CSE 124
February 2, 2016
Winter 2016, UCSD
Prof. George Porter
Thought question

• How many web servers are in your home?
• Unexpected places:
  – HP printers
  – TiVOs, and some other DVRs
  – Pre-installed on many computers (e.g., MacOS)
  – Internet-of-things (NEST thermostat?)
  – Configuration interface to wireless routers
  – Raspberry pi
  – Refrigerators?

Fridge sends spam emails as attack hits smart gadgets

© 17 January 2014 | Technology
Announcements

• HW 3 due today (2/2); Project 2 due 2/9
• Mid-course survey due Thursday (2/4)
• **No class Thursday**
  – Instead, project 2 office hours in my office, 8-9:20am

• Today’s plan:
  – Performance measurement and presentation
Project 2 idea

- You may find the `sendfile()` system call to be useful in implementing your web server
  - Copies data from a file descriptor to a socket
Application performance characterization

How we do quantify the performance of our software?
Simple Network Model

- Network is a pipe connecting two computers

- Basic Metrics
  - Bandwidth, throughput, latency, overhead, error rate, and message size

Packets/messages
Performance Metrics

- **Bandwidth**: number of bits capable of being transmitted per unit of time
- **Throughput**: number of bits transmitted per unit time
- **Latency** = Propagation + Transmit + Queue
  - Propagation = Distance/SpeedOfLight(*)
  - Transmit = 1 bit/Bandwidth
- **Overhead**
  - # secs for CPU to put message on wire
- **Error rate**
  - Probability P that message will not arrive intact

* Speed of light varies depending on the medium
## Bandwidth vs. Latency

### 1 Byte Object

<table>
<thead>
<tr>
<th>Bandwidth: 1 Mbps</th>
<th>Latency: 1 ms</th>
<th>Latency: 100 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1,008 µs</td>
<td>100,008 µs</td>
</tr>
<tr>
<td>Bandwidth: 100 Mbps</td>
<td>1,000 µs</td>
<td>100,000 µs</td>
</tr>
</tbody>
</table>

### 10 MB Object

<table>
<thead>
<tr>
<th>Bandwidth: 1 Mbps</th>
<th>Latency: 1 ms</th>
<th>Latency: 100 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>80.001 s</td>
<td>80.1 s</td>
</tr>
<tr>
<td>Bandwidth: 100 Mbps</td>
<td>.801 s</td>
<td>.9 s</td>
</tr>
</tbody>
</table>

⇒ Desirable network qualities based on application semantics
What Type of Pipe Do You Want?
The “ratio game”

<table>
<thead>
<tr>
<th>System</th>
<th>Workload 1</th>
<th>Workload 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

• Class poll: Characterize the performance:
  1.  $A > B$
  2.  $B > A$
  3.  $A == B$
  4.  $A > B$ and $B > A$
The “ratio game”

Throughput in transactions/second

<table>
<thead>
<tr>
<th>System</th>
<th>Workload 1</th>
<th>Workload 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

- Scenario 1: Average ratio w/ equal workloads:

<table>
<thead>
<tr>
<th>System</th>
<th>Workload 1</th>
<th>Workload 2</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>20</td>
<td>15</td>
</tr>
</tbody>
</table>

- Result: A == B
The “ratio game”

<table>
<thead>
<tr>
<th>System</th>
<th>Workload 1</th>
<th>Workload 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

Throughput in transactions/second

• Scenario 2: Average ratio w/ workload B as baseline:

<table>
<thead>
<tr>
<th>System</th>
<th>Workload 1</th>
<th>Workload 2</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>0.5</td>
<td>1.25</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

• Result: A > B
The “ratio game”

<table>
<thead>
<tr>
<th>System</th>
<th>Workload 1</th>
<th>Workload 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

Throughput in transactions/second

- Scenario 3: Average ratio w/ workload A as baseline:

<table>
<thead>
<tr>
<th>System</th>
<th>Workload 1</th>
<th>Workload 2</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0.5</td>
<td>2</td>
<td>1.25</td>
</tr>
</tbody>
</table>

- Result: B > A
Performance Metrics Confusions

• Mega versus Mega, Kilo versus Kilo
  – Computer architecture: Mega $\rightarrow 2^{20}$, Kilo $\rightarrow 2^{10}$
  – Computer networks: Mega $\rightarrow 10^{6}$, Kilo $\rightarrow 10^{3}$

• Mbps versus MBps
  – Networks: typically megabits per second
  – Architecture: typically megabytes per second

• Bandwidth versus throughput
  – Bandwidth: available over link
  – Throughput: available to application
How to Measure Bandwidth?

1) Measure total time to receive all packets at receiver

Slow bottleneck link
How to Measure Latency?

Measure round-trip time (assumes symmetric links)
How to Measure Error Rate?

Measure number of packets acknowledged

Packet dropped

Slow bottleneck link
ab - Apache HTTP server benchmarking tool

ab is a tool for benchmarking your Apache Hypertext Transfer Protocol (HTTP) server. It is designed to give you an impression of how your current Apache installation performs. This especially shows you how many requests per second your Apache installation is capable of serving.

### Synopsis

```
ab [ -A auth-username:password ] [ -b windowsize ] [ -c concurrency ] [ -C cookie-name=value ] [ -d ] [ -e csv-file ] [ -f protocol ] [ -g gnuplot-file ] [ -h ] [ -H custom-header ] [ -i ] [ -k ] [ -n requests ] [ -p POST-file ] [ -P proxy-auth-username:password ] [ -q ] [ -r ] [ -s ] [ -S ] [ -t timelimit ] [ -T content-type ] [ -u PUT-file ] [ -v verbosity] [ -V ] [ -w ] [ -x <table>-attributes ] [ -X proxy[:port] ] [ -y <tr>-attributes ] [ -z <td>-attributes ] [ -Z ciphersuite ] [http[s]://]hostname[:port]/path
```

### Options

- **-A auth-username:password**  
  Supply BASIC Authentication credentials to the server. The username and password are separated by a single : and sent on the wire base64 encoded. The string is sent regardless of whether the server needs it (i.e., has sent an 401 authentication needed).

- **-b windowsize**  
  Size of TCP send/receive buffer, in bytes.

- **-c concurrency**  
  Number of multiple requests to perform at a time. Default is one request at a time.

- **-C cookie-name=value**  
  Add a Cookie: line to the request. The argument is typically in the form of a name=value pair. This field is repeatable.

- **-d**  
  Do not display the "percentage served within XX [ms] table". (legacy support).

- **-e csv-file**  
  Write the output in CSV format to the specified file.

- **-f protocol**  
  Use the specified protocol. Supported protocols are http, https, and http2.

- **-g gnuplot-file**  
  Write the output in gnuplot format to the specified file.

- **-h**  
  Display help message and exit.

- **-H custom-header**  
  Add a custom HTTP header to every request.

- **-i**  
  Interpret all lengths as bytes instead of as lines.

- **-k**  
  Keep connection open after the request is sent.

- **-n requests**  
  Specify the number of requests to send. Default is 100.

- **-p POST-file**  
  Send a POST request with the given file.

- **-P proxy-auth-username:password**  
  Send a proxy request with the given credentials.

- **-q**  
  Quiet mode. No progress or status messages are displayed.

- **-r**  
  Randomly re-order requests.

- **-s**  
  Specify the server to send requests to.

- **-S**  
  Specify the server to send requests to with SSL.

- **-t timelimit**  
  Set the time limit in seconds.

- **-T content-type**  
  Specify the content type of the request.

- **-u PUT-file**  
  Send a PUT request with the given file.

- **-v verbosity**  
  Specify the level of verbosity for the output.

- **-V**  
  Verbose output.

- **-w**  
  Specify the window size for the client.

- **-x <table>-attributes**  
  Add attributes to the table output.

- **-X proxy[:port]**  
  Specify the proxy server to use.

- **-y <tr>-attributes**  
  Add attributes to the table output.

- **-z <td>-attributes**  
  Add attributes to the table output.

- **-Z ciphersuite**  
  Specify the cipher suite to use.

- **[http[s]://]hostname[:port]/path**  
  Specify the server to send requests to.

- **[http[s]://]hostname[:port]/path**  
  Specify the server to send requests to with SSL.
In-class exercise

Measuring web performance
Part 1: Measure latency of web requests to your servers

- Break into small groups
- Write program to request a small document (few 10s to 100s of bytes long)
- Measure latency
- Run 10x times
- Calculate:
  - Mean
  - Standard deviation

\[ \sigma = \sqrt{\frac{\sum(x - \bar{x})^2}{N}} \]

where

\[ \sigma = \text{the standard deviation} \]

\[ x = \text{each value in the population} \]

\[ \bar{x} = \text{the mean of the values} \]

\[ N = \text{the number of values (the population)} \]
Part 2: Measure requests/sec of your server

- Write program to request a small document (few 10s to 100s of bytes long)
- Run within a loop e.g., 100 times
- Calculate:
  - Total time
  - Time per request
  - Requests per second
Part 3: Measure throughput of your server

• Write program to request document of size:
  – 100 bytes
  – 100KB
  – 1MB
  – 10MB

• Run $N=10x$ times

• Calculate:
  – Mb/sec for the set of N requests
  – May need to increase N, especially for small documents