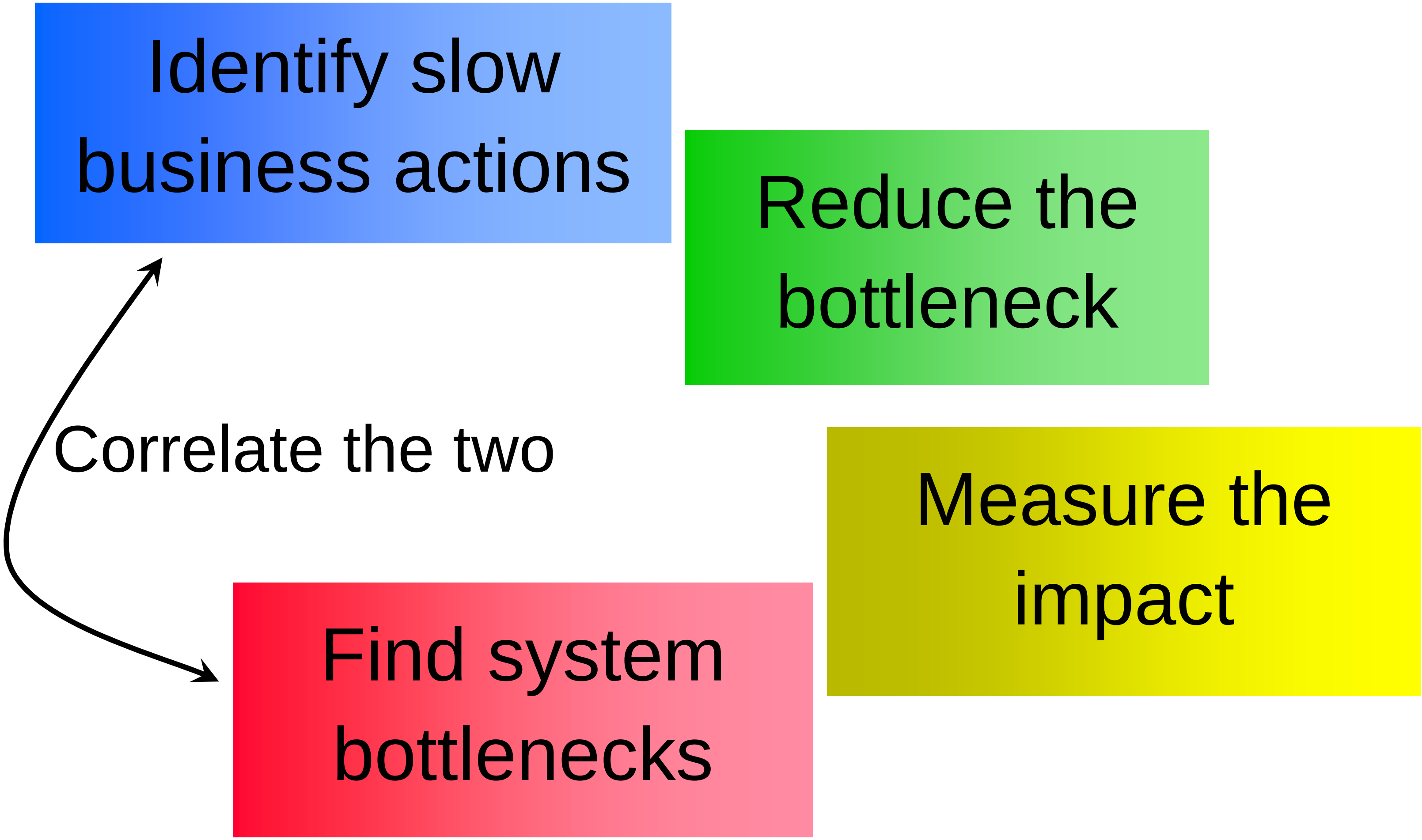
Common Performance Monitoring Mistakes

Virag Saksena
CEO
Auptyma Corporation

peakperformance@auptyma.com

Tuning Approach – BUS X SYS



Identify slow business actions

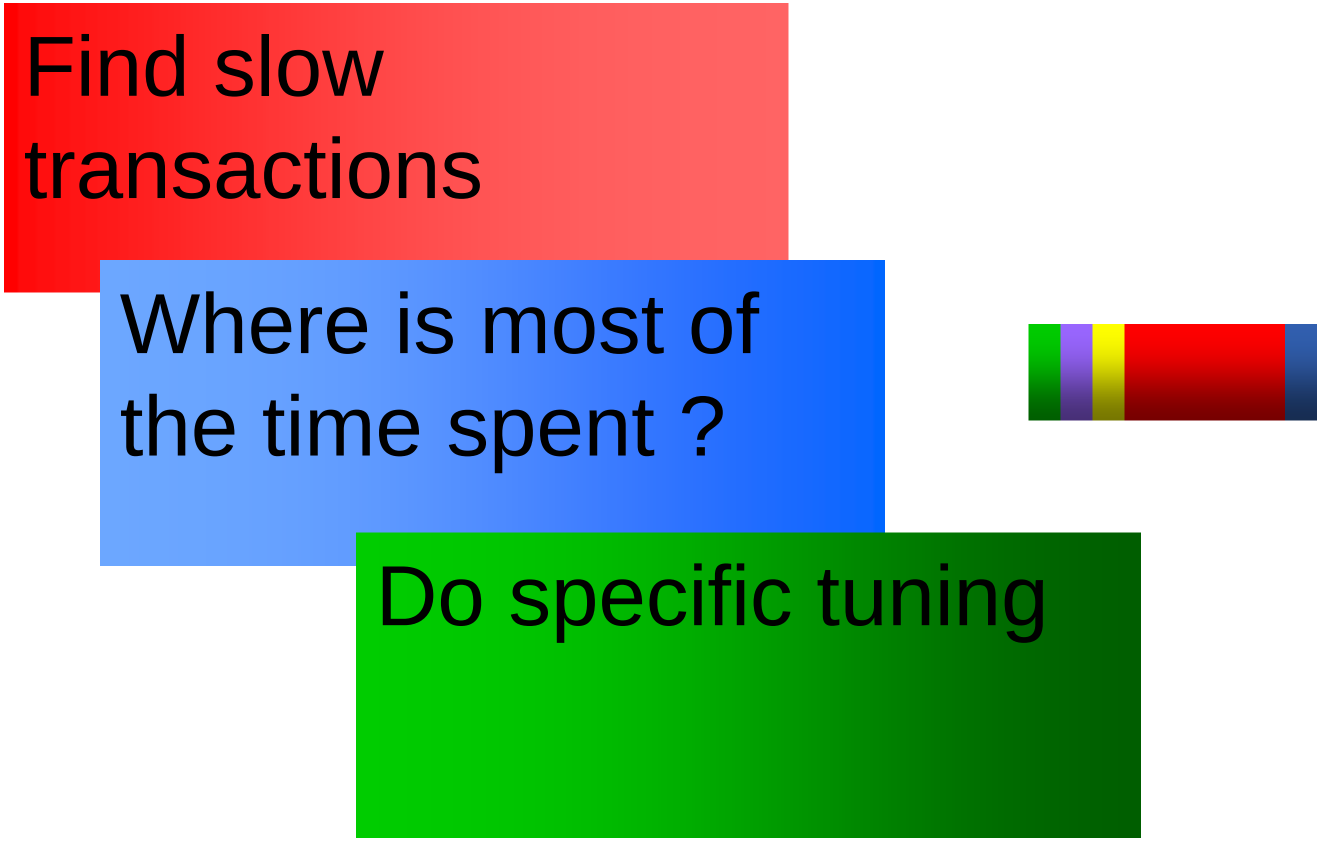
Reduce the bottleneck

Correlate the two

Measure the impact

Find system bottlenecks

Tuning Approach BUS2SYS



Find slow transactions

Where is most of the time spent ?

Do specific tuning

Tuning Approach SYS2BUS

Identify system bottlenecks

Find affected transactions

Reduce the bottleneck

I don’t have any CPU left

19:23:48	**%usr**	**%sys**	**%wio**	**%idle**
19:23:53	11	7	76	6
19:23:58	11	10	79	0
19:24:03	11	7	82	0
19:24:08	12	8	79	1
19:24:13	10	5	85	0
Average	**11**	**7**	**80**	**1**

19:27:17	**%usr**	**%sys**	**%wio**	**%idle**
19:27:22	40	21	4	35
19:27:27	43	13	1	44
19:27:32	42	19	1	38
19:27:37	35	14	0	51
19:27:42	42	19	1	38
Average	**40**	**17**	**1**	**41**

- **Which system is using more CPU ?**

CPU Bottleneck - Symptoms

- **sar, vmstat, other tools report high CPU usage**
- **System is slow**
- **Run-queue is high**

CPU Bottleneck - Analysis

- Is this an Oracle issue ?
- Which processes are using CPU
  - Large amount of CPU Usage
  - Runnable processes
- Look inside the database

```
select s.sid, s.value
  from v$sesstat s, v$statname n
 where s.statistic# = n.statistic#
   and n.name = 'CPU Used by this session'
 order by 2 desc
```

```
select ...
  from v$session_wait w, v$session s
 where s.sid = w.sid
   and w.wait_time <> 0
   and s.status = 'ACTIVE'
```

Relating the processes using the CPU to the session which is running

```
select …
  from v$session s, v$process p
 where p.addr = s.paddr
   and p.spid = :my_cpu_hog
```

Is CPU a direct or indirect problem is it my or someone else's problem

- **CPU is a road – you have to drive from LA to San Francisco at night**
- **Your transaction is using large amounts of CPU**
- **CPU is a road – you have to cross 101/85 interchange at 4 PM in the evening**
- **Other transactions are using large amounts of CPU, you are just stuck waiting for it to be free**

I do not have free memory

- **Do not look at free memory**
- **Look at page scans/sec**
- **When you have a memory bottleneck this number will start going up**

Stale/missing statistics

- **number one cause for poor execution plans (and higher resource usages)**
- **Will cause the optimizer to make incorrect decisions**
- **Gather statistics whenever data volumes/ distribution change significantly**

Table with stale statistics

Table	Rows		Distinct Keys	
	Actual	**Stats**	**Actual**	**Stats**
LINES	**1.36M**	**461**		
LINES_N1(ORG_ID, ORDER_NUMBER, PART_NUMBER)	**1.36M**	**455**	**1.36M**	**455**
LINES_N2(ORG_ID, PART_NUMBER)	**1.36M**	**455**	**422**	**211**
ORDERS	**3247**	**3178**		
ORDERS_N1(ORG_ID, CUSTOMER)	**3247**	**3247**	**806**	**806**

With and Without the statistics

```
select o.order_number, l.revenue
  from lines l, orders o
 where o.customer_number = :b1
   and o.org_id = :b2
   and o.order_number = l.order_number
   and l.org_id = :b3
```

```
  Rows  Row Source Operation
   211  HASH JOIN
     1   TABLE ACCESS BY INDEX ROWID ORDERS
     2    INDEX RANGE SCAN ORDERS_N1
684695   TABLE ACCESS BY INDEX ROWID LINES
684696    INDEX RANGE SCAN LINES_N2
```

```
  Rows  Row Source Operation
   211  NESTED LOOPS
     2   TABLE ACCESS BY INDEX ROWID ORDERS
     2    INDEX RANGE SCAN ORDERS_N1
   211   TABLE ACCESS BY INDEX ROWID LINES
   212    INDEX RANGE SCAN LINES_N1
```

The Impact

Before

```
call     count    cpu  elapsed  disk   query  current    rows
------- ------  ----- -------- ----- ------- --------  ------
Parse        1   0.00     0.01     0       4        0       0
Execute      1   0.00     0.00     0       0        0       0
Fetch       16   7.75     7.83   494  325874        0     211
------- ------  ----- -------- ----- ------- --------  ------
total       18   7.75     7.84   494  325878        0     211
```

After

```
call     count    cpu  elapsed  disk  query  current    rows
------- ------  ----- -------- ----- ------ --------  ------
Parse        1   0.00     0.00     0      0        0       0
Execute      1   0.00     0.00     0      0        0       0
Fetch       16   0.01     0.01     3     37        0     211
------- ------  ----- -------- ----- ------ --------  ------
total       18   0.01     0.01     3     37        0     211
```

High buffer cache hit ratios = good

- **What cache hit ratio do you expect in a database with only full table scans ?**
- **30% ? 60 % ? 90 % ?**
- **93% for db_file_multiblock_read_count = 16**
- **Access to buffered blocks bring ratios up**
- **However ratios which are too high might indicate another problem**

SQL Statement

- **Lines has 20M Rows**
- **Orders has 1M Rows**
- **50% of the lines are LICENSE**
- **Avg orders/customer = 10**

```
select o.order_number, l.amount
  from orders o, lines l
 where l.order_number = o.order_number
   and o.customer_number = :b1
   and l.line_type = 'LICENSE'
```

Execution Plans

Rows		Cache Hit Ratio 95.34
Returned	Accessed	Operation
100		Nested Loop
10	10	Table Access by Index ROWID Orders
10	11	Index Range Scan Orders(CUSTOMER_NUMBER)
100	200	Table Access by Index ROWID Lines
200	210	Index Range Scan Lines(ORDER_NUMBER)

Rows		Cache Hit Ratio 93.78
Returned	Accessed	Operation
100		Nested Loop
10	1M	Table Access FULL Orders
100	200	Table Access by Index ROWID Lines
200	210	Index Range Scan Lines(ORDER_NUMBER)

Rows		Cache Hit Ratio 99.99
Returned	Accessed	Operation
100		Nested Loop
10M	10M	Table Access by Index ROWID Lines
10M	10M+1	Index Range Scan Lines(LICENSE_TYPE)
100	10M	Table Access by Index ROWID Orders
10M	10M	Index Unique Scan Orders(ORDER_NUMBER)

Run txn on a large rollback segment to avoid Snapshot too Old

- **Snapshot too old happens because other txns are changing data you are reading**
- **After changing the data they commit**
- **Eventually the undo gets over-written**
- **You need to prevent the undo for committed transactions from being overwritten**
- **Specify UNDO_RETENTION in 9i+**
- **It should be as long as the expected query duration**
- **Need enough space in the tablespace**
- **Alternatively increase all the rollback segments**

Look at numbers too, not just ratios

Redo Copy Latch

- Miss Rate : 35% - Is this a problem ?
- Immediate Miss Rate : 0.12 %

Gets	156	Immediate Gets	110,634
Misses	55	Immediate Misses	140

Full Table scans are bad

- **Full table scan can provide superior IO performance when more than X% of table blocks are accessed by a query**
- **X can be as low as 6.25 % for db_file_multiblock_read_count = 16**

Break up of an IO Call	
Rotational Latency at 10000 RPM	3 ms
Head seek time	5 ms
8 KB Data Xfer at 100 MB/sec	0.08 ms
Total 8 KB IO Service Time	**~ 9 ms**
128 KB Data Xfer at 100 MB/sec	1.28 ms
Total 128 KB IO Service Time	**~ 10 ms**

Full Table scans are bad

Consider a table with following statistics

- **Rows : 1.25 Million**
- **Blocks under High Water Mark : 25000**
- **Height of Index : 2**
- **Leaf Blocks : 4000**

- **For a query accessing 10% of the rows**
- **Min number of blocks will be 10%**
- **With 50 rows/block, we could even access 100% of the blocks**
- **Full Table scan will need 25000/16 = 1563 IO calls**

Index vs Full Table Scan for 10% rows

Expected Table blocks accessed	10%	30%
Index Leaf Block PIO Calls	400	400
Table Block PIO Calls	2500	7500
Expected Total PIO Calls	2900	7900
Number of Rows Returned	125K	125K
Approx PIOs with 95% CHR	6250	6250
Index IO Time with 10 ms IO calls	**29 sec**	**79 sec**
Expected PIO Calls with FT Scan	1563	1563
FT Scan IO Time with 12 ms IO calls	**18 sec**	**18 sec**

Corollary – Hash Joins, Anti Joins are bad

- **When accessing large %ages of table data, Hash Joins offer superior performance to Nested Loop Joins, or sub-queries**
- **Similarly Hash Anti Join can offer better performance as opposed to the Not exists correlated sub-query**

Prioritize on time waited /sec

- Specially true when comparing IO waits, latch waits, enqueues, CPU Usage
- On a 12 CPU box, you have 12 seconds of CPU available every second
- IO Wait is order of 10ms
- 12 seconds of IO Wait = 1200 IO/sec
- Latch misses is order of microsec
- Latch sleeps is order of 10ms
- Locks are order of seconds

If it is not fast enough, add parallelism

- **Add parallel hint**
- **Increase the number of batch jobs**
- **Doing this will increase the load on the system**
- **Often the solution might be rearchitecture**
  - **Do not use row by row processing**
  - **Use set operations or single SQL statement as opposed to fetching and processing data row by row**
  - **Build validation logic into the statement**

Upgrade to larger size disks

- **Often upgrading to larger size disks also means upgrading to fewer disks**
- **Moving from 100 x 9 Gig disks to 50 x 36 Gig disks**
- **If you are doing 6,000 IO/seconds, earlier you would have 60 IO/sec/disk**
- **With 10ms/IO, this would run to about 60% utilization**
- **On the new configuration you would go to 120 IO/sec/disk !!**
- **Disk speeds have not increased that much**

Sizing by disk usage, not by IO rates

- **10,000 IO Calls/sec, 90% Read calls**
- **On a RAID 5 scenario, this translates into 10,000 Read Calls, 2,000 Write Calls for a total of 12,000 IOs/sec**
- **For a 60% utilization we'd need 200 disks**
- **On a RAID 0+1, we get 9,000 Read Calls, 2000 Write Calls for a total of 11,000 IO/sec**
- **For a 60% utilization we'd need 183 disks**
- **However 183 disks represent capacity of 92 disks, while the 200 disks in RAID 5 represent capacity of 160 disks**
- **If we had sized by disk capacity, RAID 5 would have given us only 57% performance for RAID 0+1**

About us

- **Virag Saksena has over 12 years of experience in Oracle and Sytem Performance. He was Director, Performance Group for Oracle's CRM Products Division**
- **His company Auptyma Corporation http://www.auptyma.com is focused on helping customers with with Java and Oracle performance problems**
- **You can reach him at peakperformance@auptyma.com**