EA872: Laboratório de Programação de Software Básico

Atividade 11
Programação de servidor Web com *threads*

Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas

Eleri Cardozo
Matheus Fernandes de Oliveira
Marco A. Amaral Henriques

1 Projeto do servidor Web com *threads*: visão geral

Na última versão de seu servidor web, você utilizou processos para torná-lo concorrente e mais eficiente no tratamento de múltiplas requisições. Nesta atividade, você deverá implementar a concorrência por meio de *threads*, com base na experiência adquirida nas últimas atividades.

O trecho referente à criação de um processo-filho para tratamento de uma nova requisição deverá ser isolado em uma função que executará como thread. A chamada `fork` deverá ser substituída pela chamada `pthread_create` e o servidor deverá criar uma thread por requisição. Como as *threads* acessarão apenas o sistema de arquivos, nenhum mecanismo de sincronização se faz necessário (o sistema operacional cuida do acesso concorrente ao sistema de arquivos, não permitindo que um arquivo seja aberto por mais de um processo ou thread para escrita).

Nesta atividade você deve documentar em detalhes o novo código desenvolvido (apenas as partes novas relacionadas com *threads*) e incluí-lo no relatório (veja mais adiante).

Para avaliar as mudanças causadas pela adoção de *threads* em lugar de processos, você deverá utilizar o programa http_load (disponível em www.acme.com/software/http_load e na página da disciplina) e fazer medidas de desempenho para três versões de servidor web: a versão com apenas um processo, a versão com vários processos de uma só thread em cada um e esta nova versão com um processo e várias *threads*. Procure estressar o servidor simulando várias requisições simultâneas e busque o número de *threads* e de processos que resultem no melhor desempenho em cada caso (você irá notar que, a partir de um certo número de *threads* ou de processos, o desempenho começa a piorar, em vez de melhorar).

Siga as orientações abaixo na preparação dos relatórios.

2 Atividades para serem feitas durante a aula

1. (2,0) Compile o programa http_load e use-o para testar servidores web da Unicamp, tais como: www.unicamp.br, www.fee.unicamp.br, www.ccuec.unicamp.br, www.comvest.unicamp.br, www.bc.unicamp.br etc. Fazendo testes com servidores do nosso campus, evitamos problemas de gargalo de conexão com a internet que podem aparecer no caso de conexão para fora. Mas vamos testar alguns servidores externos também para ver seu desempenho.

   Um ponto importante: esses testes deverão ser feitos a partir de uma das máquinas do laboratório conectadas à rede cabeada. Isso serve para eliminar a influência da instabilidade no desempenho causada pela grande dependência que o acesso sem fio tem do número de colisões de pacotes no canal de rádio (pacotes transmitidos simultaneamente colidem, são descartados e precisam ser retransmitidos pelo ponto de acesso sem fio e/ou pela interface de rede sem fio do notebook, resultando em diminuição da taxa de transmissão).

   (a) (0,4) O http_load tem dois parâmetros de partida (*rate* e *parallel*) e dois de término (*fetches* e *seconds*). Explique o significado/objetivo de cada um deles.

   (b) (1,0) Escolha dois servidores web (um dentro e outro fora da Unicamp) e proponha parâmetros de teste que mostrem as diferenças de desempenho entre eles (medido em número de *fetches/sec*). Aplique esses testes e documente os resultados, indicando os servidores testados, o IP da máquina do laboratório (máquina linux ligada à rede cabeada) a partir de onde você fez os testes e os resultados obtidos. Não faça estes testes a partir da rede sem fio, já que os resultados serão pouco confiáveis devido à maior instabilidade causada por colisões de pacotes de rede durante os testes. Repita cada teste pelo menos três vezes (em momentos espaçados) e adote o melhor resultado, pois ele mostra a capacidade máxima do servidor em atender requisições. Obs.: o arquivo de entrada para o programa http_load não pode conter linhas em branco e deve conter apenas um URL para cada teste. Observe que o protocolo HTTPS pode ser bem mais pesado que o HTTP. Por isso, não misture comparações de testes de HTTP com outros de HTTPS. Um manual do http_load está disponível no arquivo http_load.1 que pode ser lido com o comando "man -l http_load.1".

   (c) (0,6) Faça uma discussão dos resultados obtidos, justificando os parâmetros adotados e descrevendo a influência dos mesmos nos resultados.

3 Atividades para entrega antes da próxima aula

1. (1,0) Documente em detalhes no relatório o novo código desenvolvido (apenas as partes novas relacionadas com *threads* e outras que tenham sofrido alterações). A simples inclusão de códigos-fonte não documentados (não comentados devidamente) poderá prejudicar a avaliação deste item.

2. (5,0) Utilizando o programa http_load, faça medidas de desempenho para as três versões de servidor web citadas na Seção 1. Todos os testes e medidas devem estar devidamente documentados no relatório.

   (a) (2,0) Usando os servidores multiprocessos e *multithreads*, varie o número máximo de processos ou de *threads* de acordo com a seguinte lista: 1, 2, 4, 8, 16, 32 e 64 processos (ou *threads*). Para cada número de processos (ou *threads*) listado, determine com

ajuda do http_load qual é a melhor velocidade que pode ser obtida (em requisições atendidas por segundo (*fetches/sec*) e plote os resultados em um gráfico "núm. de processos (ou *threads*) X velocidade".

Sugerimos que o http_load seja executado com os seguintes parâmetros, mas você pode e deve experimentar variações que mostrem de forma mais clara as mudanças de desempenho de seus servidores:

`./http_load -parallel P -fetches F url_do_servidor.txt` , onde:

- P é o número máximo de requisições em paralelo que o http_load irá enviar: mesmo que fosse P=1, o teste já seria útil, pois seriam muitas requisições por segundo chegando ao servidor. Porém, sugerimos um valor entre 8 e 32 para estressar mais os servidores;
- F é o número total de buscas que serão feitas no servidor: é um critério de parada para o teste e sugerimos um valor entre 5.000 e 10.000 para ter dados suficientes que permitam identificar diferenças;
- url_do_servidor.txt é um arquivo texto contendo apenas uma linha com o link completo para o servidor como, por exemplo, http://localhost:porta .

Para cada configuração, execute o http_load pelo menos três vezes e use apenas o maior número de requisições atendidas por segundo, para sabermos do que o servidor em teste é capaz. Uma só execução poderá coincidir com um momento de sistema mais sobrecarregado e isso irá influenciar os resultados.

Como o próprio http_load é uma fonte de sobrecarga, será mais adequado se as máquinas cliente (executando o http_load) e servidor (executando seu servidor web) forem distintas fisicamente. Assim você terá melhores condições de avaliar o desempenho de seu servidor, sem que a máquina esteja sendo sobrecarregada pelo próprio programa de testes. Portanto, use duas máquinas diferentes do LE-27 para realizar os testes, sem compartilhá-las com mais ninguém e assim obter dados mais consistentes.

Tabele os dados coletados e plote pelo menos um gráfico (eixo x: *threads* e processos ; eixo y: requisições atendidas/segundo) que contenha os resultados para os servidores multiprocessos e para os *multithreads*. Caso os resultados destes dois sejam muito diferentes e isso dificulte a visualização simultânea em um único gráfico, plote-os em gráficos separados, cada um com sua própria escala no eixo y.

(b) (1,0) Para o servidor monoprocesso (o mais antigo) varie o parâmetro P na execução de http_load de 1 a 64, dobrando-o a cada execução para verificar se há alterações no desempenho do mesmo à medida que o número de requisições paralelas aumenta. Aqui também é recomendável fazer pelo menos três execuções de cada teste e pegar o melhor resultado.

(c) (1,0) Avalie e discuta seus resultados, justificando-os. Se for necessário, procure alterar os parâmetros usados no http_load e adotar valores que sejam suficientes para estressar os servidores web e mostrar as diferenças entre eles.

(d) (1,0) Discuta possíveis melhorias pontuais que poderiam ser feitas em cada uma das três versões do servidor para aumentar o desempenho máximo obtido nestes testes iniciais.

(e) (bônus de até 2,0) Nesta questão opcional, o aluno poderá receber até 2,0 pontos de bônus (válidos para esta atividade ou para outras) se implementar as melhorias discutidas no item anterior e comparar (por meio de testes e gráficos) os novos desempenhos do servidor com aqueles obtidos anteriormente. Deverá haver uma discussão explicando os motivos das melhorias terem (ou não terem) produzido os resultados esperados.

3. (2,0) Entregue em um arquivo zip a versão mais atual de cada um de seus três tipos de servidor. Cada servidor deverá estar em uma pasta própria com instruções de compilação (incluindo parte de lex e yacc) e de execução em arquivo README.TXT, a fim de que possamos compilar e testar cada tipo.