The Impact of Weights on the Performance of Server Load Balancing (SLB) Systems

Jörg Jung

University of Potsdam
Institute for Computer Science
Operating Systems and Distributed Systems

March 2013
# Outline

1 Introduction .................................................. 3

2 Determine Weights ........................................... 5

3 Metrics for Benchmarking .................................... 7

4 Measurements and Evaluation ............................... 13

5 Conclusions .................................................. 24
Dispatcher based Server Load Balancing (SLB): scalable, flexible and fault tolerance services
1 Introduction

Motivation

Simulations in [Lehmann et al. 2008] confirm impact of incorrectly estimated weights

Small deviation of 10 % results in significant higher number of dropped requests

Compare algorithms:

- Weighted Round Robin (WRR) and Weighted Least Connection (WLC)

→ Measure the impact of weights on the performance
2
Determine Weights

System administrator *may* run local benchmarks and does an “educated guess”
Factors on Weights

Hardware differences: CPU, Memory, HDD, NIC and PCI bus speed

Software differences: utilized SLB and back end server software

Workload scenarios: which trace characteristics are given

→ Hard to find mappings to set factors into relations

→ Each SLB systems with given setup requires benchmarking
3

Metrics for Benchmarking

SLB algorithm metrics for Internet Service Providers (ISPs) Service Level Agreement (SLA) definitions
3 Metrics for Benchmarking

**Connect Time and (First) Response Time**

- **Connect Time**:
  - Client: `connect()` → `t1` → `Syn` → `Syn-ACK` → `ACK` → `Server: accept()`

- **(First) Response Time**:
  - Client: `write()` → `t2` → `HTTP Request` → `Server: read()` → `HTTP Response` → `write()`
Algorithm Metrics

*Connect Time and (First) Response Time* at client side from start $t_1$ until sending start $t_2$ and until the receive of the first byte $t_3$

*Transfer Time* the time required to fulfill a request – starts at $t_2$ and ends with last byte of the response, usually somewhere past $t_3$

*Throughput* Connection Throughput, Session Throughput and Byte Throughput representing the number of connections, session or bytes per second handled by the application

*(Request) Errors and Drops* on the network layer or service protocol specific due to Overloaded Servers or even an Overloaded Network
3 Metrics for Benchmarking

Server Load Balancing Penalty

\[
SLB \text{ Penalty} = \left( \frac{\text{response}_{\text{mean}}}{\text{response}_{\text{max}}} \right) \times \left( \frac{\text{request\_error}_{\text{mean}}}{\text{requests}_{\text{total}}} \right)
\]

\text{mean} \text{ and max values are calculated from all measurement iterations}

\text{errors} \text{ include network and protocol errors e.g. HTTP 5xx Server Errors}

→ Created with ISP requirements in mind:

Duration is ignored as not required for SLA definitions
Metrics and Timestamps

Exclude *Connect Time* from *(First) Response Time* as persistent connections are re-used with HTTP/1.1 (keep-alive)

Several time related functions and instructions should be avoided for benchmarking:

- `time()` and `gettimeofday()`: both return the so called *Best Guess* of the *Wall Time* which can jump (e.g. influenced by NTP)

- RDTSC instruction: With SMP TSC might not be synchronized between cores, might stop or change its frequency when the CPU enters lower power modes, hence probably jump [Brunner 2005]

→ *httpperf* [Mosberger et al. 2013] and *http_load* [Poskanzer 2006] use the wrong function: `gettimeofday()`
3 Metrics for Benchmarking

servload

The web server benchmark *servload*

Load, optionally increase and replay workloads

Use correct timestamp functions and provide metrics

Support for HTTP and DNS
Measurements and Evaluation


Dispatcher based SLB scenario: two armed, *NAT* based and using route path.

Comparing *WRR* and *WLC* algorithms with different weights.
Outcomes and Metrics

Service of the SLB cluster is to answer HTTP requests

Requests can be successfully completed or fail

- Failures on the network connection may result in aborted or incomplete requests and responses
- Fail due to Overloaded Servers may result in aborted requests and wrong, incomplete or aborted responses

*SLB Penalty* is used for comparison

Wikipedia instance access traces from 2008 are used as available from [Pierre 2010]

Input workload is from 12. November 2007:

- Reduced to the first ten minutes of the log
- Filtered and reduced to common upload content (e.g. images) and English requests
- Converted to Common Log Format as input for servload

→ Remaining 1,584,996 requests are reduced to three final traces
Workload: Reduced Traces

Number of requests from the first ten minutes of the *Wikipedia* trace

<table>
<thead>
<tr>
<th>Factor</th>
<th>Number of Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{32}$</td>
<td>49,532 requests</td>
</tr>
<tr>
<td>$\frac{1}{16}$</td>
<td>99,063 requests</td>
</tr>
<tr>
<td>$\frac{1}{8}$</td>
<td>198,125 requests</td>
</tr>
</tbody>
</table>
Environment: Hardware

<table>
<thead>
<tr>
<th>Hostname</th>
<th>Client LB and Web Server 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Dual 1.8 GHz AMD Opteron 244 with 1,024 KByte Cache</td>
</tr>
<tr>
<td>GE NIC</td>
<td>Broadcom BCM95704A7</td>
</tr>
<tr>
<td>Hostname</td>
<td>Web Server 2</td>
</tr>
<tr>
<td>CPU</td>
<td>2.8 GHz Intel Pentium 4 with 1,024 KByte Cache</td>
</tr>
<tr>
<td>GE NIC</td>
<td>Broadcom BCM5721</td>
</tr>
<tr>
<td>Hostname</td>
<td>Web Server 3</td>
</tr>
<tr>
<td>CPU</td>
<td>1.86 GHz Dual Core Intel Xeon 3040 with 2,048 KByte Cache</td>
</tr>
<tr>
<td>GE NIC</td>
<td>Broadcom BCM95754</td>
</tr>
</tbody>
</table>

All machines have 4 GByte memory and GBit links
4 Measurements and Evaluation

Environment: Software

3 *Apache HTTP* server 2.2.3 configured to handle 96 Clients at maximum each

*LVS LB* with *ipvsadm* 1.24

Client with *servload* 0.5 configured to 1,021 concurrent sessions at maximum

*OS LB* and *Servers*: *CentOS Linux* 5.7 with kernel 2.6.18-274.12.1.el5

*OS Client*: *Debian Linux* 5.0.10 with kernel 2.6.26-2-amd64

*Monitoring*: *SNMPv1* requests once a minute from *LB* to localhost, client and web servers
Weights and Scenarios

<table>
<thead>
<tr>
<th>Web Server 1</th>
<th>Web Server 2</th>
<th>Web Server 3</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>RR/LC</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>23</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>788</td>
<td>623</td>
<td>1181</td>
<td>Byte-Unixbench</td>
</tr>
<tr>
<td>39</td>
<td>21</td>
<td>55</td>
<td></td>
</tr>
</tbody>
</table>

Each pass for **WRR/WLC**: 11 times with $\frac{1}{32}$, $\frac{1}{16}$ and $\frac{1}{8}$
Results: (First) Response Time

![Graph showing (First) Response Time for different workloads and scheduling policies.](image-url)
Results: (Request) Errors

- **RR**:
  - WRR(1,2,5)
  - WRR(4,8,10)
  - WRR(1,2,3)
  - WRR(23,42,73)
  - WRR(623,788,1181)

- **LC**: WLC(1,2,5), WLC(4,8,10), WLC(1,2,3), WLC(23,42,73), WLC(623,788,1181), WLC(21,39,55)
Results: SLB Penalty
4 Measurements and Evaluation

Load Averages on Web Servers for $\frac{1}{8}$ workload

--- Server 1 --- Server 2 --- Server 3

--- Server 1 --- Server 2 --- Server 3

Duration [min]

WRR (21, 39, 55)

WLC (21, 39, 55)
5 Conclusions

*SLB Penalty* of WRR and WLC with triple (623, 788, 1181)
5 Conclusions

Conclusions and Future Work

*SLB Penalty* introduced

Previous simulations are confirmed

Badly chosen weights may lead to unpredictable substantive worse results

*Byte-Unixbench* is a good option to determine weights

*WRR* may be better choice in ISP scenarios and under peak load

Next step: *SALBNET* and self-adapting weights
References


