Designing and Tuning Large High Performance systems: Tips and Tricks

Graham Wood
Architect, Server Technologies
Designing and Tuning: Tips and Tricks

NOT

Designing for Oracle

- Changes over time
  - CPU
  - IO
  - Networks

- BUT Designing and Tuning, Large High Performance systems is NOT about tips and tricks, or fashion
- It is about good engineering practice
Continuous Innovation

Oracle 11g
Exadata Storage
Real Application Testing
Advanced Compression
Automatic Storage Management
Transparent Data Encryption
Self Managing Database
XML Database
Oracle Data Guard
Real Application Clusters
Flashback Query
Virtual Private Database
Built in Java VM
Partitioning Support
Built in Messaging
Object Relational Support
Multimedia Support
Data Warehousing Optimizations
Parallel Operations
Distributed SQL & Transaction Support
Cluster and MPP Support
Multiversion Read Consistency
Client/Server Support
Platform Portability
Commercial SQL Implementation

Oracle 10g
Oracle 9i
Oracle 11g
Oracle 8i
Oracle 8
Oracle 7
Oracle 6
Oracle 5
Oracle 2

CPU Transistor Counts 1971-2008 & Moore’s Law

Curve shows ‘Moore’s Law’: transistor count doubling every two years

Date of introduction

Transistor count

2,000,000,000
1,000,000,000
100,000,000
10,000,000
100,000
10,000
1,000
100
10

Database Performance Basics
Schemas and SQL Statements

- Good schema/data design is the foundation to good database performance.
- Good schema/data design makes downstream activities both simpler and less error prone
  - Selection of Index and Partitioning Columns
  - Gathering and maintenance of schema statistics
  - Writing correct SQL statements
  - Optimization of SQL statements
  - Avoidance of serialization points and transactional bottlenecks

Scaling
What is it?

1. Linear
2. Unlimited
3. Maximum
4. Vertical
5. Horizontal
6. Unprecedented
7. Ultimate
8. Predictable
9. Diagonal
Scalability Quiz
Label the axes?

Scaling
Some Possible Definitions

- When adding hardware resources to a system:
  - Number of transactions processed increases proportionately
  - Query times reduce proportionately
  - Query times remain constant when data volumes increase proportionately

- Hardware resources include
  - Nodes
  - CPU
  - Memory
  - Network
  - I/O
What Prevents Scaling

With respect to CPUs

- Scaling is severely impacted when the following scenarios arise
  - Hardware resources are operating beyond their practical maximum capacity for work
  - Software serialization takes place
    - Application e.g. Row lock contention
    - Database e.g. Latch contention
    - Operating System e.g. Process allocation

Scaling the Oracle Database

In theory

- Requirements when scaling the Oracle Database
  - Effective hardware capacity planning
    - A balanced design
    - Hardware components sized to absorb peak workload
  - Effective application architecture
    - All software components tested and validated
    - Safe working limits identified
Scaling the Database

In reality

- Common Application Scaling Issues
- Single Instance Scaling Issues
- Multiple Instance Scaling Issues

Common Application Issues

- Incorrect use of Sessions
  - High connect/disconnect rates to the database
  - High load on O/S for process creation/teardown
  - High load mapping/unmapping SGA
  - CPU load on establishing database state
- Incorrect use of Cursors
  - Reparsing is best avoided
  - Hard parses use lots of CPU and will serialize
  - Soft parses scale better but they are not free
Database Performance Basics
Sessions & Cursors

(Common Application Issues)

- Poor SQL
  - Poor Access Paths
  - Excessive resource usage (CPU, I/O)
- Poor connection management
  - High numbers of connections can cause problems
  - Use connection pooling/concentration in the middle-tier
  - Use of Shared Servers is often just a band-aid
Common Configuration Issues

• Non-default initialization parameters
• I/O problems
  • Poor use of available resources resulting in hot spots
    • Bandwidth limitations
    • Poor response times
  • Use ASM to eliminate unpredictable hot spots
• Space management
  • Use Automatic Undo Management
  • Use Auto Segment Storage Management

Tuning Oracle

• Changes over time
  • Hardware
  • Software
  • Functionality

• BUT Designing and Tuning, Large High Performance systems is NOT about tips and tricks, or fashion
• It is about good engineering practice
• Correct diagnosis is key to successful tuning
Oracle Tuning Methods: A History

- Prehistoric (v5)
  - Debug code
- Dark Ages (v6)
  - Counters/Ratios
  - BSTAT/ESTAT
  - SQL*Trace
- Renaissance (v7/v8)
  - Introduction of Wait Event instrumentation
  - Move from counters to timers
  - STATSPACK
- Modernity (v10)
  - DB Time Tuning – Tuning using fundamental notion of time spent in database
  - Multiple scoping levels
  - Always on, non-intrusive
  - Built into infrastructure: instrumentation, ASH, AWR, ADDM, EM

Why Do We Care About Time?

- Human time is critical to the enterprise
- Systems performance affects business goals
  - Human time + technology resource time
- “Time is money”
- Performance improvement means doing things faster

*Performance is always and only about time*
Database Time and Average Active Sessions

Database Time (DB Time)

• Total time in database calls by foreground sessions
• Includes CPU time, IO time and non-idle wait time
• DB Time <> response time
• New lingua franca for Oracle performance analysis

*Database time is total time spent by user processes either actively working or actively waiting in a database call.*
A Single Session

Single session with Database Black Box server

Database Black Box server

Browse Books
Read Reviews For One Book
Add to Cart
Checkout

TIME = time spent in database

Active Session = Session currently spending time in a database call

Database Time (DB Time) = Total time session spent in all database calls

Average Activity of the Session (% Activity) = The ratio of time active to total wall clock time

Fundamental Concepts
**Multiple Sessions**

DB Time = Sum of DB Time Over All Sessions

Avg. Active Sessions = Sum of Avg. Activity Over All Sessions

At time $t$ we have 2 active sessions

```
TIME

User 1
User 2
User 3
User n
```

= time spent in database
Visualizing DB Time

\[
\text{Avg. Active Sessions} = \frac{\text{Total Database Time}}{\text{Wall Clock (Elapsed) Time}}
\]

User 1
User 2
User 3
User n

\(t_0\)
\(t_1\)

Active Sessions over time

EM Performance Page

- Active Sessions by wait class over time
- Colored area = amount of DB time
- “Click on the big stuff”
Where to find DB Time?

- **V$SYS_TIME_MODEL, V$SESS_TIME_MODEL**
  - STAT_NAME = ‘DB time’

- **V$SYSMETRIC_HISTORY**
  - “Database Time Per Second”, “CPU Usage Per Sec”
  - 10g units = centi-secs/sec (100xAvg. Active Sessions)
  - 11g new metric “Average Active Sessions”

- **V$SQL**
  - ELAPSED_TIME and CPU_TIME
  - Wait class times:
    - APPLICATION, CONCURRENCY, CLUSTER, USER_IO

- **V$ACTIVE_SESSION_HISTORY**

---

Active Session History
Active Session History (ASH)

- All ‘Active’ sessions captured every second
  - Foregrounds and backgrounds are sampled
  - Active foregrounds contribute to DB Time

- In-memory: V$ACTIVE_SESSION_HISTORY
  - Sampling interval = 1 second

- On-disk: DBA_HIST_ACTIVE_SESS_HISTORY
  - Sampling interval = 10 second

- ASH is a system-wide record of database activity

Active Sessions and DB Time

- Active sessions
- DB time is area under curve
- ASH sample count is value of active sessions function at sample times
- \( \Delta t = 1 \text{ sec} \)
Estimating DB Time with ASH

- ASH sample counts = DB Time in seconds
  - Low sample counts are less reliable

- Enables DB Time analysis over many dimensions
  - Sqlid, session id, instance, service, module, action
  - 10gR2
    - Blocking_sid (10gR2)
    - XID
  - 11g
    - Row source
    - Execution ID
    - Operation type
      - Connect
      - Java/SQL/PLSQL
      - parse, bind, execute/fetch, close

DB Time: ASH vs Time Model
Where is DB Time used?

- ADDM
- EM Performance page and drill downs
- ASH report
- AWR and AWR compare periods reports
- SYSMETRICS and Server-generated Alerts

Techniques:

The DB Time Method
The DB Time Method: Short Course

or

just ask ADDM

The DB Time Method: Process

1. Identify performance issue
2. Scope the issue
3. Set goals
4. Data capture (NO OP)
5. Investigate DB time distribution
   • Identify the largest potential for improvement
6. Modify system to tune for largest gain
7. Evaluate against goals
   • Repeat from step 4 if goals not met

Performance tuning by removing excess DB time
Investigate DB Time Distribution

- Identify uneven distributions of DB time (skew)
  - => Largest potential improvement within scope

- System scope:
  - Resource limits – is problem outside the DB?

- Application scope:
  - Service, module, action
  - Resource contention (e.g. latches)
  - SQLID, rowsource

- Session scope:
  - Long running SQL
  - Resource contention (e.g. enqueues)
Identify Potential Solutions

- Session contention issues
  - Kill session
  - Fix application

- SQL issues
  - SQL Tuning Advisor => Indexes, SQL profile
  - Re-write SQL

- Design issues
  - Access Advisor => Indexes, physical layout

- System issues
  - Initialization parameters
  - Add resources

Modify System

- Start with the largest DB time issues first
  - Address root causes, not symptoms

- Match solution scope to problem scope
  - Don’t tweak optimizer parameters before tuning SQL

- Proceed iteratively one fix at a time
  - Concurrent fixes should be orthogonal

- Measure and validate results at each successive step

- Stop when goals are met
The DB Time Method: Advantages

- Tunes the one thing that affects users: Time
- Data capture scoping not necessary
  - ‘Always on’ data collection
  - No requirement to reproduce problem
- Works for concurrency problems such as locking
- Combines best of current methods
  - Less intrusive, more inclusive

Method Summary

- DB time is the fundamental performance metric
- The method allows DB time analysis at many scopes
  - Proper scoping of problems and solutions is critical to success
- DB time based diagnosis removes value judgments
  - Scientific method, not sorcerer’s magic
- Performance improvement means doing the same work in less DB Time
Tools:

- ADDM
- Enterprise Manager
- Reports

Tools for Applying DB Time Method

Two use-cases, one method:

1. Tuning steady-state performance
   - Improve overall workload throughput or response time
   - Best practice: use ADDM

2. Diagnosing transient performance problems
   - Confirm and investigate reported performance issues
   - Best practice: use EM real-time screens
Best Practice: Use ADDM

- Embedded expert system using the DB time method
  - Identifies root causes behind the symptoms

- Variably scoped:
  - Host to instance to SQL and even database block
  - Scoped to database for RAC (new in 11g)

- Findings prioritized by impact on DB time
  - Finding history allows flexible time scoping
  - Directives can filter findings

- Recommendations by benefit (reduction) to DB time
Best Practice: EM Real-time Interface

- Transient (sub-hour) or immediate time scope
  - Requires interactivity of UI

- ‘Click on the big stuff’
  - Data visualizations display skew directly

- Takes some expertise to separate symptoms from root causes
Conclusions

- Design of Large, High Performance Systems is engineering, not a mystical art
- Good database design leads to
  - Simpler implementation
  - Scalability
- Oracle database instrumentation
  - Simpler performance diagnosis
  - Increases probability of success
Time for Your Questions