hotsos
www.hotsos.com

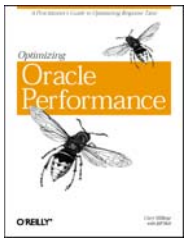
Why “System” is a Four-Letter Word

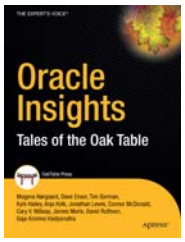
Cary Millsap (cary.millsap@hotsos.com)
Hotsos Enterprises, Ltd.
Oracle Applications User Group / Grapevine, Texas
9:45am–10:45am Thursday 16 June 2005

hotsos
www.hotsos.com

Introduction

- Cary Millsap, *hotsos.com*
- Hotsos is dedicated to Oracle system performance
  - Education – www.hotsos.com/education
  - Software – www.hotsos.com/products
  - Services – www.hotsos.com/services
- Two books
  - “OOP” for method and details
  - “TOTOT” for stories





Agenda

- Measuring “faster than”
- Amdahl’s law
- How a “faster system” can actually perform worse
- Bragan’s law
- Summary
- Your questions

How many of these have you heard?

_____ will make your system go x% faster.

- Adding more CPUs
- Upgrading to faster CPUs
- Adding more memory
- Adding a faster SAN
- Using solid-state disk
- Increasing your database buffer cache hit ratio to 99%
- Reducing your latch miss rate to 1%
- Tuning a SQL statement
- Creating an index
- Dropping an index

The problem we'll address today: Making a "system go *x*% faster" is not always a good thing.

- Two problems you have to watch out for…
  - "System is *x*% faster," but the users can't tell
    - Big waste of time, energy, money, …
  - "System is *x*% faster," but things get much worse
    - That's right: investment creates more pain

First, a definition: How do we measure "faster than"?

- I don't care how you define it, as long as you define it
- My definitions
  - *A* is *x*% faster than *B* if and only if $x = (B - A)/B \times 100\%$
  - *A* is *n* times faster than *B* if and only if $n = B/A$
- Notice that *x*% can never exceed 100%
  - Unless you can figure out how to make $A < 0$ ☺

B (before)	*A* (after)	*A* is ___ faster than *B*	
10	2	80%	5 times
3	1	67%	3 times
10	20	–100%	0.5 times

Quick quiz…

What end-user impact will be produced by a “50 times” performance improvement to inter-process communication latency?

a) 50 times better response time
b) 10 times better response time
c) No change in response time
d) It depends

d) It depends

– But upon what?

It depends upon how much the improved component was used to begin with.

- If the improved component accounted for…
  – 100% of the original response time,
  – …then the new response time will be 98% (50×) faster

Element	Old		New		Change	
	Sec	%	Sec	%		
IPC latency	10.000	100%	0.200	100%	98%	50×
all other	0.000	0%	0.000	0%	NaN	NaN
Total	**10.000**	**100%**	**0.200**	**100%**	**98%**	**50×**

It depends upon how much the improved component was used to begin with.

- If the improved component accounted for…
  - 92% of the original response time,
  - …then the new response time will be 90% (10×) faster

Element	Old		New		Change	
	Sec	%	Sec	%		
IPC latency	10.000	92%	0.200	19%	98%	50×
all other	0.870	8%	0.870	81%	0%	1×
Total	**10.870**	**100%**	**1.070**	**100%**	**90%**	**10×**

It depends upon how much the improved component was used to begin with.

- If the improved component accounted for…
  - 0.1% of original response time,
  - …then the new response time will be virtually 0% faster

Element	Old		New		Change	
	Sec	%	Sec	%		
IPC latency	10.000	0%	0.200	0%	98%	50×
all other	9,990.000	100%	9,990.200	100%	0%	1×
Total	**10,000.000**	**100%**	**9,990.200**	**100%**	**0%**	**1×**

Amdahl's law: Impact is proportional to the duration for which the improved component is used.

- Amdahl's Law
  - Response time improvement is proportional to the duration for which the improved component is used
- Some examples…
  - "The TGV [French bullet train] can go 186 mph"
    - How much time will that save me in my commute to Dallas?
  - "SSD can execute I/O 100 times faster than RAID"
    - How much time will that save a program that spends 1% of its time doing I/O?

Savepoint #1

Response time improvement for a component does not necessarily imply response time improvement for a task.

You'll never be able to predict how a task will respond unless you look at its response time profile.

What does it mean to "make your system go *x*% faster?"

_____ will make your system go *x*% faster.

- What does a statement like this mean?
  - Every program on the host goes *x*% faster?
  - One or more programs go *x*% faster (but some don't)?
    - What about the other programs?
    - Are some faster, but by less than *x*%?
    - Are some actually slower than before?
    - Is this acceptable?
- Is it acceptable to say that a "system will go *x*% faster," if some programs improve by less than *x*%?

What your users want it to mean is probably trivial to disprove.

_____ will make your system go *x*% faster.

- Your users want it to mean…
  - "Every task in the application will be *x*% faster"
- But disproving that is probably trivial
  - Find any task that's less than *x*% faster after doing ____

To you, a “system” is not the same as what your users think a “system” is.

What you think a system is…

- CPU, memory, disk, network, …?
- HP, Oracle, Apache, Java, …?
- Presentation, logic, content, …?
- GL, PO, AP, AR, HR, …?
- Key performance indicators?

What a user thinks a system is…

- “The *m* screens and *n* reports that I use to get my job done”

Different tasks on the same system respond differently to a given “tuning” attempt.

- You saw already that a 50× improvement in a component can result in…
  - 50 times better task response time
  - 10 times better task response time
  - No change in task response time

Task	% of total response time used by the improved component before “tuning”	Performance improvement	
Task A	100%	98%	50×
Task B	92%	90%	10×
Task C	0%	0%	1×

Different tasks respond differently to “tuning” because different tasks have different profiles.

- Two tasks shown at right…
  - Different profiles
  - Different reactions to “tuning”
- Note
  - The right solution in both cases was to eliminate unnecessary work

Timed event	Duration	
db file scattered read	19,051.14	69.5%
CPU service	6,889.27	25.1%
db file sequential read	1,892.70	6.9%
unaccounted-for	-405.03	-1.5%
Total	**27,428.08**	**100.0%**

Timed event	Duration	
CPU service	8,735.16	99.7%
unaccounted-for	30.00	0.3%
latch free	0.23	0.0%
db file scattered read	0.13	0.0%
Total	**8,765.52**	**100.0%**

Savepoint #2

Different tasks respond differently to “tuning” actions because different tasks have different profiles.

You’ll never be able to predict how a task will respond unless you look at its response time profile.

It's even possible for a "tuning" attempt to make a system performance problem worse.

- Below is the "after" profile
  - After upgrading to 2× faster CPUs, total response time got worse
  - That's right; performance was better before the multi-$k investment
  - What?!

Timed event	Duration		# Calls	Avg dur/call
SQL*Net message from client	984.01	49.6%	95,161	0.010 340
SQL*Net more data from client	418.82	21.1%	3,345	0.125 208
db file sequential read	279.34	14.1%	45,084	0.006 196
CPU service	248.69	12.5%	222,760	0.001 116
all other	54.33	2.7%	506	0.107 372
Total	**1,985.19**	**100.0%**		

How can improving a component make performance worse?

- Performance of your task gets worse if…
  - The "improvement" intensifies competition for the resource that is the bottleneck for your task
- The problem for this task was network competition
  - The CPU upgrade just made it worse
  - The fix took 10 minutes to implement and cost "nothing"

Timed event	Duration		# Calls	Avg dur/call
SQL*Net message from client	984.01	49.6%	95,161	0.010 340
SQL*Net more data from client	418.82	21.1%	3,345	0.125 208
db file sequential read	279.34	14.1%	45,084	0.006 196
CPU service	248.69	12.5%	222,760	0.001 116
all other	54.33	2.7%	506	0.107 372
Total	**1,985.19**	**100.0%**		

Savepoint #3

It's possible for improving a component to make your task's performance even worse.

You'll never be able to predict how a task will respond unless you look at its response time profile.

Bragan's law: Sometimes, statistics aren't a reliable way to look at a problem.

- Bragan's law
  - "Say you were standing with one foot in the oven and one foot in an ice bucket. According to the percentage people, you would be perfectly comfortable."

It is dangerous to try to communicate quantitatively about a "system" as if it were a single unit.

- Saying anything about the performance of a "system" is probably wrong
- Even something like this isn't good enough…
  - "99.9% of users are now ecstatic about system performance."
- What if the most important 10 of your 10,000 customers now have worse performance?

Savepoint #4

You'll never be able to predict how a task will respond to "tuning" unless you look at its response time profile.

Summary points…

> "Systems don't have performance; tasks have durations."
>
> —Mike Ryan

- Communicating about "system performance" as if it were a single unit is misleading and harmful to the decision-making process…
  - It's bad communication
- Useful communication about system performance recognizes the importance of individual task durations
  - Tasks can respond very differently to a given "tuning" action
- To make informed decisions about improving performance, you need to study your tasks' response time profiles