My PX Goes to 11

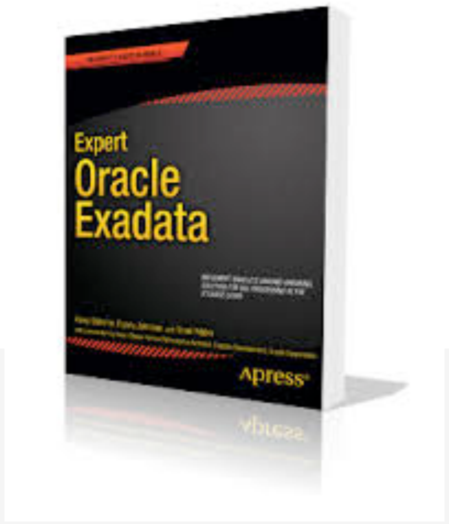
Kerry Osborne – Enkitec

Chris Wones - dunnhumby

E4 2014

About Me (Kerry)

- Working with Oracle since V2 (1982)
- Working with Exadata since V2 (2010)
- Co-author of Expert Oracle Exadata & Pro Oracle SQL
- I Work for Enkitec!



About Me (Chris)

- Working with Oracle since 1992
- Working with Exadata since 2012
- I work for dunnhumby!
- I have a party boat!

Obligatory Marketing Slide

- Enkitec is Awesome!

Obligatory Marketing Slide

- dunnhumby is also Awesome!

What's the Point?

- Big Data Warehouse
- Running on SAND DB (column oriented)
- Conversion to Exadata (X3-8)
- Pay Down Technical Debt
- Wanted consolidation from many data marts
- Needed Strong Vendor (Oracle) and partner (Enkitec)
- Security, Innovation, and Performance (SIP)

Teamwork

- Joint Effort
  - dunnhumby and Enkitec
- Already had good relationship
- Both teams good technically
- dunnhumby provided domain knowledge
- Enkitec provided outside view

- Team members:
  - dunnhumby: David, Michael, Jay, Springers, Rashmi, …
  - Enkitec: Tanel, Karen, Alex, Kerry, …
  - Oracle: Maria, Tom, Hermann, Sue, …

Original Project Goals

- Verify Exadata Configuration was Optimal
- Individual Report Performance
- Overall Report Throughput
- Recommend Additional Hardware

Original Project Goals

- Configuration
  - Just wanted make sure configuration was optimal
  - As you know, Oracle is very configurable!

Original Project Goals

- Individual Report Performance
  - 90% of “10% sample” reports finish within 15 minutes
    - (queue time + SQL run-time + serialisation time)
  - 90% of “100% sample” reports finish within 50 minutes
    - (queue time + SQL run-time + serialisation time)

Original Project Goals

- Report Throughput
  - Be able to run 1,000 reports per day
  - 977 was highest volume ever run on old system
  - Test Cycle was 12 hours

Original Project Goals

- Recommend Additional Hardware
  - I’ve never been asked ahead of time to recommend an alternative in case of failure. ☺
  - Probably should have been scared
  - But we’re always optimistic!

Some Numbers to Start With

- DB - 11.2.0.3
- cellsrv – 11.2.3.2.1 (version before auto flash cache scans)
- Best 12 hour throughput test was 224
- Biggest table was 4TB+
- Many other tables in 0.5-1TB range
- Platform was 4 node cluster (2 Exadata X3-8's)
- Reports were heavily parallelized
- Using HCC Heavily
- Using Smart Scans Heavily
- Write Back Cache On
- Top SQL was mostly CTAS
- Some SQL Doing 100M+ gets and 150M+ pio

Initial Challenges - business

- Restatement of card to household relationship
- Product filter groups unknown
- Date ranges not always standard
- Segmentations defined using request criteria

Initial Challenges - technical

- Extremely Large Data Set
- No Pre-Aggregation
- Throughput was way lower than expectations
- Temp Usage was biggest bottle neck
- Variability of DOP was biggest frustration
- Code was still being modified

It Was a Little Out of Control!

3 Pronged Approach:

Get DOP Settled Down
- consistent DOP
- eliminate downgrades

Remove Bottle Necks
- start with Temp
- move to next one

Improve Code
- to do less work

•Note that we focused on throughput

Digression – Tale w/ Multiple Story Lines

- Initial Architecture / Configuration
- Bottleneck Removal / Avoidance
- Coding Changes
- Oracle Features

Initial PX Setup

parallel_adaptive_multi_user	FALSE
parallel_degree_limit	24
parallel_degree_policy	AUTO
parallel_force_local	TRUE
parallel_max_servers	1280
parallel_min_servers	96
parallel_servers_target	960
parallel_threads_per_cpu	1

- X3-8 - 80 cores (for most of the project we had it capped at about 65%)

Initial DOP Control

AutoDOP
Tables Defined with DEFAULT
Tables Defined with specific Degrees
Hints used in Statements
Alter Session used to control PX
RESMGR used to control PX

Downgrades Occurring?

DOP Control – Too Many Knobs?

Yes there are a lot of knobs!

AutoDOP is an effort to fix this!

Evaluating Potential DOP Setups

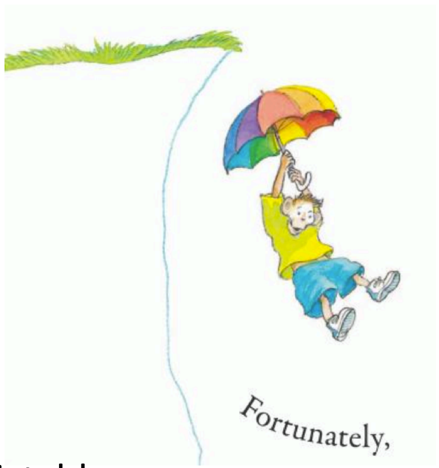
Goal No. 1 – Improve stability
Needed to decide quickly
Came Up with 3 Options

Potential Setup #1 - Classic

- Policy = Manual
- Control with Hints (or alter session)
- Turn Everything Else Off
- Tried and True
  - Has Worked in Past on Big Systems
- PX queuing = True

Digression – PX Queuing

- I'm a Big Fan!
- Unfortunately it's Linked to autoDOP
- Fortunately there is a separate parameter for it
- Unfortunately it's a hidden parameter
- Fortunately Some Brave Souls at Oracle Have Documented It



•Seems to be some debate inside Oracle about _parallel_statement_queuing
•In Memory Parallel can short circuit smart scans making performance unpredictable

Potential Setup #2 - LTD

Policy = Limited
Big Tables Degree = Default
Turn Everything Else Off
PX queuing = True

Potential Setup #3 - Jetson

Policy = Auto
Turn Everything Else Off
It's the Wave of the Future
But it's Not Widely Accepted
And It's Not Well Documented
But Some Day We'll All Have Jetpacks!

- Note that setting policy=Auto turns on In Memory PX as well
- Note also that Auto mode can kick in when policy = manual

We Picked Classic!

```
parallel_degree_policy  = MANUAL
parallel_force_local = FALSE
_parallel_statement_queuing = TRUE
```

Mitigating Factors

- Time was short

Further DOP Challenges

Still Too Much of a Good Thing!

- -Eliminate PX for SQL accessing < 1M rows
- -Reduce DOP in general
- -Differentiate between query and create on CTAS

Still Lot's of Downgrades

- -Confusing because setting should have eliminated them
- -Forensics difficult
- -SQL Monitor has data but it ages out quickly

*^$%#% Downgrades!

Setup
- -parallel_adaptive_multi_user=false
- -Queuing On (target set below max)

Shouldn't have downgrades
- (unless single stmt asks for more than max)
- (or bugs)

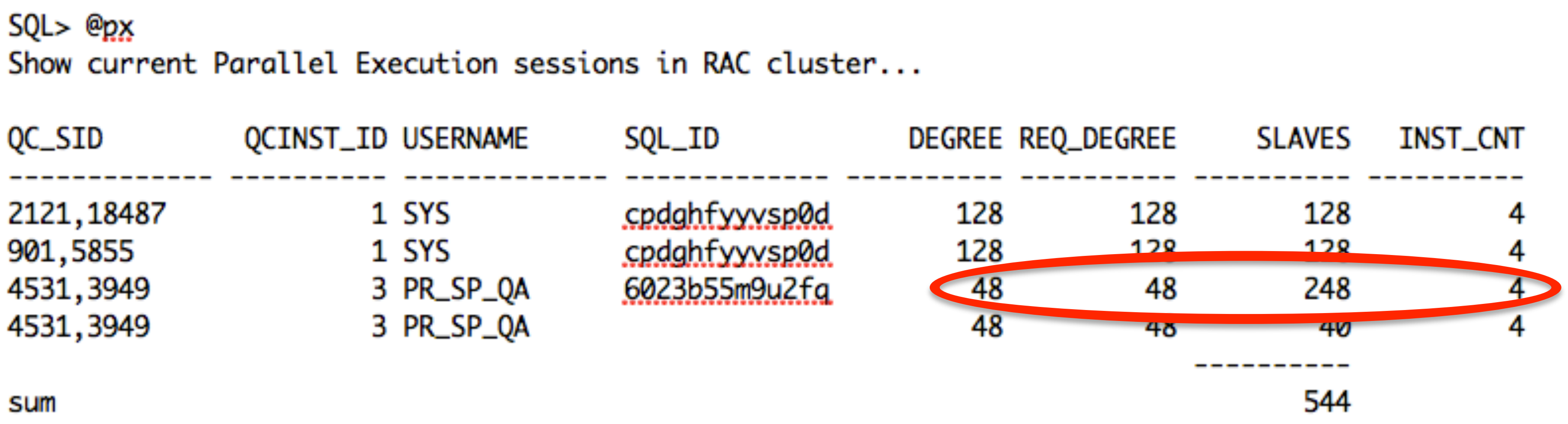
Difficulty in tracking slaves/execution
- started capturing v$pq_sesstat after each statement
- Realized CTE's and Unions causing multiple DFO's
  - - resulting in multiple sets of slaves (bug)

- Note that resmgr only limits DOP, not number of slaves (in 11g)

enkitec

*^$%#% Downgrades!

Using gv$px_session (via Tanel's handy px.sql script):

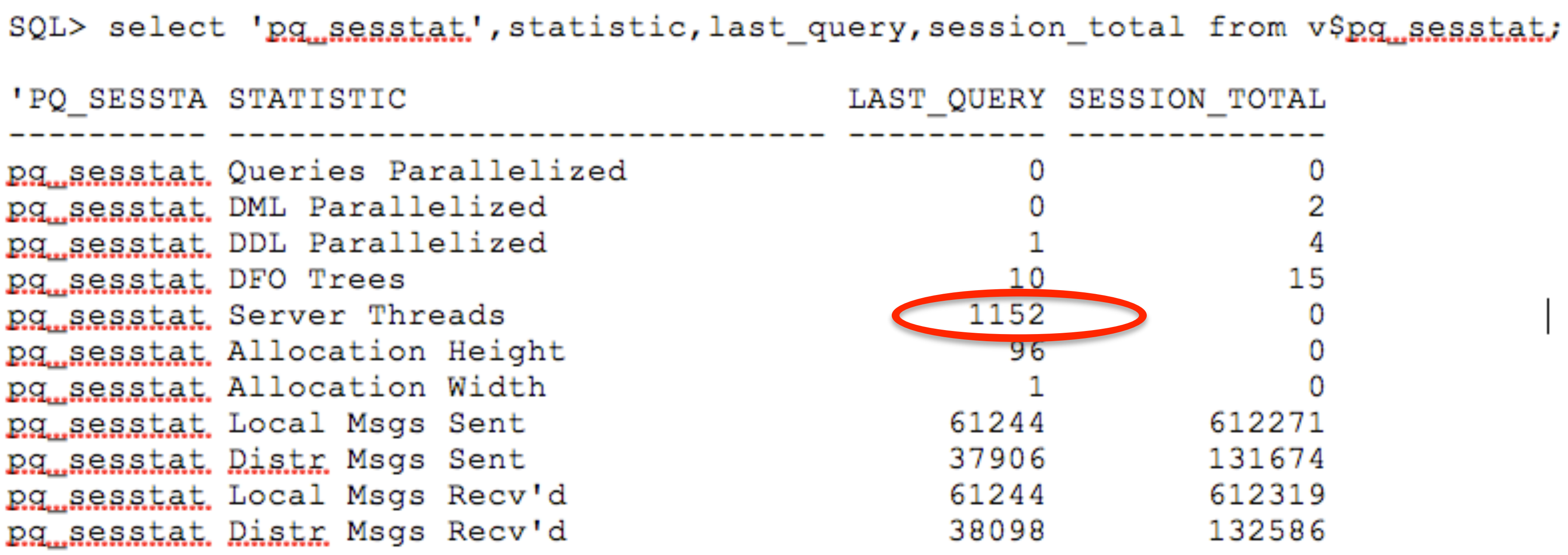


```
SQL> @px
Show current Parallel Execution sessions in RAC cluster...

QC_SID          QCINST_ID USERNAME      SQL_ID            DEGREE REQ_DEGREE     SLAVES   INST_CNT
-------------- ---------- ------------- ------------- ---------- ---------- ---------- ----------
2121,18487              1 SYS           cpdghfyyvsp0d        128        128        128          4
901,5855                1 SYS           cpdghfyyvsp0d        128        128        128          4
4531,3949               3 PR_SP_QA      6023b55m9u2fq         48         48        248          4
4531,3949               3 PR_SP_QA                            48         48         48          4
                                                                               ----------
sum                                                                                   544
```

- 248's not too bad, but there we're many that were over 1000

*^$%#% Downgrades!

Using v$pq_sesstat:



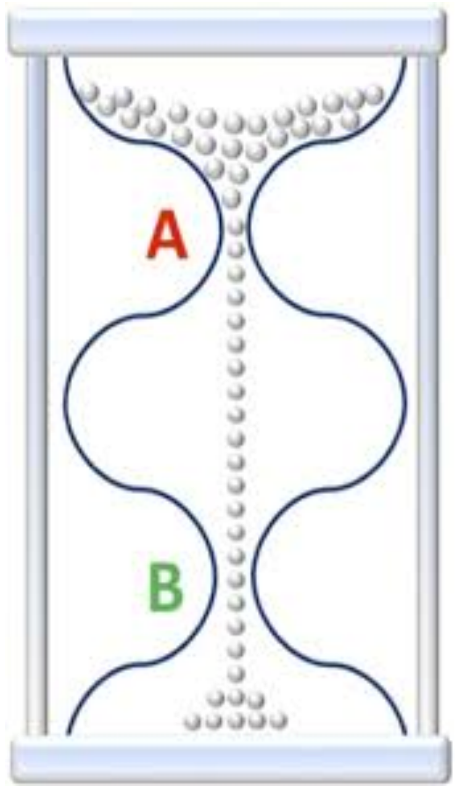
```
SQL> select 'pq_sesstat',statistic,last_query,session_total from v$pq_sesstat;

'PQ_SESSTA STATISTIC                      LAST_QUERY SESSION_TOTAL
---------- ------------------------------ ---------- -------------
pq_sesstat Queries Parallelized                    0             0
pq_sesstat DML Parallelized                        0             2
pq_sesstat DDL Parallelized                        1             4
pq_sesstat DFO Trees                              10            15
pq_sesstat Server Threads                       1152             0
pq_sesstat Allocation Height                      96             0
pq_sesstat Allocation Width                        1             0
pq_sesstat Local Msgs Sent                     61244        612271
pq_sesstat Distr Msgs Sent                     37906        131674
pq_sesstat Local Msgs Recv'd                   61244        612319
pq_sesstat Distr Msgs Recv'd                   38098        132586
```

- 1152 = 96x12,
- if target is 1000 and max is 1200, 2 of these would run and 2nd would be downgraded

Bottlenecks

But back to Winterfell . . .

Spilling to Temp
- added as much PGA memory as possible
  - see Alex’s presentation, Hotsos 2014
- increased PX for worst offenders
- proposed moving to ZFS
  - see Alex’s presentation, E4 2014
- of course doing less work is best
  - changes to SQL most effective tool

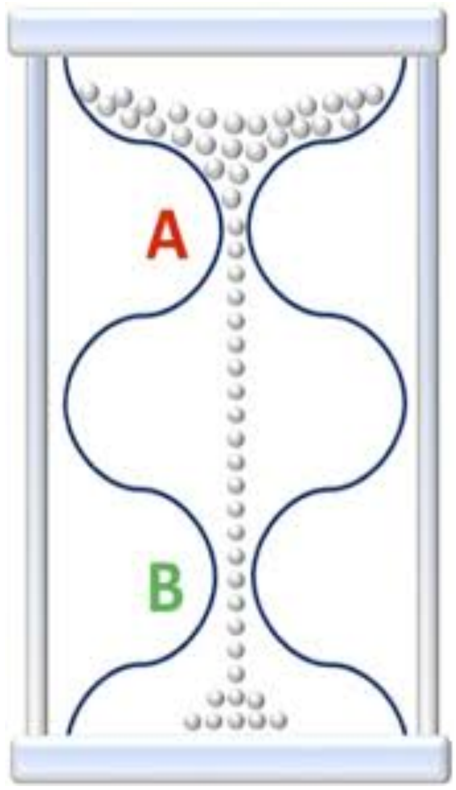


- The fastest way to do anything is not to do it. ~ Cary Millsap

Bottlenecks

Contention

- Overloading CPUs
- AMM
- seg$
- HWM Brokering
- Temp file header (enq: SS)
- CF enque
- etc…



Coding

Now Back to King’s Landing . . .

Limit DOP
Re-write to eliminate CTE’s and Union Alls
- To avoid multiple sets of slaves
- To eliminate multiple passes through the same table

Use EXIST clause rather than joining
Separate Joins and Aggregations
Use Advanced Grouping (Cube, Rollup, Grouping Sets)
Experiment with NDV Synopsis
Reconsider Pre-Aggregation
Grouping via chunking

Oracle Additions

Now Back to the Wall . . .

Aggressive Bloom Filters
General Education on PX and Resmgr
NDV – hyperloglog algorythm - approx_count_distinct

Results (to date)

As of March - 1700+ reports in 12 hours = 3400 / 24 hours

- ~3.5X the initial throughput goal

In Production Now

As of last week (from AWR) the system was spending

> > 50% dbtime on cell smart scan
>
> ~ 20-30% - read / write temp
>
> < 10% CPU

Customer Feedback – Holy &^%!

Things to Come

LTD and Maybe even Jetson
In Memory (12c)
- X3-8 has 2 TB RAM

Move temp to ZFS and free up some PGA
NDV enhancements (PL/SQL functions too slow)
Apply 11.2.3.3.0 – adds flash cache for full scans
Pre-aggregation?

enkitec

Questions

Email: chris.wones@us.dunnhumby.com
Blog: www.bzzagent.com/blog/
Twitter: @chriswones

Email: kerry.osborne@enkitec.com
Blog: kerryosborne.oracle-guy.com
Twitter: @kerryoracleguy

enkitec