Harnessing Power of Parallelism in an Oracle Data Warehouse

UTOUG Training Days 2016

Kasey Parker
Sr. Enterprise Architect
Kasey.Parker@centroid.com
Centroid Overview

Leader in Technology, Consulting and Managed Services since 1997

- Part of Oracle’s Top 25 Strategic Partner Program
- Focused on Consulting, Managed Services, Cloud Services and Resell
- Capabilities From Oracle Applications to Technology to Infrastructure
- Oracle Engineered Systems & Tech Partner of the Year Winner: 2014 & 2015

Specializations
Oracle Database & Core Technologies
Oracle Engineered Systems
Oracle Server & Storage Solutions
Oracle Performance Tuning
Oracle Data Warehousing
Oracle Business Intelligence
Oracle Applications
About Kasey Parker

Quick Facts

- **Reside in Plano, TX with wife and 5 kids** – recently moved from Utah
- **Oracle Architect / DBA**
  - Over 15 Years of Oracle Experience
  - Oracle Certified Exadata Specialist
  - Oracle Certified Professional - DBA
  - Performance Tuning specialist
  - Data Warehousing specialist
  - Last 5 years focused on Oracle Engineered Systems
- **Many industries**
  - Financial Services, Manufacturing, Health/Nutrition, City/County Government, Retail
- **Serve on Board of the Utah Oracle Users Group (UTOUG)**
- **Academic**
  - Brigham Young University Alumnus
  - Management Information Systems
Agenda

- Oracle Parallel Overview & Architecture
- Implementing Parallelism
- Controlling Parallelism
- Monitoring Parallelism
- Summary
Parallel Architecture

Simple Model

- Managed Services
- Cloud Services
- Consulting Services
- Licensing
## Parallel Architecture

### Simple Model

<table>
<thead>
<tr>
<th>Operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td></td>
</tr>
<tr>
<td>PXCOORDINATOR</td>
<td></td>
</tr>
<tr>
<td>PX SEND QC (RANDOM)</td>
<td>:TQ10000</td>
</tr>
<tr>
<td>PX BLOCK ITERATOR</td>
<td></td>
</tr>
<tr>
<td>TABLE ACCESS STORAGE FULL</td>
<td>INVENTORY</td>
</tr>
</tbody>
</table>

- **Query Coordinator**
- **PX Servers**
## Parallel Architecture

### Producer / Consumer Model

<table>
<thead>
<tr>
<th>Operation</th>
<th>Name</th>
<th>Line ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>PX COORDINATOR</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>PX SEND QC (ORDER)</td>
<td>:TQ10001</td>
<td>2</td>
</tr>
<tr>
<td>SORT ORDER BY</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>PX RECEIVE</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>PX SEND RANGE</td>
<td>:TQ10000</td>
<td>5</td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>TABLE ACCESS FULL</td>
<td>JOBS</td>
<td>7</td>
</tr>
<tr>
<td>PX BLOCK ITERATOR</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>TABLE ACCESS FULL</td>
<td>EMPLOYEES</td>
<td>9</td>
</tr>
</tbody>
</table>
Parallel Architecture

Data Redistribution

- Data is redistributed between producers / consumers
- Several Parallel Distribution Methods
  - E.g. Hash, Broadcast, Range
  - Method determined based on object statistics
- Adaptive Parallel Distribution
  - Hybrid Hash distribution method
  - Method can change at run-time
  - Via Adaptive Execution Plans (Statistics Collector)
- Parallel Partition-wise Joins
  - Minimize data distribution for better performance
Implementing Parallelism
Data Warehouse or
Mixed Workload Database
Implementing Parallelism

Parallelism and Partitioning

- Best to Partition Tables Used With Parallelism
  - Partition-wise Joins + Parallel = Great Performance
- Partition Large Fact and Dimension Tables
  - Facts
    - Range Partition by Load Frequency
    - Hash Sub-partition by Common Join Key
  - Dimensions
    - Hash Partition by Join Key
Implementing Parallelism

- Parallelism Enabled by Default for
  - Parallel Query
  - Parallel DDL

- Parallel DML Must be Explicitly Enabled
  - At Session Level:
    - `ALTER SESSION ENABLE PARALLEL DML;`
  - At Statement Level (New in 12c)
    - `ENABLE_PARALLEL_DML` Hint:
      - `/*+ enable_parallel_dml parallel(x) */`

```
INSERT /*+ parallel(8) enable_parallel_dml */
INTO t1 SELECT * FROM t2;
```
Implementing Parallelism

Two Methods to Manage Degree of Parallelism (DOP)

- Auto DOP
- Manual DOP

Controlled By `PARALLEL_DEGREE_POLICY`

- MANUAL (Default)
- LIMITED
- AUTO
- ADAPTIVE
Implementing Parallelism

PARALLEL_DEGREE_POLICY

- MANUAL
  - Statements parallelized by hint or giving object a DOP
    - Note: If hint specifies no DOP – AUTO DOP Calc will be used
- LIMITED
  - Auto DOP Calculation Only Used with Default DOP
  - Parallel Statement Queuing and In-memory Parallel Disabled
- AUTO
  - All Auto Parallel Features Enabled
  - Auto DOP Calculation Regardless of Hints or Object DOP
- ADAPTIVE (New in 12c)
  - Similar to AUTO, but query may “Adapt” based on stats feedback
Implementing Parallelism

Auto DOP

- Oracle decides whether to Parallelize and the DOP
- Controlled By
  - Resource Requirements (Cost)
  - `PARALLEL_MIN_TIME_THRESHOLD`
    - Default: AUTO (10 seconds)
    - Statements expected to run longer are candidates for parallel
- Should Collect IO Calibration Stats Before Using
  - `DBMSRESOURCE_MANAGER.CALIBRATE_IO`
  - Recommended for 12c, Required for 11gR2
- Also Enables
  - Parallel In-Memory
  - Parallel Statement Queuing
Implementing Parallelism

Auto DOP Decision Path (without DBRM)

1. SQL query is executed.
2. Optimizer determines the execution plan.
3. If the estimated time is greater than the threshold, Auto DOP is determined.
4. If the estimated time is less than the threshold, the statement executes serially.
5. The actual DOP is calculated as MIN(PARALLEL_DEGREE_LIMIT, Auto DOP).
6. If the DOP is greater than 1, the statement executes in parallel.

Slide image courtesy of Oracle
Implementing Parallelism

Auto DOP: Parallel In-Memory

- Traditionally, Parallel Query Uses Direct Reads
  - Data read into PGA, bypassing Buffer Cache
- Parallel In-memory Uses Buffer Cache for Parallel Query
- Challenges and Limitations in 11gR2
- Significantly Enhanced in 12c
  - Automatic Big Table Caching (ABTC)
    - Separate Part of Buffer Cache for Parallel In-Memory
    - Controlled By: DB_BIG_TABLE_CACHE_PERCENT_TARGET
      - % of SGA reserved for Parallel In-Memory
      - Uses Object Based Algorithm (opposed to standard LRU)
  - Highly Recommended If Using 12c In-Memory Option with RAC
    - Enables QC to be IMCU Instance Location Aware
Implementing Parallelism

Auto DOP: 12c Parallel In-Memory and RAC

• ABTC Splits Objects Across RAC Nodes
  • Shared Nothing Architecture
  • Each Fragment Mapped to Specific Node
    • Range of Blocks or Individual Partitions
  • All Access for Fragment Isolated to Its Node
    • PX Servers on Node Access Data and Return to QC
    • No Data Moved Via Cache Fusion

• Works Same Way for In-Memory IMCUs on RAC
Implementing Parallelism

Auto DOP: 12c Parallel In-Memory and RAC

Queries directed to PX servers on node containing the partition
Implementing Parallelism

Auto DOP: Parallel Statement Queuing

• Limits Concurrent Parallel Processes
  – Queues until enough slaves available
• Avoids overwhelming server with parallel processes
• Delivers a more consistent performance profile
• `PARALLEL_SERVER_TARGET` controls when queuing starts
  – Default: `8 * CPU_COUNT * PARALLEL_THREADS_PER_CPU`
• Statements queued in FIFO method by default
  – But can be prioritized using DBRM
• Bypassed By
  – `NO_STATEMENT_QUEUING` Hint
  – `DBRM: PARALLEL_STATEMENT_CRITICAL = BYPASS_QUEUE`
Implementing Parallelism

Auto DOP: Parallel Statement Queuing

- PX server 1-128 available to run queries before queuing kicks in
- Total PX servers available

Parallel Max Servers

Parallel Server Target

CPU Count

Managed Services
Cloud Services
Consulting Services
Licensing
Implementing Parallelism

Auto DOP: Controlling the DOP

PARALLEL_DEGREE_LEVEL

- Allows Controlling the Auto Calculated DOP
  - Use if Auto DOP is Too Aggressive or Conservative
- Set to Percentage of Calculated DOP Desired
  - Default = 100
  - 10 = Reduce to 10% of Calculated DOP
  - 200 = Double Calculated DOP
Implementing Parallelism

Auto DOP: Controlling the DOP

PARALLEL_DEGREE_LIMIT

- Limits Max DOP for AUTO DOP Calculations
- Default = CPU
  - Uses Default DOP for Database
  - PARALLEL_THREADS_PER_CPU * SUM(CPU_COUNT)
    - CPU_COUNT for All RAC Nodes
- ACTUAL DOP = MIN(Ideal DOP, PARALLEL_DEGREE_LIMIT)
- Use DBRM for More Fine-grained Control
Implementing Parallelism

Manual DOP

• You Manage Parallel Execution

• Fixed DOP or Default DOP
  • Default meant for Single-user Workloads
    • Starves System Resources in Multi-user Environments
    • PARALLEL_THREADS_PER_CPU * SUM(CPU_COUNT)
  • Set By:
    • ALTER TABLE [Table Name] PARALLEL;
    • SELECT /*+ parallel(default) */ ...

• Fixed DOP is Preferred
  • Allows You to Choose Specific DOP
  • Set By:
    • ALTER TABLE [Table Name] PARALLEL 16;
    • SELECT /*+ parallel(8) */ ...
Implementing Parallelism

Manual DOP: Fixed DOP

• Parallelism is for large, resource-intensive operations
• Designed to break a large task into smaller tasks
• Rules of Thumb
  • Be Deliberate with DOP by Setting on Table
  • Use a Power of 2 for DOP
  • Table DOP Guidelines
    • > 200MB = DOP between 4 and 16
      • Exadata: >100MB
    • > 5GB = DOP 32
      • Exadata: DOP 16-32
Implementing Parallelism

Manual DOP: Fixed DOP

- Index DOP Guidelines
  - Oracle Recommends Using Same as Table DOP
  - Exadata: Generally Don’t Use
  - Adjust DOP After Parallel Index DDL

- Results Will Vary - Test
  - Test what DOP offers best performance for the resources
  - Exadata: Generally need less DOP, but smaller tables benefit
Implementing Parallelism

Manual DOP vs Auto DOP

- Auto DOP Worth Evaluating With 12c
  - Avoid in 11g
  - Avoid with Exadata (unless In-Memory Option)
  - Use if Using In-Memory Option on RAC
  - Tweak as Necessary Using PARALLEL_DEGREE_LEVEL
  - Best Option for Mixed with Significant OLTP Workload

- Manual DOP Still Provides Most Controlled Results
  - Be Deliberate in Setting DOP on Objects
  - Possible to Still Use Parallel Statement Queuing
    - `_parallel_statement_queuing = TRUE`
Controlling Parallelism
i.e Harnessing its Power
### Controlling Parallelism

#### Key Init Parameters

- **PARALLEL_ADAPTIVE_MULTI_USER**
  - Throttles DOP Based on Concurrent Workload
  - Default: TRUE
  - Recommend: FALSE

- **PARALLEL_THREADS_PER_CPU**
  - Number of parallel threads a CPU can handle
  - Default: 2
  - Consider setting to 1 on systems with hyper-threading
    - 1 is recommended for Exadata
Controlling Parallelism

Key Init Parameters continued...

- **PARALLEL_MAX_SERVERS**
  - Max # of instance parallel server processes
  - Default: CPU_COUNT * PARALLEL_THREADS_PER_CPU * 20
    - If using SGA_TARGET or MEMORY_TARGET
  - May need to lower if high concurrency

- **PARALLEL_MIN_SERVERS**
  - Min # of instance parallel server processes (default = 0)
  - Helps control overhead of creating and destroying PX Processes
  - Recommend setting to high daily average of parallel slaves
Controlling Parallelism

DBRM: Managing Concurrent Parallel Processing

• Enables Using Parallelism in Mixed Workload Environment

• Control # of Parallel Servers
  • Statement Level
    • DOP Limit
  • Consumer Group Level
    • Max % of PARALLEL_SERVERS_TARGET

• Prioritize Parallel Statements Across Consumer Groups
  • Separate Queues for Statement Queuing
Controlling Parallelism

DBRM: Overview

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Consumer Group</td>
<td>Container for a group of sessions with same resource requirements, to which DBRM allocates resources</td>
</tr>
<tr>
<td>Resource Plan</td>
<td>Container for directives that specify how resources are allocated to consumer groups</td>
</tr>
<tr>
<td>Resource Plan Directive</td>
<td>Associates a consumer group with a resource plan and how resources are allocated</td>
</tr>
</tbody>
</table>
Controlling Parallelism

DBRM: Plan Directive Parallel Controls

- Limit DOP for Any Parallel Operation in a Consumer Group
  - \texttt{PARALLEL\_DEGREE\_LIMIT\_P1} directive attribute
- Limit # of Parallel Servers Used in a Consumer Group
  - \texttt{PARALLEL\_SERVER\_LIMIT} directive attribute
  - Set to Max % of \texttt{PARALLEL\_SERVER\_TARGET}
  - Affects When Parallel Statements are Queued
- Parallel Statement Queuing Controls
  - Allow Parallel Statements to Bypass Parallel Queue
    - \texttt{PARALLEL\_STMT\_CRITICAL} directive attribute
    - Set to \texttt{BYPASS\_QUEUE}
Controlling Parallelism

DBRM: Plan Directive Parallel Controls Continued...

- Prioritize Parallel Statement Selection For Consumer Groups
  - Separate Parallel Queues for Each Consumer Group
  - MGMT_Pn directive attribute
    - Higher Value Increases Priority of Statement Selection From Queue
    - Same Directive To Specific CPU (and IO on Exadata) %
- Limit Time a Statement Waits in Parallel Statement Queue
  - PARALLEL_QUEUE_TIMEOUT directive attribute
  - Set to Max Time in Seconds a Statement Will Wait
  - Timeout Raises a ORA-07454 Error
Controlling Parallelism

DBRM: Parallel Plan Directive Attributes Example

BEGIN

SYS.DBMS_RESOURCE_MANAGER.create_pending_area();
SYS.DBMS_RESOURCE_MANAGER.create_plan_directive(
    plan => 'DW_REPORT_CRIT_PLAN',
    group_or_subplan => 'DSS_CRITICAL_GROUP',
    mgmt_p1 => 90,
    parallel_degree_limit_p1 => 32,
    parallel_queue_timeout => 300,
    parallel_stmt_critical => bypass_queue);
SYS.DBMS_RESOURCE_MANAGER.create_plan_directive(
    plan => 'DW_REPORT_CRIT_PLAN',
    group_or_subplan => 'OTHER_GROUPS',
    mgmt_p1 => 10,
    parallel_degree_limit_p1 => 12,
    parallel_queue_timeout => 300,
    parallel_server_limit => 50);
SYS.DBMS_RESOURCE_MANAGER.submit_pending_area();

END;
Controlling Parallelism

DBRM: Parallel Plan Directive Attributes – OEM

```
Specify a limit on the degree of parallelism for any operation issued by this consumer group, a limit on the total number of parallel servers that can be used by all sessions is maximum time a parallel statement can be queued.
```

<table>
<thead>
<tr>
<th>Group</th>
<th>Max Degree of Parallelism</th>
<th>Max Percentage of Parallel Servers Target</th>
<th>Parallel Queue Timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>LONG_SQL_GROUP</td>
<td>64</td>
<td>25</td>
<td>UNLIMITED</td>
</tr>
<tr>
<td>MEDIUM_SQL_GROUP</td>
<td>64</td>
<td>50</td>
<td>UNLIMITED</td>
</tr>
<tr>
<td>OTHER_GROUPS</td>
<td>16</td>
<td>25</td>
<td>UNLIMITED</td>
</tr>
<tr>
<td>SHORT_SQL_GROUP</td>
<td>128</td>
<td>UNLIMITED</td>
<td>UNLIMITED</td>
</tr>
<tr>
<td>SYS_GROUP</td>
<td>UNLIMITED</td>
<td>UNLIMITED</td>
<td>UNLIMITED</td>
</tr>
</tbody>
</table>
Monitoring Parallelism
Parallelism “Gotchas”

• Parallel Downgrade or Serialization
• Parallel DML – Forgetting to Enable
• Partially Parallelized SQL
• Parallel Index Creation
  – Use but don’t forget to remove or drop DOP
• Parallel Bulk Load Operations
  – Use with Direct Path Loading
  – Find out how to use with 3rd Party ETL tool to:
    • Enable Parallel DML
    • Direct Path: i.e. CTAS, Insert /*+ APPEND */ , Bulk Load API, etc
Monitoring Parallelism

• SQL Monitor Report
  – Accessed via EM or DBMS_SQL_MONITOR

• GV$PX PROCESS
  – One Record per Parallel Worker

• GV$SQL_MONITOR
  – Also Shows Queued Parallel Statements

• Parallel Health-Check and Diagnostics Report
  – pxhcdr.sql script (MOS Note: 1460440.1)
# Monitoring Parallelism

## SQL Monitor: Parallel Executions

<table>
<thead>
<tr>
<th>Status</th>
<th>Duration</th>
<th>Type</th>
<th>ID</th>
<th>SQL Plan Hash</th>
<th>User</th>
<th>Parallel</th>
<th>Database Time</th>
<th>IO Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Green Circle]</td>
<td>26.0s</td>
<td>![Red Circle]</td>
<td>7np8cpxhp9rzd</td>
<td>57783867</td>
<td>SW</td>
<td>![Red Circle]</td>
<td>25.1s</td>
<td>3,523</td>
</tr>
<tr>
<td>![Green Circle]</td>
<td>1.0m</td>
<td>![Red Circle]</td>
<td>2hxdwjm8p86pw1</td>
<td>57783867</td>
<td>SW</td>
<td>![Red Circle]</td>
<td>1.6m</td>
<td>2,953</td>
</tr>
<tr>
<td>![Green Circle]</td>
<td>1.1m</td>
<td>![Red Circle]</td>
<td>9w1u6mrc2prb7</td>
<td>261098460</td>
<td>SW</td>
<td>![Red Circle]</td>
<td>1.4m</td>
<td>2,953</td>
</tr>
<tr>
<td>![Green Circle]</td>
<td>1.3m</td>
<td>![Red Circle]</td>
<td>7shhs73v5k8zf</td>
<td>57783867</td>
<td>SW</td>
<td>![Red Circle]</td>
<td>2.4m</td>
<td>6,323</td>
</tr>
<tr>
<td>![Green Circle]</td>
<td>1.2m</td>
<td>![Red Circle]</td>
<td>8a6sk6jpdwg2</td>
<td>57783867</td>
<td>SW</td>
<td>![Red Circle]</td>
<td>2.4m</td>
<td>6,323</td>
</tr>
<tr>
<td>![Green Circle]</td>
<td>1.2m</td>
<td>![Red Circle]</td>
<td>b42g2czhyvzb</td>
<td>57783867</td>
<td>SW</td>
<td>![Red Circle]</td>
<td>2.3m</td>
<td>6,323</td>
</tr>
<tr>
<td>![Green Circle]</td>
<td>1.5m</td>
<td>![Red Circle]</td>
<td>b2hwjanmpcats</td>
<td>1822208142</td>
<td>SW</td>
<td>![Red Circle]</td>
<td>2.8m</td>
<td>10K</td>
</tr>
<tr>
<td>![Green Circle]</td>
<td>1.2m</td>
<td>![Red Circle]</td>
<td>catz3s8zt170</td>
<td>57783867</td>
<td>SW</td>
<td>![Red Circle]</td>
<td>2.3m</td>
<td>6,323</td>
</tr>
</tbody>
</table>
## Monitoring Parallelism

### SQL Monitor: Queued Parallel Executions

<table>
<thead>
<tr>
<th>Status</th>
<th>Duration</th>
<th>Type</th>
<th>ID</th>
<th>SQL Plan Hash</th>
<th>User</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚾️</td>
<td>26.0s</td>
<td></td>
<td>7np8cpxhp9rzd</td>
<td>57783867</td>
<td>SW</td>
<td></td>
</tr>
<tr>
<td>🌞</td>
<td>1.0m</td>
<td></td>
<td>2hxdwjm8b6pw1</td>
<td>57783867</td>
<td>SW</td>
<td></td>
</tr>
<tr>
<td>🌞</td>
<td>1.1m</td>
<td></td>
<td>9w1u6mrc2prb7</td>
<td>2610198460</td>
<td>SW</td>
<td>4</td>
</tr>
<tr>
<td>✅</td>
<td>1.3m</td>
<td></td>
<td>7shhs73v5k8zf</td>
<td>57783867</td>
<td>SW</td>
<td>4</td>
</tr>
<tr>
<td>✅</td>
<td>1.2m</td>
<td></td>
<td>8a6tsk6jpdwg2</td>
<td>57783867</td>
<td>SW</td>
<td>4</td>
</tr>
<tr>
<td>✅</td>
<td>1.2m</td>
<td></td>
<td>b42g2czt1vynzb</td>
<td>57783867</td>
<td>SW</td>
<td>4</td>
</tr>
<tr>
<td>✖️</td>
<td>1.5m</td>
<td></td>
<td>b2hwjanmpcats</td>
<td>1822280142</td>
<td>SW</td>
<td>4</td>
</tr>
<tr>
<td>✅</td>
<td>1.2m</td>
<td></td>
<td>catz3s8z7t170</td>
<td>57783867</td>
<td>SW</td>
<td>4</td>
</tr>
</tbody>
</table>
Monitoring Parallelism

SQL Monitor: Parallel Execution Example

![SQL Monitor screenshot](image)

- Managed Services
- Cloud Services
- Consulting Services
- Licensing
Monitoring Parallelism

SQL Monitor: Parallel Execution Example

<table>
<thead>
<tr>
<th>Operation</th>
<th>Name</th>
<th>Line</th>
<th>Estimated Rows</th>
<th>Cost</th>
<th>Timeline(219s)</th>
<th>Execs...</th>
<th>Actual Rows</th>
<th>Memory (MB)</th>
<th>Temp (Max)</th>
<th>Olts</th>
<th>IO Requests</th>
<th>IO Bytes</th>
<th>Activity %</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td></td>
<td>0</td>
<td>1</td>
<td>16K</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>.48</td>
</tr>
<tr>
<td>SORT AGGREGATE</td>
<td></td>
<td>1</td>
<td>1</td>
<td>16K</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>.24</td>
</tr>
<tr>
<td>PK COORDINATOR</td>
<td></td>
<td>2</td>
<td></td>
<td>190K</td>
<td></td>
<td>9</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>.24</td>
</tr>
<tr>
<td>PK SEND QC (RANDOM)</td>
<td>TQ100002</td>
<td>3</td>
<td>1</td>
<td>430K</td>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>.24</td>
</tr>
<tr>
<td>SORT AGGREGATE</td>
<td></td>
<td>4</td>
<td>1</td>
<td>430K</td>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>.24</td>
</tr>
<tr>
<td>MERGE JOIN</td>
<td></td>
<td>5</td>
<td>10M</td>
<td>430K</td>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>.24</td>
</tr>
<tr>
<td>SORT JOIN</td>
<td></td>
<td>6</td>
<td>3,328K</td>
<td>16K</td>
<td></td>
<td>4</td>
<td>1,524K</td>
<td>14MB</td>
<td>20MB</td>
<td>424</td>
<td>35MB</td>
<td></td>
<td>.24</td>
</tr>
<tr>
<td>PK RECEIVE</td>
<td></td>
<td>7</td>
<td>3,328K</td>
<td>3,179</td>
<td></td>
<td>4</td>
<td>1,524K</td>
<td>14MB</td>
<td>20MB</td>
<td>424</td>
<td>35MB</td>
<td></td>
<td>.24</td>
</tr>
<tr>
<td>PK SEND QC (RANDOM)</td>
<td>TQ100000</td>
<td>8</td>
<td>3,328K</td>
<td>16K</td>
<td></td>
<td>4</td>
<td>1,524K</td>
<td>14MB</td>
<td>20MB</td>
<td>424</td>
<td>35MB</td>
<td></td>
<td>.24</td>
</tr>
<tr>
<td>STATISTICS COLLECTOR</td>
<td></td>
<td>9</td>
<td></td>
<td>16K</td>
<td></td>
<td>4</td>
<td>1,524K</td>
<td>14MB</td>
<td>20MB</td>
<td>424</td>
<td>35MB</td>
<td></td>
<td>.24</td>
</tr>
<tr>
<td>PK BLOCK ITERATOR</td>
<td></td>
<td>10</td>
<td>3,328K</td>
<td>3,179</td>
<td></td>
<td>4</td>
<td>1,524K</td>
<td>14MB</td>
<td>20MB</td>
<td>424</td>
<td>35MB</td>
<td></td>
<td>.24</td>
</tr>
<tr>
<td>TABLE ACCESS FULL</td>
<td>CUSTOMERS</td>
<td>11</td>
<td>3,328K</td>
<td>3,179</td>
<td></td>
<td>53</td>
<td>1,524K</td>
<td>14MB</td>
<td>20MB</td>
<td>424</td>
<td>35MB</td>
<td></td>
<td>.24</td>
</tr>
<tr>
<td>SORT JOIN</td>
<td></td>
<td>12</td>
<td>66M</td>
<td>414K</td>
<td></td>
<td>1,524K</td>
<td>5,584K</td>
<td>14MB</td>
<td>1GB</td>
<td>51</td>
<td>392K</td>
<td>28MB</td>
<td>.12</td>
</tr>
<tr>
<td>PK RECEIVE</td>
<td></td>
<td>13</td>
<td>66M</td>
<td>63K</td>
<td></td>
<td>4</td>
<td>102M</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>.6</td>
</tr>
<tr>
<td>PK SEND HYBRID HASH</td>
<td>TQ100001</td>
<td>14</td>
<td>66M</td>
<td>63K</td>
<td></td>
<td>4</td>
<td>102M</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>.6</td>
</tr>
<tr>
<td>PK BLOCK ITERATOR</td>
<td></td>
<td>15</td>
<td>66M</td>
<td>63K</td>
<td></td>
<td>4</td>
<td>102M</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>.6</td>
</tr>
<tr>
<td>TABLE ACCESS FULL</td>
<td>SALES</td>
<td>16</td>
<td>66M</td>
<td>63K</td>
<td></td>
<td>69</td>
<td>102M</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>85</td>
</tr>
</tbody>
</table>

Buffer Gets: 876K
Buffer Gets: 499K
Buffer Gets: 34GB
Monitoring Parallelism

SQL Monitor: Parallel Execution Plan Note
Monitoring Parallelism

SQL Monitor: Parallel DOP and Downgrade

SQL Text: SELECT t.time_id, TO_CHAR(SUM(amount_sold), '9,999,999')

Execution Plan: 2

Execution Started: Fri Apr 3, 2015 10:32:28 AM

Last Refresh Time: Fri Apr 3, 2015 10:32:28 AM

Execution ID: 1672

User: SW

Fetch Calls: 1

Degree of Parallelism: 2

Degree of Parallelism Downgrade: 50%

Parallel Execution Server Requested: 8

Parallel Execution Server Allocated: 4
Monitoring Parallelism

SQL Monitor: Parallel DOP and Downgrade

Downgrade Reasons:
• 350 = due to adaptive DOP
• 351 = due to resource manager max DOP
• 352 = due to insufficient number of processes
• 353 = because slaves failed to join
# Monitoring Parallelism

## SQL Monitor: PX Server Work Distribution

<table>
<thead>
<tr>
<th>Parallel Server</th>
<th>Database Time</th>
<th>Activity %</th>
<th>IO Requests</th>
<th>IO Bytes</th>
<th>Buffer Gets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Coordinator</td>
<td>0.9s</td>
<td>24</td>
<td></td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>Parallel Set 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Server 1 (p000)</td>
<td>2.9m</td>
<td>21</td>
<td>96K</td>
<td>7GB</td>
<td>7,728</td>
</tr>
<tr>
<td>Parallel Server 2 (p001)</td>
<td>3.0m</td>
<td>22</td>
<td>96K</td>
<td>7GB</td>
<td>7,713</td>
</tr>
<tr>
<td>Parallel Server 3 (p002)</td>
<td>3.0m</td>
<td>22</td>
<td>96K</td>
<td>7GB</td>
<td>7,743</td>
</tr>
<tr>
<td>Parallel Server 4 (p003)</td>
<td>3.0m</td>
<td>22</td>
<td>96K</td>
<td>7GB</td>
<td>7,741</td>
</tr>
<tr>
<td>Parallel Set 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Server 1 (p004)</td>
<td>29.9s</td>
<td>2.9</td>
<td>27K</td>
<td>2GB</td>
<td>213K</td>
</tr>
<tr>
<td>Parallel Server 2 (p005)</td>
<td>32.2s</td>
<td>3.5</td>
<td>26K</td>
<td>2GB</td>
<td>211K</td>
</tr>
<tr>
<td>Parallel Server 3 (p006)</td>
<td>32.4s</td>
<td>3.86</td>
<td>26K</td>
<td>2GB</td>
<td>209K</td>
</tr>
<tr>
<td>Parallel Server 4 (p007)</td>
<td>30.9s</td>
<td>3.02</td>
<td>27K</td>
<td>2GB</td>
<td>212K</td>
</tr>
</tbody>
</table>
Monitoring Parallelism

SQL Monitor: Not Enabling Parallel DML
Monitoring Parallelism

SQL Monitor: Partially Parallelized SQL
Summary

• Parallelism Overview
  – Splits single DB Operation into Multiple Concurrent Processes
  – Better utilizes available system resources
  – Useful for large, resource-intensive operations
  – Suited for Data Warehouse and Mixed Workload DBs

• Works Best with Partitioning

• Auto DOP vs Manual DOP

• Control with Init Parameters, DBRM and Queuing

• Monitor with SQL Monitor
Questions?
References

- Parallel Execution Fundamentals White Paper
  - [Link](http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-parallel-execution-fundamentals-133639.pdf)
- Parallel Execution and Workload Management White Paper
  - [Link](http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-bidw-parallel-execution-130766.pdf)
- Monitoring Parallel Execution Using RSM
  - [Link](https://blogs.oracle.com/datawarehousing/entry/monitoring_parallel_execution_using_real)
- MOS Notes:
  - Using Parallel Execution (Doc ID 203238.1)
  - Setup, Monitor, And Tune Parallelism In The DB (Doc ID 1549214.1)
  - Parallel Execution Health-Check and Diagnostics (Doc ID 1460440.1)
  - Do You Need IO Calibrate Stats for Auto DOP in 12C? (Doc 1963310.1)