Self-Adapting Load Balancing for DNS

Jörg Jung, Simon Kiertscher, Sebastian Menski, and Bettina Schnor

University of Potsdam
Institute of Computer Science
Operating Systems and Distributed Systems
Before we start ...
Outline

1. Introduction
2. Credit based SLB
3. Implementation
4. Measurements and Evaluation
5. Conclusion and Future Work

University of Potsdam, Simon Kiertscher
Dispatcher based Server Load Balancing (SLB): scalable, flexible, and fault tolerance services
Motivation

• Sophisticated algorithms are required for **heterogeneous** workloads and **heterogeneous** back end servers

• **No weights** to determine/tune

• Self-adapting **credit** based SLB for better performance

• Measurements show the advantages of credit based SLB for **TCP services** like HTTP [Jung et al. 2014]

• contribution:
  → Efficient implementation for **UDP services** like DNS required, using a **suited credit metric**
  → Measurements to compare traditional and credit based SLB algorithms: WRR vs. DPR
Credit based SLB

• introduced in [Jung et al. 2014]:
  • Application independent metrics are used to calculate credits
  • Back end server push credits to the LB
• Linux environments
• Credits represent the free capacity of the UDP receive queue / TCP backlog queue (in number of UDP requests or TCP connections)
receive queue vs backlog queue
Credit Metric: UDP Receive Queue

• UDP receive queue as credit metric implicitly reflects the current load on the back end servers:
  • A filled receive queue indicates a busy application
  • With a full queue the application if fully engaged and might not respond to requests
Implementation - Overview of salbnet

Diagram showing the implementation details of salbnet, including nodes such as Dispatcher, Server, and various libraries and protocols like RDMA, libnetmsg, librdmacm, libc, LVS scheduler, TCP/UDP, IP, IPoIB, IB CM, Verbs, Ethernet, and the integration with LD_PRELOAD and httpd/named.
Implementation

• Using salbnet libnethook for recvmsg() system call interception
  → get size of the received message (payload)
  → transparent for UDP applications, like DNS server (BIND)

• UDP packet size and required space in memory needs to be predicted
  → Find out free capacity
  → Experimentally determined for Linux kernel
  → Proof of concept prototype: dependent on specific kernel version
Measurements and Evaluation

• Measurements in Internet Service Provider (ISP) like SLB environment using BIND instances
• Heterogeneous hardware and homogenous software versions
• WRR vs. DPR
• 3 heterogeneous back end servers require weights for traditional WRR algorithm
  → Determined in beforehand measurements
  [Zinke and Schnor 2013]
Environment: Setup

Diagram:
- Node015
- IB1
- VIP
- GE switch
- IB4
- IB6
- IB8

University of Potsdam, Simon Kiertscher
Environment: Hardware

<table>
<thead>
<tr>
<th>Hostname</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVS and IB4</td>
<td>Dual 1.8 GHz AMD Opteron 244</td>
</tr>
<tr>
<td>IB6</td>
<td>2.8 GHZ Intel Pentium 4</td>
</tr>
<tr>
<td>IB8</td>
<td>1.86 GHz Dual Core Intel Xeon 3040</td>
</tr>
</tbody>
</table>

- All machines have 4 GByte memory and GBit links
- Client (load generator) runs with 12 GByte memory
Benchmark: servload [Zinke et al. 2012]

• Existing DNS benchmarks like *queryperf* or *DNSPerf* and *ResPerf* are not able to replay queries

• In contrast benchmark *servload* tries to replay real user sessions
  → Factor feature replicates real user sessions
  → Support for DNS protocol added to *servload*
Environment: Software

- 3 BIND DNS servers v9.3.6
- LVS LB with ipvsadmin v1.24
- Client with servload v0.5.1
- OS LB and Servers: CentOS Linux 5.7 with kernel 2.6.18
- OS Client: Debian Linux with kernel 2.6.26
- Monitoring: SNMPv1 requests once a minute from LB (in addition to the credits)
Workload: Increased DNS Traces

Number of requests from five minutes of a trace from the 29th September 2011 from DNS servers running haiti.cs.uni-potsdam.de

<table>
<thead>
<tr>
<th>Factor</th>
<th>Requests</th>
<th>Sessions</th>
<th>Mean req/s</th>
<th>Max req/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22,594</td>
<td>33</td>
<td>75.31</td>
<td>204</td>
</tr>
<tr>
<td>400</td>
<td>9,037,600</td>
<td>13,200</td>
<td>30,125</td>
<td>81,600</td>
</tr>
<tr>
<td>800</td>
<td>18,075,200</td>
<td>26,400</td>
<td>60,250</td>
<td>163,200</td>
</tr>
<tr>
<td>1,600</td>
<td>36,150,400</td>
<td>52,800</td>
<td>120,501</td>
<td>326,400</td>
</tr>
</tbody>
</table>
Results: (First) Response Time

<table>
<thead>
<tr>
<th>Method</th>
<th>F 400</th>
<th>F 800</th>
<th>F 1600</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRR</td>
<td>4.05 ms</td>
<td>4.33 ms</td>
<td>4.24 ms</td>
</tr>
<tr>
<td>DPR</td>
<td>0.64 ms</td>
<td>0.62 ms</td>
<td>0.60 ms</td>
</tr>
</tbody>
</table>
Results: (Request) Errors

<table>
<thead>
<tr>
<th>Method</th>
<th>F 400</th>
<th>F 800</th>
<th>F 1600</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRR</td>
<td>1273 (0.01%)</td>
<td>2038 (0.01%)</td>
<td>2910 (0.01%)</td>
</tr>
<tr>
<td>DPR</td>
<td>343 (0.00%)</td>
<td>759 (0.00%)</td>
<td>1612 (0.00%)</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

• Credit based SLB extended to support UDP services
• servload benchmark extended to work with DNS
• Previous TCP (HTTP) measurements are confirmed
• DPR outperforms traditional WRR with already tuned weights
  • by 15% (FRT) and 25%-50% (error rate)
• salbnet scales
• Next: transition from proof concept to mature solution
Thank you for your attention!
Any questions?

Contact:
jozinke@cs.uni-potsdam.de
www.cs.uni-potsdam.de
References


