Hidden Scalability Gotchas in Memcached and Friends

Neil Gunther, *Performance Dynamics*
Shanti Subramanyakam, *Oracle Corporation*
Stefan Parvu, *Oracle Finland*

*Velocity 2010 Web Performance and Operations Conference*
Scalability
Memcached scale out

- Tier of older servers
- Mostly single CPU
- Single threading ok
Scalability Strategies

• **Qualitative** scalability
  – Scale **up**, e.g., big SMP servers
  – Scale **out**, e.g., many cheap servers (Unis)

• **Quantitative** scalability
  – What this talk is about
  – Need **controlled** measurements
  – Need **numbers** to see cost-benefit
Capacity Planning

- You know you need it
  - The *planning* bit, especially
  - Data ain’t information
  - Info is *hidden* in the data
- Just like finance, you need a *model*

**Metrics + Models == Information**
Controlled Measurements
Why Controlled Measurements?

Trying to predict scalability by looking at time series data is like trying to predict the stock mkt by watching the DJX ticker.
Bad Throughput Measurements

Need throughput measured in steady state (which this isn’t)

Need x-axis to be load (N) defined in terms of processes or users.
Average Throughput in Time

This is what steady state looks like as function of time. It corresponds to ONE throughput load point (N).
Controlled MCD Tests

Load Drivers
2 Sun Fire X4170
2 sockets, 64 GB

10 Gbe Switch

Memcached
Sun Fire X4170
2 sockets, 64 GB

SUT
Memcached scaling is thread limited
Better on SPARC Multicore
Quantifying Scalability

Universal Scalability Law

USL
1. Equal Bang for The Buck

Ideal parallelism

Capacity vs. Load

Graph showing the relationship between capacity and load, with a peak at an ideal load for maximum capacity.
2. Cost of Sharing Resources

![Graph showing the relationship between capacity and load](image)

- Capacity
- Load

---

Velocity 2010, June 24
3. Resource Limitation

Amdahl’s law

Capacity

Load

Capacity

Velocity 2010, June 24
4. Degradation Negative Return
Universal Scalability Law (USL)

\[ C(N) = \frac{N}{1 + \alpha(N - 1) + \beta N(N - 1)} \]

- **Concurrency**: \( \alpha = 0, \beta = 0 \)
- **Contention**: \( \alpha > 0, \beta = 0 \)
- **Coherency**: \( \alpha > 0, \beta > 0 \)
USL regression in Excel

<table>
<thead>
<tr>
<th>Users (N)</th>
<th>TPS X(N)</th>
<th>RelCap C=X(N)/X(1)</th>
<th>Efficiency CN</th>
<th>Inverse NC</th>
<th>Linear N-1</th>
<th>Deviation (N/C)-1</th>
<th>Trendline Quadratic</th>
<th>Parameters Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0</td>
<td>0.00</td>
<td>a 0.000340</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>94</td>
<td>9.78</td>
<td>0.98</td>
<td>1.02</td>
<td>9</td>
<td>0.02</td>
<td>b 0.001400</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>181</td>
<td>18.71</td>
<td>0.94</td>
<td>1.07</td>
<td>19</td>
<td>0.07</td>
<td>c 0.00003</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>238</td>
<td>24.70</td>
<td>0.82</td>
<td>1.21</td>
<td>29</td>
<td>0.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>280</td>
<td>26.96</td>
<td>0.67</td>
<td>1.48</td>
<td>39</td>
<td>0.48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>254</td>
<td>28.35</td>
<td>0.53</td>
<td>1.90</td>
<td>49</td>
<td>0.90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>246</td>
<td>25.44</td>
<td>0.42</td>
<td>2.36</td>
<td>59</td>
<td>1.36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>239</td>
<td>24.79</td>
<td>0.35</td>
<td>2.82</td>
<td>69</td>
<td>1.82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>237</td>
<td>24.60</td>
<td>0.31</td>
<td>3.25</td>
<td>79</td>
<td>2.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>229</td>
<td>23.77</td>
<td>0.26</td>
<td>3.79</td>
<td>89</td>
<td>2.79</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>222</td>
<td>23.03</td>
<td>0.23</td>
<td>4.30</td>
<td>98</td>
<td>3.30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A miracle happens …

Fit to Deviation from Linearity

\[ y = 0.00034x^2 + 0.00140x \]
Memcached Scalability

Quantitative USL Analysis
Scalability of mcd 1.2.8

N_{max} = 7
\alpha = 0.0255, \beta = 0.0210
Scalability of mcd 1.4.1

N_{max} = 6
\alpha = 0.0821, \beta = 0.0207

USL curve (not Excel)
Scalability of mcd 1.4.5

\[ N_{max} = 6 \]
\[ \alpha = 0.0988, \beta = 0.0209 \]

USL curve (not Excel)
Scalability of SPARC version

\[ N_{\text{max}} = 22 \]
\[ \alpha = 0.0041, \beta = 0.00197 \]

USL curves (not Excel)

\[ \alpha = 0, \beta = 0.000434 \]
USL projected scalability

\[ \alpha = 0.0041, \beta = 0.00197 \]

\[ N_{\text{max}} = 22 \]

\[ N_{\text{max}} = 48 \]

\[ \alpha = 0, \beta = 0.000434 \]

USL curves (not Excel)
Parameter interpretation

• Why $\alpha \sim 0$
  – Cache further partitioned
  – Single lock replaced by multiple locks

• Why $\beta > 0$?
  – Is it in mcd code?
  – Could it be in O/S, H/W, …?
Scaling Among Friends

Scalability as a function of virtual users (“friends”) not threads
JAppServer USL Analysis

USL curves (not Excel)

N = 700 users
\( \alpha = 0.00001486 \)
\( \beta = 6.7E-9 \)

N = 1200 users
\( \alpha = 0 \)
\( \beta = 2.4E-7 \)
Scalability on Amazon EC2

N_{max} = 22
\alpha = 0.038988298
\beta = 0.001432176

USL curve (not Excel)
Memcached Gotchas
Just throw more hardware at it!
Old scaling rules will be broken

• Current scale-out strategy relies on using older cheap hardware
• Older hardware is often single CPU
  – Single-threadedness of mcd is ok
• Newer hardware will be multicore
  – New hardware is faster with lots of cores
  – But mcd won’t be able to utilize all cores
  – Multiple mcd instances are mgmt headache
Single threading can wreck you
Summary

• Current mcd versions are thread limited
  – OK for older uniprocessor servers
  – Not OK for deployment on new multicores
  – Reason: unused processor capacity costs money

• Controlled measurements
  – Not time-series data from prod (but maybe can work)
  – Steady state throughput (or pick small prod window)

• Quantify scalability
  – Metrics + Models == Information
  – Goal is to reduce contention ($\alpha$) and coherency ($\beta$)
  – Nmax in mcd: Increased from 6 to 48 threads
Resources

• **Neil**
  - perfodynamics.blogspot.com
  - twitter.com/DrQz
  - www.perfdynamics.com/books.html
  - www.perfdynamics.com/Manifesto/USLscalability.html

• **Shanti**
  - perfwork.wordpress.com
  - twitter.com/shantiS

• **Stefan**
  - www.systemdatarecorder.org
  - twitter.com/sperformance