses mecanismos. Nesse trabalho procuramos minimizar essa sobrecarga utilizando co-rotinas ao invés de *pools*. Porém a maior diferença é que Hu et. al. estavam preocupados em medir o desempenho utilizando os diferentes recursos de cada SO a fim de obter a melhor configuração para cada ambiente. Nesse trabalho nos preocupamos mais em usar recursos portáveis entre as plataforma e esperávamos resultados semelhantes nas diferentes plataformas. O único modelo não portável para todas as plataformas testadas foi o de clonagem de processos que não é suportado nativamente na família de SO Windows. Clonagem entrou na lista de testes para servir de referência, já que esse mecanismo serviu para implementar um mecanismo de criação de processos semelhante ao Windows em sistemas POSIX.

O livro "*Web Performance Tuning*" (Killelea98) discute as características das principais tecnologias web, aponta os seus pontos fracos e sugere otimizações para sistemas que utilizam essas tecnologias. São apresentadas diversas ferramentas e fatores para teste de desempenho. O uso de diferentes modelos de concorrência é apresentado como uma evolução de um modelo para o outro mas não fornece medições para a sua comparação. Nesse livro a questão da persistência é discutida no contexto de CGIs e Servlets Java.

8.2 Contribuições

Nesse trabalho conseguimos confirmar alguns comportamentos relativos aos modelos de concorrência e *sandboxes* aplicados à linguagem Lua. Exemplos são a criação de co-rotinas ser mais leve que a criação de *threads* e processos , a semântica de criação de processos do Windows ser mais lenta que a semântica Windows e que qualquer das *sandbox* testadas imprime uma sobrecarga de processamento na sua criação. Conseguimos desmentir outros que antes eram apenas suposições, como a criação e uso de uma *sandbox* Rings não ser mais lenta que uma *sandbox* Venv e que processos com sandbox nem sempre são mais lentos. Também pudemos verificar como alguns modelos de *sandboxes* influenciam o comportamento da concorrência. Conseguimos medidas reais dando uma idéia do quanto os modelos de concorrência e de *sandboxes* realmente influenciam no desempenho.

Mostramos que, mesmo com semânticas diferentes, é possível trabalhar na linguagem Lua com processos de forma análoga tanto nos sistemas POSIX quanto Windows. Nesse trabalho, implementamos a semântica de criação de processos do Windows no Linux, além de medir o peso dessa implementação

através dos testes de desempenho, comparando com uma implementação da semântica POSIX no Linux.

A partir do servidor web Xavante criamos um arcabouço que, além de permitir que outros modelos sejam testados, pode ser estendido ainda mais na direção de um servidor de aplicações como o J2EE, com os mais diversos serviços e funcionalidades.

Mostramos que não existe entre os modelos de concorrência e de *sandbox* uma combinação que seja definitivamente melhor, pois todos eles apresentam alguns pontos fortes e em contrapartida alguns pontos fracos.

Finalmente, algumas das modificações implementadas no Xavante neste trabalho foram aproveitadas pelo projeto Kepler e hoje já fazem parte do Xavante. Foram aproveitadas as modificações do módulo Copas para permitir que uma tarefa gerencie múltiplos sockets, permitindo que uma aplicação web possa abrir novas conexões inclusive com o próprio servidor web recursivamente. Também foram aproveitados ajustes no tratamento dos cabeçallhos HTTP e graças ao reaproveitamento de código para implementar o módulo de teste da *sandbox* Rings, foi corrigido um vazamento de memória no manipulador que dispara a sandbox Rings para o CGILua.

8.3 Trabalhos Futuros

Esse trabalho pode servir de base para a realização de testes de outros modelos de concorrência. Notamos que os modelos mais utilizados em sistemas que se propõem a atender altas cargas se valem de *pools,* como por exemplo *pool* de *threads* no Internet Information Server (IIS) ou no Apache 2.0 e *pool* de processos no Apache 1.3. O Xavante modificado nesse trabalho pode ser estendido sem maiores dificuldades para usar esses modelos de concorrência. Será interessante comparar os modelos baseados em *pools* com o modelo de co-rotinas.

Outra extensão interessante desse trabalho seria um estudo do quanto os mecanismos de sincronização pesam em um modelo *multi-threads,* comparando com um modelo com co-rotinas e outros mecanismos de IPC.

A questão da persistência, em diversos níveis do conjunto servidor e aplicação web, foi vista nesse trabalho de forma muito superficial, porém, acreditamos que essa é uma questão fundamental para criar um sistema na linha dos servidores de aplicação. Acreditamos que a questão do desempenho na persistência entre requisições à mesma aplicação passa pela questão de como fazer persistir a *sandbox*. Reaproveitar uma *sandbox* com a aplicação já carregada nela entre diversas requisições teoricamente elimina toda a sobrecarga de carga

da aplicação. O Xavante modificado nesse trabalho também parece indicado para fazer esse tipo de teste.