

Mini-Trabalho 2 – Entrega 31 de maio de 2010.

Título: Análise de Desempenho do Servidor Web DC/UFSCar

ATENÇÃO: LEIA ATÉ O FIM, DIVIDA AS TAREFAS NO SEU GRUPO E COMECE A TRABALHAR O MAIS BREVE POSSÍVEL

Descrição do Projeto:

O objetivo do projeto é analisar o desempenho de um servidor web - APACHE - que roda no departamento (DC). O servidor Web será instalado em sua máquina rodando em ambiente Linux. Serão feitos experimentos para geração de tráfego afim de caracterizar as demandas dentro do sistema. Neste ambiente experimental (Um servidor e um conjunto de conexões web) serão iniciadas. Os dados de medidas extraídos de vmstat, iostat e outros serão armazenadas em arquivos de log. Esse dados coletados serão usados para construir um modelo analítico do sistema. Esse sistema será modelado como um modelo aberto multi-classe onde as classes serão formadas segundo critérios extraídos de clusterização como tamanho do arquivo e frequência de aparecimento usando os logs para análise. Uma vez coletada as informações de demanda de acordo de diversos parametros, poderemos utilizar o modelo para extrair resultados teóricos do desempenho. Após a fase de análise de desempenho será feita uma previsão sobre a carga e o consumo dos recursos para o dobro da carga de trabalho. E finalmente, será validado através de experimentos testes para verificar se o modelo preve corretamente os resultados em comparação com o sistema real e com o sistema com carga dobrada de modo a testar a acurácia do modelo.

Fase I (Obtendo os **service demands**):

Separar duas máquinas para a fase de experimentação (pode ser seu laptop ou desktop de casa, sendo uma máquina para o grupo).

A) Instalar em uma das máquinas a última versão **stable** do servidor web apache (<http://httpd.apache.org/>), instalar PHP nessa máquina e configurar o servidor web para acessar o PHP (<http://www.vivaolinux.com.br/artigo/Configurando-PHP-com-Apache2-no-Linux/>). Essa máquina, doravante chamada SERVIDOR, deverá estar operando em Linux e ter todas as ferramentas de monitoração vmstat, iostat, etc.

B) Instalar na outra máquina, doravante chamada CLIENTE, o cliente web WGET (<http://www.gnu.org/software/wget/>), essa separação de cliente/servidor pode ser opcional para os grupos que não tiverem acesso a 2 máquinas. Entretanto, é bastante importante para separar os service demands.

C) Criar arquivos HTML aleatórios (ou pode copiar arquivos da Internet, EXCETO páginas HTML copiados do DC) com diferentes tamanhos, criar arquivos de imagem aleatórios de diferentes tamanhos (ou copiar da net) e criar pequenos programas em PHP, como Hello World e Mostrando as Variáveis de Sistema via Web.

D) Criar um script com testes na máquina CLIENTE contendo requisições HTML, IMG e PHP com diferentes tamanhos e intensidades. Para gerar a intensidade desejada utilizar chamadas em background do comando WGET intercaladas por comandos SLEEP. Segue um exemplo de script.

```
# assumindo CLIENTE 192.168.0.2 e SERVIDOR 192.168.0.1 conectados diretamente
# por Ethernet ou por um switch/hub, os nomes dos arquivos indicam seus tamanhos

# obtem uma página de 10KB e faz sleep por 1 seg (X0 = 1 req/sec)
wget http://192.168.0.2/10KB.html &
usleep 1000

# obtem uma página de 50KB e faz sleep por 1 seg (X0 = 1 req/sec)
wget http://192.168.0.2/10KB.html &
usleep 1000

# obtem uma imagem de 500KB e faz sleep por 1 seg (X0 = 1 req/sec)
wget http://192.168.0.2/500KB.jpg &
usleep 1000

# executa uma operação PHP e faz sleep por 1 seg (X0 = 1 req/sec)
wget http://192.168.0.2/hello.php &
usleep 1000

# obtem duas página de 10KB em um intervalo de 1 seg (X0 = 2 req/sec)
wget http://192.168.0.2/10KBa.html &
usleep 500
wget http://192.168.0.2/10KBb.html &
usleep 500

# requisições concorrentes da mesma “classe” – X0 = 3 req / seg
wget http://192.168.0.2/10KBa.html &
wget http://192.168.0.2/20KBb.html &
wget http://192.168.0.2/10KBc.html &
usleep 1000
....
entre outros tantos dezenas de testes (depende do seu planejamento e deve ser descrito mais
tarde no relatório que experimentos de caracterização do servidor APACHE foram feitos)
```

Para o conjunto de testes acima, execute os testes no mínimo 30 vezes por teste (para calcular médias) e prepare uma planilha com resultados coletando dados de vmstat (CPU), iostat (IO) entre outros. Colete esses dados rodando as ferramentas de monitoramento em conjunto com os testes e tente extrair quais são os valores de **service demand (D_i) para a CPU e o disco** para diferentes tamanho de arquivos, diferentes tipos de arquivos (html, imagem, php) e para diferentes intensidades ou requisições concorrentes por segundo (homogeneas = mesmo tamanho de arquivo e heterogeneas = diferentes tamanho, e tipos).

Extrapolar os resultados matematicamente para qualquer tamanho de arquivo e qualquer tipo (usando uma reta de fitting a partir dos seus resultados).

Fase II (Caracterização da Carga de Trabalho do Departamento de Computação)

- A) A partir do LOG de acesso do servidor Web do DC, desenvolver scripts para extrair 1) as URLs e montar uma tabela hash de URLs, 2) os tempos entre chegadas de requisições, 3) os tamanhos dos arquivos e 4) filtrar pelo código de retorno (ver formato do LOG do servidor web apache - <http://httpd.apache.org/docs/2.0/logs.html#accesslog>).

Consideraremos somente as requisições que deram certo 200 OK (opcionalmente, os alunos que quiserem fazer a caracterização do service demand para outros tipos de resposta como 404 Not Found, estão livres para fazê-lo). Portanto, é necessário filtrar o LOG.

- B) Filtrar dos logs por tipos: (a) HTML, (b) IMG, (c) programas executáveis.
- C) Fazer um estudo de similaridade da carga de trabalho para cada tipo filtrado usando algum programa de clusterização (k-means), pode usar o Matlab se quiser.
- Trocar o nome da URL por um código de ponto (ex.: URL: ~marcondes/index.html = 1 ou ponto 1), e usar 2 dimensões para clusterização: tamanho de arquivo e frequência de aparecimento no arquivo de log.
 - Rodar k-means para 3 clusters e calcular o tamanho e frequência do ponto médio
- D) Calcular a taxa de chegada de cada classe baseado nos logs (serão 9 classes, HTML_A, HTML_B, HTML_C, IMG_A, IMG_B, IMG_C, EXE_A, EXE_B, EXE_C) e a taxa de chegada de toda a carga agregada ao servidor DC.
- E) Passar os resultados para uma planilha parecida com a do Menasce, contendo 2 recursos (CPU e Disco) e 9 classes. Calcular utilização para cada classe/recurso e tempo de resposta para cada classe/recurso e somar para obter o tempo de resposta total. Em caso de desconfiar que a conta esteja errada, testar o tempo de resposta com o conjunto CLIENTE / SERVIDOR, gerando uma requisição WGET e verificando quanto tempo para fazer o download.
- F) Finalmente, extrapolar os resultados para uma situação hipotética do DOBRO DA CARGA DE TRABALHO DE DC. Ou seja, calcular utilizações e tempos de resposta (usando a fórmula da M/M/1) para $2 \times X_0$.

Fase III (Validação dos Resultados)

Essa é a última fase e a idéia é gerar experimentos controlados que possam EMULAR o ambiente de requisições Web do DC. Para tanto, será necessário preparar um extenso SCRIPT contendo várias chamadas WGET e USLEEP intercalados que possam fazer a MIMICA exata do acesso a páginas do DC. Entretanto, ao invés de termos as páginas do DC, vamos gerar páginas “aleatórias” com os tamanhos médios das classes (POR FAVOR, NÃO QUERO QUE NINGUÉM FAÇA TESTES COM O SERVIDOR DE PRODUÇÃO DO DC – QUEM FIZER PODE SER SEVERAMENTE PUNIDO).

- A) Crie páginas aleatórias para os experimentos dessa fase, use comando do UNIX (dd). O comando abaixo exemplo, extraído de http://en.wikipedia.org/wiki/Dd_%28Unix%29, cria um arquivo de 100 bytes aleatório. Anexe a esse um cabeçalho HTML ou HTML/IMG ou HTML/PHP. `dd if=/dev/urandom of=/home/sam/myrandom bs=100 count=1`

As páginas aleatórias devem ter os tamanhos e tipos de acordo com a caracterização de carga de trabalho acima (mas não devem ser somente um arquivo por classe HTML_A.html) é preciso adicionar um pouco de variabilidade. Por exemplo, suponhamos que a classe HTML_A seja de todas as requisições HTTP de até 100 KB, e que o desvio padrão dos tamanhos seja de 10KB. Então crie X (a seu critério) arquivos aleatórios com tamanhos variando entre 90KB e 110KB, contendo os nomes HTML_A1.html, HTML_A2.html, HTML_A3.html ... HTML_AX.html e posicione no diretório HTDOCS do Apache.

- B) Baseando-se no log filtrado e tratado (somente com 200 OK e com as classes) substituir cada entrada do LOG por uma chamada WGET para um dos arquivos aleatórios gerados acima de acordo com sua classe. Ex.:

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
```

Substituir no script por

```
WGET http://192.168.0.1/IMG_A.gif
```

- C) Baseando-se no log filtrado e tratado (somente com 200 OK e com as classes) substituir cada entrada do LOG por microsleeps do script BASH tendo EXATAMENTE o mesmo tempo entre chegadas do DC. Isso efetivamente vai gerar tráfego na mesma cadência que o servidor do DC recebe da Internet. Se tiverem chamadas concorrentes, com zero segundos entre elas, usar o & para colocar todos os WGET em ação ao mesmo tempo. Segue um exemplo geral de como montar o script:

```
# log -----
```

```
207.46.199.39 - - [04/Apr/2010:04:04:30 -0300] "GET ...
```

```
81.52.143.29 - - [04/Apr/2010:04:05:07 -0300] "GET ...
```

```
# script a ser criado -----
```

```
wget http://192.168.0.1/HTML\_A1.html
```

```
# 37 segundos, diferença entre 4hs 05m 07seg e 4hs 04m 30seg.
```

```
usleep 37000
```

```
wget http://192.168.0.1/HTML\_A2.html
```

- D) Executar o grande script coletando medidas de tempo de resposta em cada WGET (<http://www.gnu.org/software/wget/manual/wget.html#Logging-and-Input-File-Options>). Será necessário colocar um parâmetro para cada WGET `http:/... -o wget1.log`, onde `wget1.log` é o primeiro wget realizado, e assim sucessivamente. Em paralelo, executar o monitoramento do SERVIDOR apache e extrair utilização de CPU e DISCO. De volta ao CLIENTE, extrair dos logs os tempos de resposta e fazer a média para cada CLASSE.

- E) Comparar os resultados de validação com os resultados analíticos.

- F) Refazer os passos C, D e E modificando os intervalos entre chegadas para a METADE (1/2) do que eles são. Isso efetivamente vai fazer o tráfego (a carga de trabalho) DOBRAR. Por exemplo, acima:

```
# log -----  
207.46.199.39 - - [04/Apr/2010:04:04:30 -0300] "GET ...  
81.52.143.29 - - [04/Apr/2010:04:05:07 -0300] "GET ...  
  
# script a ser criado -----  
wget http://192.168.0.1/HTML\_A1.html  
# 37 segundos, diferença entre 4hs 05m 07seg e 4hs 04m 30seg.  
# fazer 37 / 2 = 18 segundos arredondado  
usleep 18000  
    wget http://192.168.0.1/HTML\_A2.html
```

Fase IV (Relatório) – Escrever um relatório de 4 a 6 páginas descrevendo todos os resultados. Comparar os tempos de resposta ANALÍTICO versus EMULADO, através de gráficos CDF (<http://www.mathworks.com/access/helpdesk/help/toolbox/stats/cdfplot.html>) dos tempos de resposta sobrepostos. Para o caso do TRÁFEGO NORMAL e para o caso do TRÁFEGO DOBRADO.

NÃO SUBSTIME O TRABALHO.

BOM TRABALHO !!!

Cesar Marcondes