Why you can’t see your real performance problems

Cary Millsap (cary.millsap@hotsos.com)
Hotsos Enterprises, Ltd.
Northern California Oracle Users Group / San Francisco
9:30am–10:30am Thursday 2 November 2006

Agenda

- Two distinct modes of operational management
- Diagnosis and repair: how to begin
- Why performance improvement projects fail
- Skew in diagnostic data
- The whole story in under a minute

Two distinct modes of operational management

Routine maintenance versus diagnosis and repair
A domestic analogy…

**Normal day**
- Critical areas
  - Work down the to-do list
  - Maybe nothing
- So…
  - Reorganize closet
  - Dust tops of door facings
- What matters
  - You’re content
  - Maybe you plan ahead

**Big party tonight**
- Critical areas
  - Living room/kitchen
  - Guest bathroom
- So…
  - Focus on guest’s experience
  - NOTHING ELSE
- What matters
  - The party
  - Tonight

---

**MAINTENANCE**

**REPAIR**

---

How can you tell when you’re in system maintenance mode versus diagnosis and repair mode?

**Normal day**
- You look for problems
- A computer says what’s wrong
- Laissez faire, calm
- V$/$X$ dashboards, Statspack, ADDM, AWR, ASH, etc. can give you something to do

**Performance problem**
- Problems look for you
- A person says what’s wrong
- Micromanagement, tension
- The tools you know and trust can lead you down months-long rat holes

---

**MAINTENANCE**

**REPAIR**
Review: Which mode of operation does each picture represent?

Diagnosis and repair  Routine maintenance

Recap

- Routine maintenance is what you do during periods of calm.
- Diagnosis and repair is what you do in response to a problem.

Problem diagnosis and repair mode requires better diagnostic data than you probably use in routine maintenance mode.
Diagnosis and repair: how to begin

Why you should be looking at tasks instead of resources

A sequence diagram illustrates what's going on inside your system.
Definition of *bottleneck*...

- The resource with the highest utilization is called the *bottleneck*.

Which resource on this system is the bottleneck?

- It’s easy
  - DB
    - Least idle time
    - Highest utilization
But what if the red task is what you need to fix?

<table>
<thead>
<tr>
<th>Resource</th>
<th>Duration (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>61 57.0%</td>
</tr>
<tr>
<td>DB</td>
<td>18 16.8%</td>
</tr>
<tr>
<td>LAN</td>
<td>12 11.2%</td>
</tr>
<tr>
<td>DB queue</td>
<td>8  7.5%</td>
</tr>
<tr>
<td>AS</td>
<td>8   7.5%</td>
</tr>
<tr>
<td>Total</td>
<td>107 100.0%</td>
</tr>
</tbody>
</table>

Then either eliminate the red task’s use of OS, or make OS respond more quickly.

Of course, improving the speed of DB may speed up the blue task…

<table>
<thead>
<tr>
<th>Resource</th>
<th>Duration (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>40 50.6%</td>
</tr>
<tr>
<td>AS</td>
<td>20 25.3%</td>
</tr>
<tr>
<td>LAN</td>
<td>10 12.7%</td>
</tr>
<tr>
<td>DB queue</td>
<td>9  11.4%</td>
</tr>
<tr>
<td>OS</td>
<td>-     0.0%</td>
</tr>
<tr>
<td>Total</td>
<td>79 100.0%</td>
</tr>
</tbody>
</table>

…but if the blue task is not the issue you need to fix, then who cares?
Definition of bottleneck (reprise)...

- The resource with the highest utilization is called the bottleneck.

  True, but misleading, because it omits context.

- The bottleneck is the component where a transaction spends most of its time.
  - [Menascé and Almeida (2002), p10]

  Much better.

Recap

- Your business needs you to prioritize your performance repairs by business need.
  - ...Which doesn’t necessarily map to what your resource monitors say.
- Analyze your tasks in business priority order.
  - ...So you’ll actually fix what needs fixing.

Make your system work for your business, not the other way around.
Why performance improvement projects fail

Every failed performance project I’ve witnessed since 1989 has had the same root cause.
Some classic examples of failed performance improvement projects…

- Three examples
  - SQL tune-up didn’t help
  - Disk tune-up didn’t help
  - CPU upgrade didn’t help

Example 1: SQL tune-up didn’t help…

- Effort
  - Tool says “bad SQL”
  - Tune “top SQL”

- Result
  - Key task no perceptibly faster

What happened?!

<table>
<thead>
<tr>
<th>Resource</th>
<th>Duration before (seconds)</th>
<th>Duration after (seconds)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS and LAN</td>
<td>8</td>
<td>8</td>
<td>0%</td>
</tr>
<tr>
<td>SQL</td>
<td>2</td>
<td>1</td>
<td>50%</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>9</td>
<td>10%</td>
</tr>
</tbody>
</table>

We improved a resource that this task didn’t really use.
Example 2: Disk tune-up didn’t help…

- Effort
  - Tool says “too much I/O”
  - Buy, install 2x faster SAN

- Result
  - Key task no perceptibly faster

What happened?! It worked last time...

<table>
<thead>
<tr>
<th>Resource</th>
<th>Duration before (seconds)</th>
<th>Duration after (seconds)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>96</td>
<td>96</td>
<td>0%</td>
</tr>
<tr>
<td>Disk</td>
<td>4</td>
<td>2</td>
<td>50%</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td>98</td>
<td>2%</td>
</tr>
</tbody>
</table>

We improved a resource that this task didn’t really use.

Example 3: CPU upgrade didn’t help…

- Effort
  - Tool says “CPU bound”
  - CPU upgrade 500MHz → 1GHz

- Result
  - Key task noticeably slower

What happened?! We spent $30,000...

<table>
<thead>
<tr>
<th>Resource</th>
<th>Duration before (seconds)</th>
<th>Duration after (seconds)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>750</td>
<td>1,250</td>
<td>-67%</td>
</tr>
<tr>
<td>CPU</td>
<td>250</td>
<td>125</td>
<td>50%</td>
</tr>
<tr>
<td>Total</td>
<td>1,000</td>
<td>1,375</td>
<td>-38%</td>
</tr>
</tbody>
</table>

We improved a resource that this task didn’t really use.
Recap

- Different tasks respond differently to a given tune-up.
- Fixing the wrong thing first…
  - Wastes your time.
  - Can actually make performance worse.

A task’s profile uniquely determines its response to a given tune-up.

See “Why ‘system’ is a four-letter word,” from NoCOUG Spring Conference in Sunnyvale, 19 May 2005.

Undiagnosed skew is the cause of every failed performance project I’ve witnessed since 1989.
Skew in diagnostic data

What is skew?

- Imagine...
  - You’re in a group of 1,000 people
  - You get “any kind of ice cream you want”
  - You want orange (in fact, you’re allergic to chocolate)
  - 1,000 people surveyed: 999 want chocolate
  - So you get chocolate
  - How happy are you?

*Provider’s measurement: 99.9% happiness rate.*
What is skew?

- Skew is a non-uniformity in your data
  - The “red rock problem” [Millsap and Holt (2003), p8]

- Examples of lists with $\Sigma = 10$, $n = 5$...
  - 2, 2, 2, 2, 2 no skew
  - 2, 2, 3, 1, 2 low skew
  - 0, 0, 10, 0, 0 high skew

Skew can trick you any time you assume uniformity in a list without proving it first.

System = list of tasks

Task = list of resource consumptions

Resource consumption = list of calls (e.g., to DB or OS)

Call = list of instructions
Skew can ruin your project on any level in your diagnostic data hierarchy.

- **System** – profiles across tasks can be non-uniform
  
  Task A is 80% disk I/O; task B is only 2% disk I/O.

- **Task** – profiles within a task can be non-uniform
  
  Task A is CPU and latch bound at 2pm, but not at 8am.

- **Consumption** – call durations can be non-uniform
  
  742nd parse call consumes 90% of total parsing duration.

- **Call** – instruction durations can be non-uniform
  
  Line 7972 consumes 99% of total program duration.

Skew at the level of “system = non-uniform list of tasks” is what ruined each of the 3 earlier example projects.

- **Three examples**
  - SQL tune-up didn’t help
  - Disk tune-up didn’t help
  - CPU upgrade didn’t help

  In each example, the system’s "bottleneck" was not the important task’s bottleneck.
How to keep skew from wiping you out…

- Any time you’re considering a list, inquire about skew among the elements

If your tool doesn’t let you drill into the elements, then you’re using the wrong tool.

There are lots of interesting skew questions.

- Some skew questions of interest…
  - Does my task have the same kind of profile as another task?
  - Does my task always have the same kind of profile?
  - Are all my task’s parses the same duration?
  - Which buffer busy waits reason? Which block?
  - Which latch?
  - How many call durations are like my worst call duration?
  - Which db calls account for the most preemption time?
  - …
Most people don’t know how important skew is because the tools they use don’t show it.

- Oracle fixed views don’t take you very far
  - Session based, not task based
  - No data for db calls, unaccounted-for time

This is why ADDM, AWR, ASH, Statspack, and tools based on V$ data or SGA polling are inadequate for problem diagnosis.

- Raw Oracle trace files get you all the way down to the call layer
- DBMS_PROFILER can get you to the (PL/SQL) instruction layer

Recap

- To detect skew, you have to drill beneath the aggregations.
- Skew is a common problem because most Oracle diagnostic data sources conceal it.
- You can detect skew
  - ...down to the call layer in raw profile data.
  - ...and down to the instruction layer with DBMS_PROFILE.

Skew is why Method R prescribes using Oracle extended SQL trace data.
The whole story in under a minute…

You probably use aggregated data all the time in routine operational maintenance.
But you can’t detect skew in aggregated diagnostic data.

And skew can wipe out your diagnosis and repair project.
That's why so many Oracle performance improvement projects fail.

...And why so many companies have systems that are way bigger and more complicated then they should be.
When you’re diagnosing a performance problem, you need to focus on task response times.

…And you need to drill down to see exactly where the time went.
You can't do that with Oracle v$ data.

…Or any tool that's based on Oracle v$ data.
...Even if it polls directly from the Oracle SGA.

But you can do it with Oracle trace data.
…Which is why trace files figure so prominently in the Method R prescription for Oracle.

Method R gives you the power to make fully-informed decisions about performance.
…Which is why Method R projects so often fix problems in a few minutes.

…Even problems that have plagued their owners for months or even years.
Thank you