Why Are There No Giants?

How to Quantify Application Scalability

Dr. Neil J. Gunther

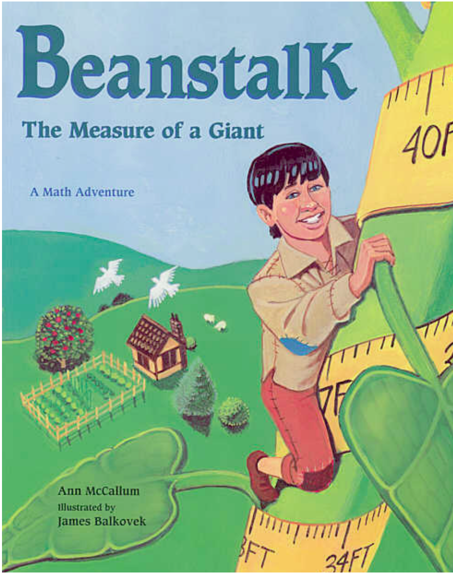
Performance Dynamics

NorCal ORACLE Users Group (NoCOUG)
Winter Conference, Feb 11, 2010
Keynote

Outline

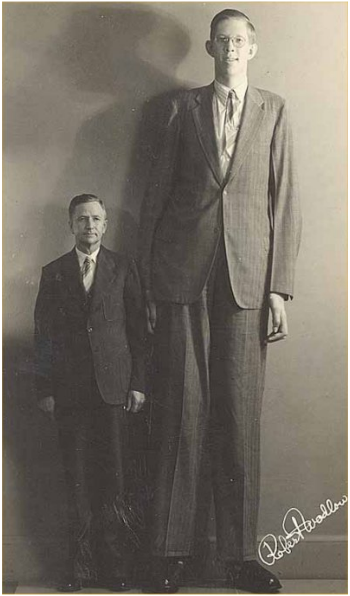
Jack and the Beanstalk

- Jack climbs a magic beanstalk up into the clouds (10,000 ft?)
- Guarded by a giant who is 10 times bigger than Jack
- *"Fee-fie-foe-fum!"* and all that

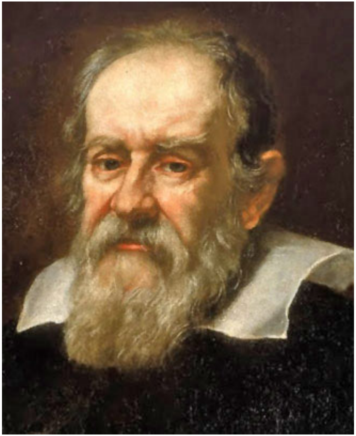


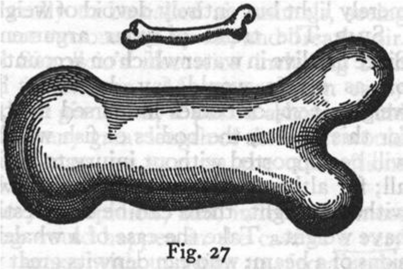
Where Are All the Giants?

- Can giants exist?
  - Can 10,000' beanstalk exist?
- Guinness world record
  - Robert P. Wadlow (USA)
  - Height: 8'11" (2.72 m)
- Jack
  - Height: 1.8 m tall (L)
  - Weight: 90 kg
- Giant (10x bigger)
  - Height: 18 m tall ($10 \times L$)
  - $L^3 \times 90$ kg = $10^3 \times 90$ kg
  - Weight: 90,000 kg
  - A bone-crushing 100 tons!



Galileo's Observation of 1638





- Double all dimensions
- Cross section: $4\times \equiv 2^2$
- But weight: $8\times \equiv 2^3$

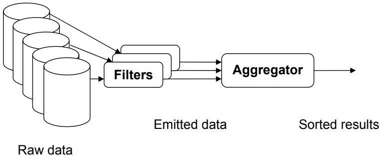
On Being the Right Size, J.B.S. Haldane, 1928

Scaling vs. Scalability

- Natural scaling
  - Inherent critical physical limits
  - When the load (volume) exceeds the material strength (area), things break
  - Load $\sim L^3$ (volume), but strength $\sim L^2$ (cross-section area)
- Computer scalability
  - No critical limit
  - Point of diminishing returns
  - Scalability is about sustainable size

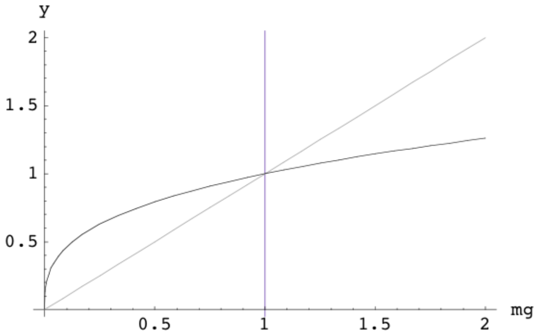
Scalability is Not a Number

- Google paper of 2005: "Parallel Analysis with Sawzall,"
  - *"If scaling were perfect, performance would be proportional to the number of machines, that is, adding one machine would contribute one machine's worth of throughput. In our test, the effect is to contribute 0.98 machines."*
  - Translation: scalability is 98% of ideal linear



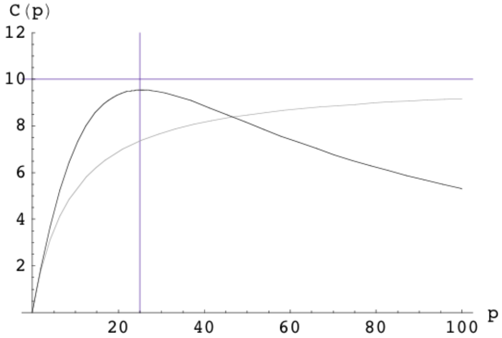
- Scalability is a *function*, not a number
  - Diminishing returns (due to increasing overhead) appears as a fall away from linearity
  - Want to express these losses as a quantitative function

Scaling Characteristics of a Natural System



Weight is linear ($y \propto m$) in mass (m), strength is curved ($y \propto m^{2/3}$)
Giant's leg bone or beanstalk stem collapses where curves cross

Scaling Characteristics of a Computer System

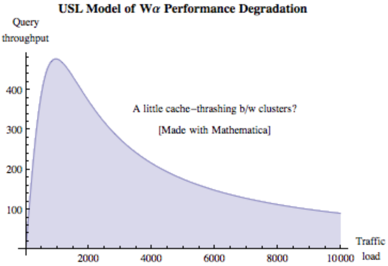


Critical point is maximum in throughput curve
Beyond max performance degradation or retrograde scalability
Can be either hardware scaling or application scaling

Quantifying Web 2.0 Scalability Fails

- Twitter.com
- Amazon EC2
- Cuil.com
- Apple iStore 2008
- Google Gmail
- WolframAlpha

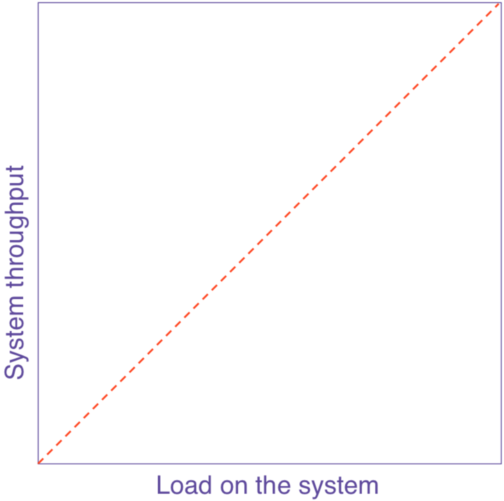
```
$Version
7.0 for Mac OS X x86 (64-bit) (November 11, 2008)

"Created by NJG on " <> DateString[]
Created by NJG on Fri 15 May 2009 18:07:47

Show[
 Plot[n / (1 + α n (n - 1) + β n (n - 1)) /. {α → 10^-7, β → 10^-6}, {n, 1, 10000},
  PlotRange → {All, All},
  PlotLabel → Style["USL Model of Wα Performance Degradation", Bold],
  AxesLabel → {"Traffic\nload", "Query\nthroughput"}, Filling → Bottom],
 Graphics[Text["A little cache-thrashing b/w clusters?", {6000, 350}]],
 Graphics[Text["[Made with Mathematica]", {6000, 300}]]
]
```



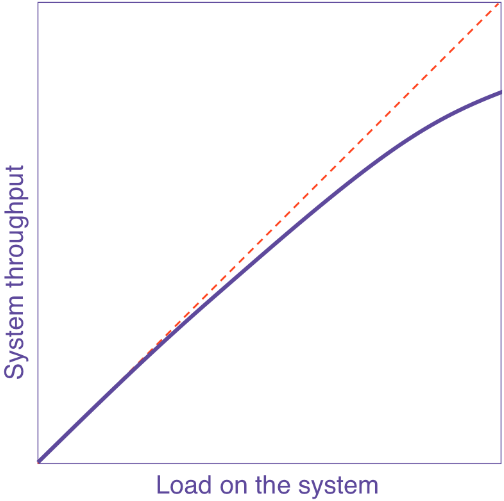
Outline

1. Scaling vs. Scalability
2. Components of Scalability
3. Example Scalability Analysis
4. Applications of the USL

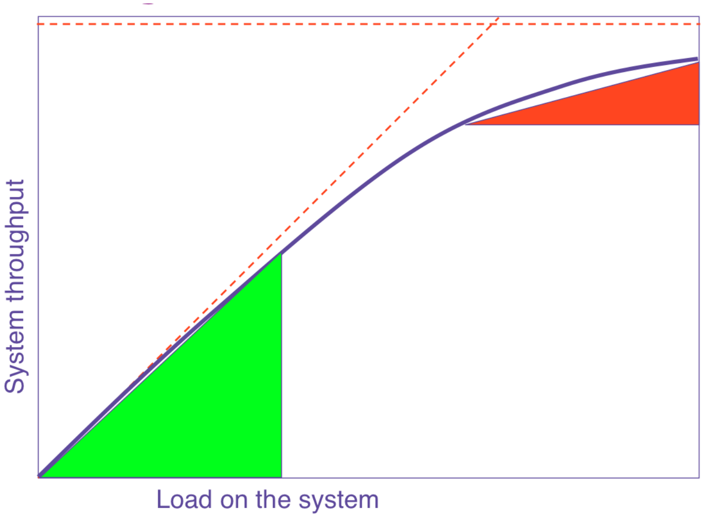
Equal Bang for the Buck (Concurrency)



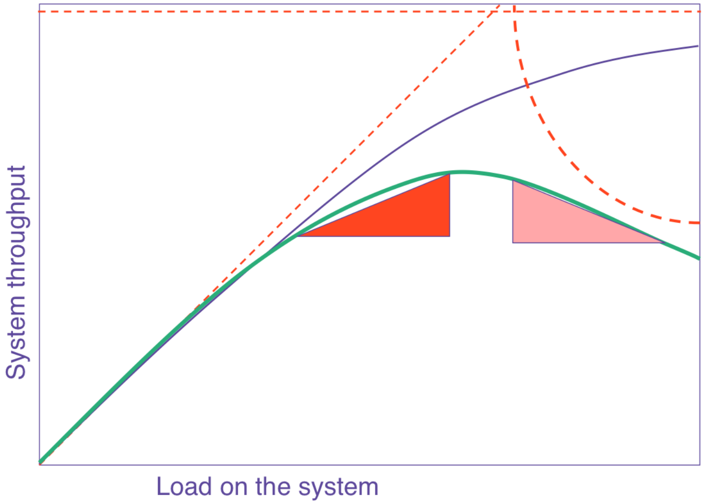
Cost of Sharing Resources (Contention)



Diminishing Returns (Saturation)

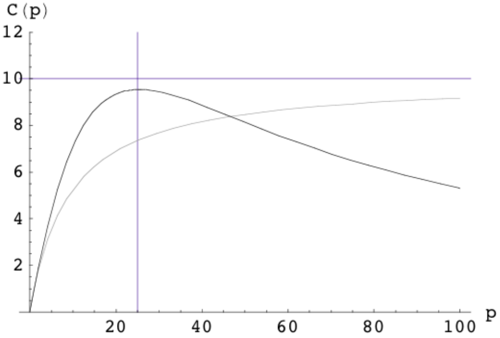


Negative ROI (Coherency Delays)



The Big Picture

Pulling all the pieces together



Would like to be able to compute this kind of scalability curve

Universal Scalability Law (USL)

- N users or processes

Universal Scalability Law (USL)

- N users or processes
- C capacity function of N

Universal Scalability Law (USL)

- N users or processes
- C capacity function of N

$$C(N, \alpha, \beta) = \frac{N}{\boxed{1} + \alpha\,(N - 1) + \beta\, N(N - 1)}$$

Universal Scalability Law (USL)

- N users or processes
- C capacity function of N

$$C(N, \alpha, \beta) = \frac{N}{\boxed{1} + \alpha\,(N-1) + \beta\,N(N-1)}$$

Three Cs:

Universal Scalability Law (USL)

- N users or processes
- C capacity function of N

$$C(N, \alpha, \beta) = \frac{N}{1 + \alpha\,(N-1) + \beta\,N(N-1)}$$

Three Cs:

1. **C**oncurrency

Universal Scalability Law (USL)

- N users or processes
- C capacity function of N

$$C(N, \alpha, \beta) = \frac{N}{1 + \alpha\,(N - 1) + \beta\, N(N - 1)}$$

Three Cs:

1. **C**oncurrency
2. **C**ontention (amount α)

Universal Scalability Law (USL)

- N users or processes
- C capacity function of N

$$C(N, \alpha, \beta) = \frac{N}{\boxed{1} + \alpha\,(N-1) + \beta\, N(N-1)}$$

Three Cs:

1. **C**oncurrency
2. **C**ontention (amount α)
3. **C**oherency (amount β)

Universal Scalability Law (USL)

- N users or processes
- C capacity function of N

$$C(N, \alpha, \beta) = \frac{N}{\boxed{1} + \alpha\,(N-1) + \beta\,N(N-1)}$$

Three Cs:

1. **C**oncurrency
2. **C**ontention (amount α)
3. **C**oherency (amount β)

Theorem (Universality)

Only need 2 parameters (α, β) to produce a maximum or critical point in $C(N)$ function.

ORACLE Example

Consider the following example:

- Many ORACLE processes running on a symmetric multiprocessor (SMP)
- ORACLE OLTP application (shared writable data)
- An ORACLE process requests to update data in the row of a table
- Must wait for RDBMS lock
- Finally, process gets the ORACLE lock (permission to write)
- ORACLE process is executing but ...
- It still cannot complete the write

Question: Why not?

ORACLE Example

Consider the following example:

- Many ORACLE processes running on a symmetric multiprocessor (SMP)
- ORACLE OLTP application (shared writable data)
- An ORACLE process requests to update data in the row of a table
- Must wait for RDBMS lock
- Finally, process gets the ORACLE lock (permission to write)
- ORACLE process is executing but ...
- It still cannot complete the write

Question: Why not?
Hint: Multiple processors means multiple local (L_2) caches

ORACLE Example

Consider the following example:

- Many ORACLE processes running on a symmetric multiprocessor (SMP)
- ORACLE OLTP application (shared writable data)
- An ORACLE process requests to update data in the row of a table
- Must wait for RDBMS lock
- Finally, process gets the ORACLE lock (permission to write)
- ORACLE process is executing but ...
- It still cannot complete the write

Question: Why not?
Hint: Multiple processors means multiple local (L_2) caches
Answer: If local cache is stale, must wait for consistent data

Data + Models: Need Both

Data		**Scalability**		**Model**
$\dfrac{X(N)}{X(1)}$	$\rightarrow$	$C(N, \alpha, \beta)$	$\leftarrow$	$\dfrac{N}{1 + \alpha(N-1) + \beta N(N-1)}$

Data + Models: Need Both

Data	Scalability	Model

$$\frac{X(N)}{X(1)} \rightarrow C(N, \alpha, \beta) \leftarrow \frac{N}{1 + \alpha(N-1) + \beta N(N-1)}$$

Theorem (Connections with queueing theory)

Data + Models: Need Both

Data		**Scalability**		**Model**
$\frac{X(N)}{X(1)}$	$\rightarrow$	$C(N, \alpha, \beta)$	$\leftarrow$	$\frac{N}{1 + \alpha(N-1) + \beta N(N-1)}$

Theorem (Connections with queueing theory)

1. *Amdahl's law* $\equiv$ *synchronous repairman (Gunther 2002)*

Data + Models: Need Both

Data		**Scalability**		**Model**
$\frac{X(N)}{X(1)}$	$\rightarrow$	$C(N, \alpha, \beta)$	$\leftarrow$	$\frac{N}{1 + \alpha(N-1) + \beta N(N-1)}$

Theorem (Connections with queueing theory)

1. *Amdahl's law* $\equiv$ *synchronous repairman (Gunther 2002)*
2. *USL* $\equiv$ *sync repairman doing exchange sort (Gunther 2008)*

Why Should You Care?

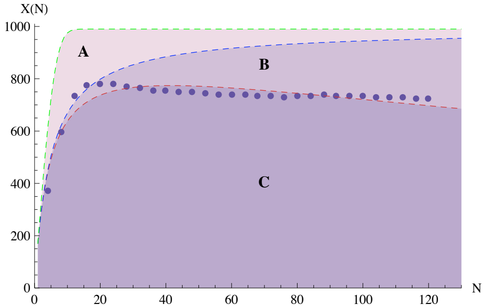
- Old reason: Concurrent programming is hard on SMPs
- New reason: Multicores are SMPs on a chip (it's back baby!)

Werner Vogels, Amazon.com CTO

"Scalability is hard because it cannot be an after-thought. Good scalability is possible, but only if we architect and engineer our systems to take scalability into account."

USL Scalability Zones

Think **zones** rather than curves



- A *A*-synchronous messaging (average queue lengths)
- B Synchronous messaging (worst queue lengths)
- C Synchronous messaging + exchange sorting

Data Are Not Divine

Data come from the Devil

Models come from the God

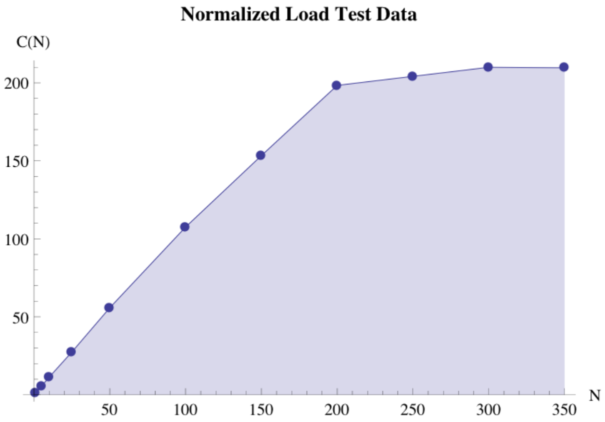
Data needs to be put in prison (a model) and made to confess the truth

Theorem

$$\textit{Data} + \textit{Models} \equiv \textit{Insight}$$

Example Measurements

J2EE web application



Throughput measurements using Apache Jmeter
Monotonically increasing, looks fine, but ...

Bad Data in Prison

Excel table of various USL quantities.

D	E	F	G	H
N	C(N)	C/N	N/C	(N/C)-1
1	1.00	1.00	1.00	0.00
5	5.67	1.13	0.88	-0.12
10	11.33	1.13	0.88	-0.12
25	27.50	1.10	0.91	-0.09
50	55.83	1.12	0.90	-0.10
100	107.50	1.08	0.93	-0.07
150	153.33	1.02	0.98	-0.02
200	198.33	0.99	1.01	0.01
250	204.17	0.82	1.22	0.22
300	210.00	0.70	1.43	0.43
350	209.67	0.60	1.67	0.67

Column F shows scaling efficiency: C/N. Between $N = 5$ and 150 vusers, efficiencies > 1.0. Can't have more than 100% of anything. Please explain?

Data + Model == Insight!

Just attempting to set up the USL model in Excel, shows measurement data (not the model) are wrong. Without it, ignorance is bliss.

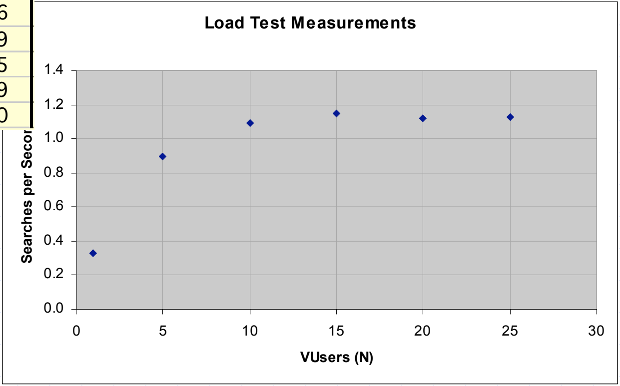
Outline

Oracle Based CRM

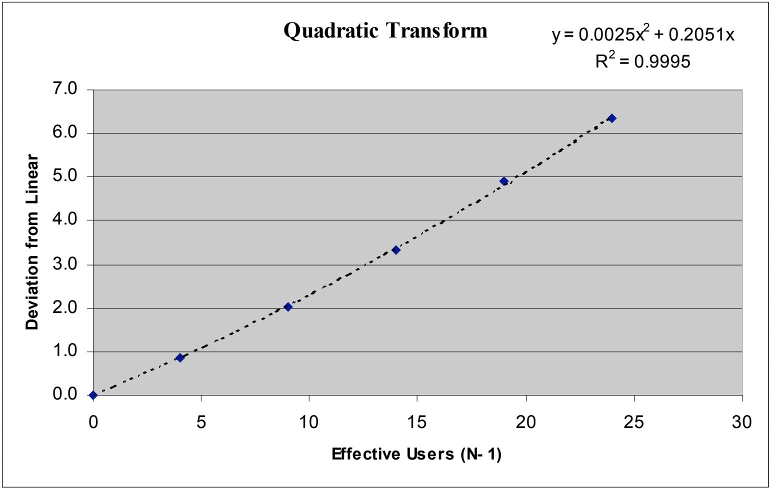
- CRM application
  - Real user login/authentification
  - Real user searches for customer parameters
- Transaction definition
  - Login (1-shot) incorporated in Init portion of LR script
  - Iterate on specific searches being evaluated
  - Mean TPS calculated as Action_Tx / Duration_Seconds
- Each VUser load must reach steady state
  - 10-15 mins. is common runtime per VU load
  - Use Rendezvous Start/Stop VUsers @ 15 mins
  - Use Goal mode rather than Scenario mode

LoadRunner Measurements

Measured VUs (N)	TPS X(N)
1	0.3311
5	0.8986
10	1.0899
15	1.1485
20	1.1199
25	1.1280



Regression Analysis in Excel



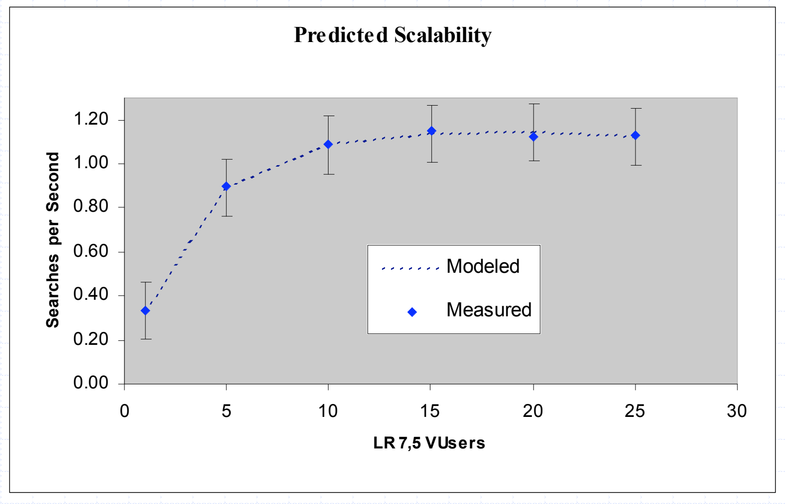
Excel Parameter Mappings

Trendline Quadratic	Parameters Coefficients	Super Parameter	Serial Values
a	2.50E-03	α	0.2026
b	0.2051	β	0.0123
c	0.0000	Nmax	19
		Nopt	5

Predicted Scalability

VUs	Predicted C(N)	Capacity Modeled	Measured
1	1.00	0.3311	0.3311
5	2.69	0.8899	0.8986
10	3.28	1.0861	1.0899
15	3.44	1.1387	1.1485
20	3.45	1.1418	1.1199
25	3.40	1.1243	1.1280

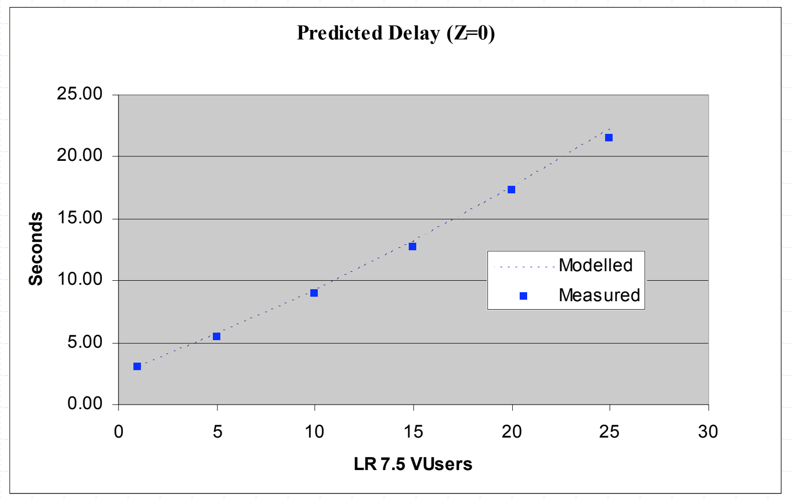
Measured and Predicted Throughput



Response Time Measurements

VU	Predicted Modelled	Delay Measured	Percent Error
1	3.02	2.96	2.03
5	5.62	5.44	3.29
10	9.21	8.94	3.00
15	13.17	12.69	3.80
20	17.52	17.23	1.66
25	22.24	21.44	3.71

Predicted Application Latency



Performance Analysis

- Accuracy:
  - Error $<$ 2% on throughput (v. good)
  - Error $<$ 4% on latency (excellent!)
- Contention (α): 0.2026 (21%)
  - Extremely high (ORACLE 2.5% to 3%)
  - ODBC calls need serious revision!
- Coherency (β): 0.0123 (Unbounded. Not
  - Relatively high
  - Database cache misses?
- Critical Loads:
  - Nmax: 19 is severely bottlenecked
  - Nopt: 5 users is untenable in production

Not ready for prime time

Outline

1 Scaling vs. Scalability

2 Components of Scalability

3 Example Scalability Analysis

4 Applications of the USL

When to Apply the USL

- Multiprocessing architectures (SMPs)
- Threaded applications
- Distributed caching instances
- Multicores are the new SMPs
- RAC-based architectures
- Multi-tier applications (Weblogic, Oracle)
- Any concurrent programming

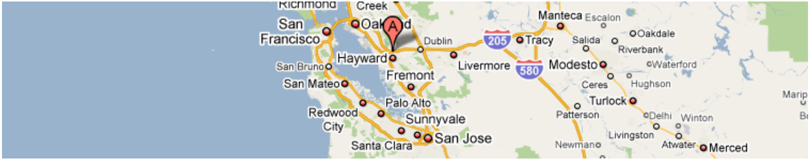
Where is Your Application?

Class A Ideal concurrency ($\alpha, \beta = 0$)	Class B Contention-only ($\alpha > 0, \beta = 0$)
Shared-nothing platform Google text search Lexus–Nexus search Read-only queries	Message-based queueing (e.g., MQSeries) Message Passing Interface (MPI) applications Transaction monitors (e.g., Tuxedo) Polling service (e.g., VMWare) Peer-to-peer (e.g., Skype)
Class C Incoherent-only ($\alpha = 0, \beta > 0$)	**Class D** Worst case ($\alpha, \beta > 0$)
Scientific HPC computations Online analytic processing (OLAP) Data mining Decision support software (DSS),	Anything with shared writes Hotel reservation system Banking online transaction processing (OLTP) Java database connectivity (JDBC)

Summary

- Giants don’t scale. Critical point.
- Applications don’t scale linearly, in general.
- Scalability is about sustainable size.
- Data are not divine. All measurement has errors.
- USL provides a framework in which to assess validity of your data.
- Classify application scalability.
- Performance tuning should focus on USL Zones.

Contact Coordinates



- Castro Valley, California, 94552
- www.perfdynamics.com
- perfdynamics.blogspot.com
- twitter.com/DrQz
- njgunther@perfdynamics.com
- +1-510-537-5758