

The Art of Tuning & Managing the Parallel Query Option - Is it a Free Lunch ??

Paper 122

Presented at
Oracle Open World
September 19-26, 1997

by
Noorali Sonawalla
Sunrise Systems, Inc.
P.O.Box 4647
Metuchen, NJ 08840
Tel : (732) 603-2200
Fax : (732) 603-2208
email : noorali@sunrisesys.com
www.sunrisesys.com

Objective

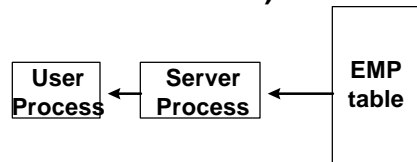
- What is Parallel Query Option (PQO)
- What can be Parallelized
- Internal Workings of PQO
- Impact of Striping on PQO
- Other Parallel Operations (v7 & v8)
- Performance Monitoring and Tuning Issues
- Summary

- **Acknowledgment**
 - ↳ Some of the Figures and Examples in this presentation are taken from

“Advanced Oracle Tuning and Administration”, Oracle Press by : Eyal Aronoff, Kevin Loney, Noorali Sonawalla

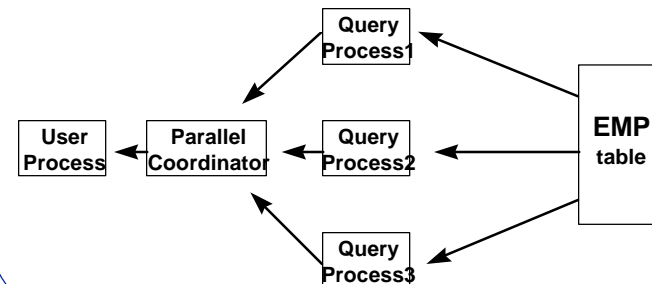
What is The Parallel Query Option (PQO) - Default

- Oracle Parallel Server (OPS) supports multiple instances on different nodes accessing one database spread over the nodes - Higher **Scaleup** for OLTP - (separate option)
- PQO splits a single task (e.g. Full Table Scan) into smaller units and process them in parallel - **Speedup**
- Traditional Serialized 2-process Scan : only one process works - to serially scan the entire table (other idle resources are not used)



How Parallel Query Works

- Parallelized Processing
- Multiple processes scan different parts of a table concurrently
- Server Process becomes the query coordinator and multiple query slave processes are started for the scan
- For MTS, the server process that processes the EXECUTE call becomes the query coordinator for the statement



Pros and Cons of PQO

- Default option with v7.1 and above
- Uses idle CPU / Memory / IO resources
- Each process requires cpu, memory and IO resources
- Overall resource consumption does not necessarily decrease - effective processing time is condensed
- PQO is more effective with multiple CPUs and disks - It can work with a single cpu and disk
- Resource-starved system will not benefit from PQO
- Overall resource consumption needs have to be monitored carefully

What are Partitioned Tables and Indexes (Oracle8)

```
create table sales      (cust_no   number
                        cust_name char(30),
                        amount    number,
                        month_no  number)
partitioned by range (month_no)
(partition sales_p1 values less than (1) tablespace t1,
 partition sales_p2 values less than (2) tablespace t2,
 . . . .
 partition sales_p12 values less than (12) tablespace
 t12);
```

- ↳ Partitions have identical logical attributes
- ↳ Partitions may have different physical and storage attributes (built in striping)
- ↳ `select * from sales partition (sales_p10);`
- ↳ Backup, recovery, managing by partition
- ↳ Indexes may also be partitioned
- ↳ Indexes on partitioned tables may be global or local
- ↳ Compare with v7 **Partition Views**

What can be Parallelized

→ For Oracle v7 :

- ↳ **SELECT** (for sorts and full table scans, including subqueries for update and delete)
- ↳ **CREATE TABLE AS SELECT** (query portion parallelized in v7.1, insert parallelized in v7.2)
- ↳ **CREATE INDEX**
- ↳ **SQL*Loader - Direct Path**
- ↳ **RECOVER**
- ↳ **INDEX SCANS ARE NOT PARALLELIZED**

→ For Oracle v8

- ↳ update / delete (only for partitioned tables; this parallelism is not possible within a partition or on a non-partitioned table)
- ↳ enable constraint (table scan parallelized)
- ↳ rebuild index / rebuild index partition
- ↳ move / split partition
- ↳ **index range scans only by partitions**
- ↳ and more

Noorali Sonawalla
noorali@sunrisesys.com

7

©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

How Oracle Parallelizes Operations - 3 Forms

→ By **contiguous block ranges** for scan operations (dynamic partitioning)

- ↳ Dynamic allocation of ROWID ranges
- ↳ ROWID range cannot span partitions
- ↳ Used for table scans and operations on a single partition or non-partitioned table
- ↳ e.g. table scans, move/split partition, rebuild index on partition plus create index/create table ... as select (for non-partitioned index & tables)

→ By **partitions** for operations on partitioned tables and indexes

- ↳ Parallel Server Processes assigned to partitions
- ↳ Mainly used for multi-partition operations e.g. create index, create table .. as select, **update**, **delete**, insert .. select, alter index ... rebuild, index range scan on partitioned index

→ By **parallel server processes** for inserts into non-partitioned tables

- ↳ Since new rows do not have ROWIDs

→ **Sorts** use a pseudo-dynamic form

Noorali Sonawalla
noorali@sunrisesys.com

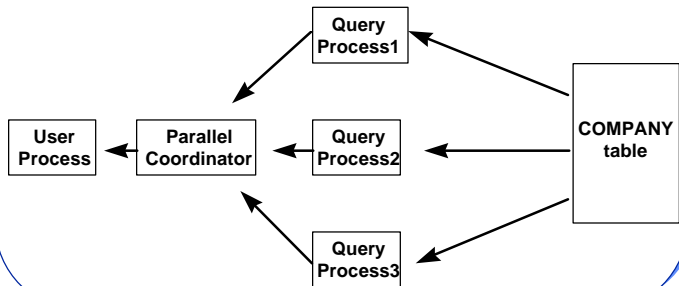
8

©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

How Does PQO Work - Full Table Scan

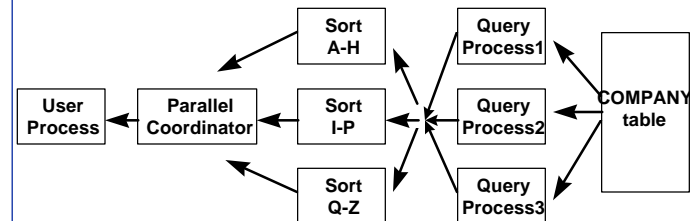
→ **select * from COMPANY;**

- ↳ Invoke Degree of Parallelism (DOP) = 3
 - ↳ **alter table COMPANY parallel (degree 3);** OR
 - ↳ **select /*+ PARALLEL(company,3) */ * from COMPANY;**
- DOP = no. of parallel server processes associated with a single operation
 - DOP specified at statement level, table/ index level or by default based on the # of disks / CPUs
 - Actual parallelism depends on free resources



How Does PQO Work - Full Table Scan with Sort

→ **select * from COMPANY order by NAME;** (DOP = 3 and not 6)



INTRA- INTER- INTRA-
Operation Operation Operation
Parallelism Parallelism Parallelism

Sort Key Value ROWID
Partitioning Partitioning

How are Query Server Processes Assigned ?

- Common Pool of Background Query Server Processes for an Instance
- Query Servers are assigned from the pool as needed - more are started up as needed, subject to 4 Key init.ora Parameters :
- **PARALLEL_MIN_SERVERS**
(minimum active servers)
- **PARALLEL_MAX_SERVERS**
(maximum active servers)
- **PARALLEL_SERVER_IDLE_TIME**
(idle servers terminated - subject to minimum limits)
- **PARALLEL_MIN_PERCENT**
(new parameter with v7.3 - prevents unexpected serialization of queries - gives an error if it cannot parallelize enough)

Noorali Sonawalla
noorali@sunrisesys.com

11

©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

Impact of Data Striping on Parallel Query Processing

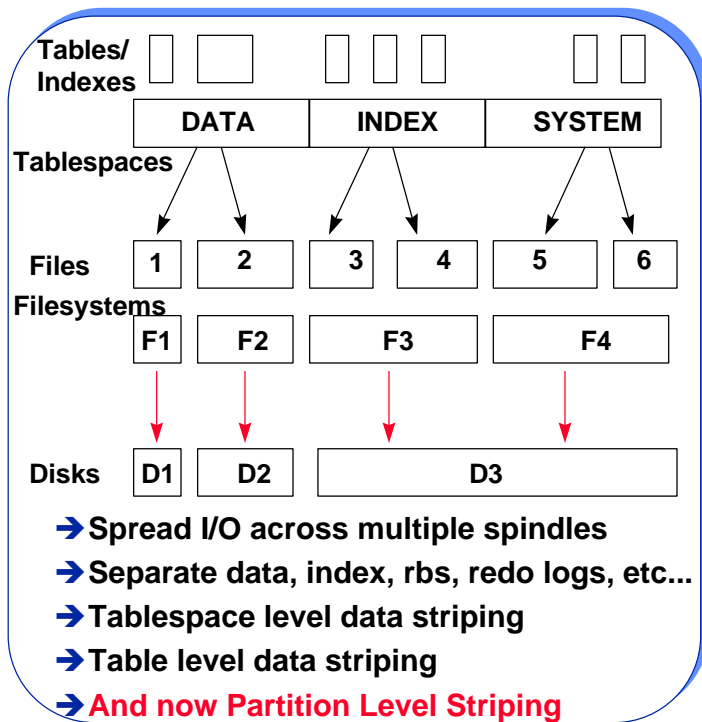
- What is Data Striping - spreading data across “multiple disks”
- Provides better concurrency - better Scaleup
- May or may not provide better batch throughput - better Speedup
- Types of Striping
 - ↳ Block-based striping using OS or hardware level RAIDs
 - ↳ Manual striping by allocating data files to different disks
 - ↳ Partitioned based striping in v8
- OS level striping is good for concurrency / scaleup - more automated - watch for any impact on batch or parallel processing
- Concurrency vs Availability - Breadth of striping may impact availability

Noorali Sonawalla
noorali@sunrisesys.com

12

©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

I/O Management - File Layout



Noorali Sonawalla
noorali@sunrisesys.com

13

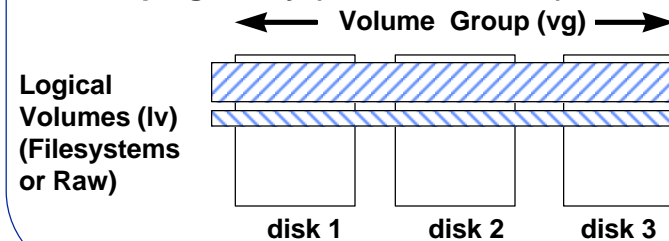
©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

I/O Management - Logical Volume Manager

→ What is Logical Volume Manager (LVM)?

- ↳ Breaks the 1-filesystem to 1-partition limitation on unix
- ↳ Multiple partitions and disks can be bunched together into a Volume Group
- ↳ Multiple Logical Volumes can be created in one Volume Group
- ↳ Each Volume Group can be used to create a filesystem or used raw

→ LVM provides operating system level striping ability (one more level !)

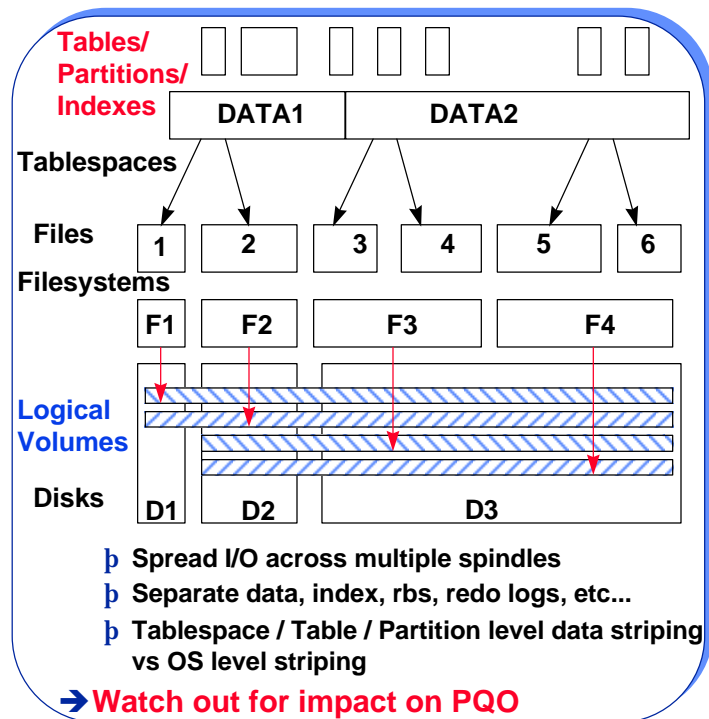


Noorali Sonawalla
noorali@sunrisesys.com

14

©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

I/O Management - File Layout



Noorali Sonawalla
noorali@sunrisesys.com

15

©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

Parallel Table Create

```
create table COMPANY2 parallel(degree 4)
as select /*+ PARALLEL(company,4) */ *
from COMPANY;
```

- SELECT parallelized in v7.1
- INSERT parallelized in v7.2
- Each Query Server Process allocates and populates a separate temporary extent (minextents of size initial!)
- All extents are combined by the query coordinator (multi-extent tables / indexes)
- Possible to create “pockets” of free space
 - ↳ external fragmentation - multiple smaller extents released
 - ↳ internal fragmentation - multiple empty unused pockets
- Writing redo logs is serialized
- Use Unrecoverable (**NoLogging in v8**) clause to prevent redo logging - esp. if DOP is high

Noorali Sonawalla
noorali@sunrisesys.com

16

©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

Parallel Index Create

```
create index COMPANY_IDX on COMPANY (City, State)
parallel(degree 4);
```

- Parallelized in v7.1
- One set of query processes get index column values and ROWIDs from table
- Second set of query processes sort the rows and create sorted subindexes
- Query coordinator builds B*-tree index from sorted lists
- Each Query Server Process allocates and populates a separate extent
- Use Unrecoverable (**NoLogging in v8**) clause to prevent redo logging - especially if DOP is high
- Cannot create an index in parallel when adding/enabling UNIQUE or PRIMARY key constraint (v7)

Noorali Sonawalla
noorali@sunrisesys.com

17

©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

Parallel Data Loading

```
sqlload userid = ME/ PASS
control = P1.CTL direct=TRUE parallel=TRUE
sqlload userid = ME/ PASS
control = P2.CTL direct=TRUE parallel=TRUE
sqlload userid = ME/ PASS
control = P3.CTL direct=TRUE parallel=TRUE
```

- Separate data, control, log, bad, discard files
- Only APPEND option allowed (no REPLACE, TRUNCATE, INSERT)
- Indexes not maintained automatically - they must be dropped before the load and later recreated
- Uses temp segments which are later merged with the table - if the load fails, no data is committed
- Check the constraints after the load
- Different mechanism than Parallel Query

Noorali Sonawalla
noorali@sunrisesys.com

18

©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

Parallel Recovery

- Supported from v7.1
- `RECOVERY_PARALLELISM = n` specifies Degree of Parallelism (init.ora)
- This can be overridden by `PARALLEL` clause of the `RECOVER` command
- A single SQLDBA or Server Manager session reads the archive logs during media recovery and passes the information to multiple recovery processes, which apply the changes to datafiles concurrently

Direct Path Options (v7) SQL*Loader and Export

- v7 Direct Path Load
 - ↳ Bypasses SQL Layer
 - ↳ Row level vs Block level access / insert
 - ↳ Preformatted Blocks “added” above HWM
 - ↳ Parallelism Supported
- v7 Direct Path Export
 - ↳ Bypasses SQL Layer for Exports
 - ↳ No support for parallelism

Direct-Load INSERT (v8)

→ v8 Direct-Load INSERT

- ↳ Bypasses SQL layer & appends data
- ↳ Supports serial and parallel modes
- ↳ Supports partitioned / non-partitioned tables
- ↳ “No-logging” mode & no undo entries
- ↳ **Serial (non-partitioned) mode** adds data above existing HWM. HWM updated on commit.
- ↳ **Parallel mode (non-partitioned) mode** allocates new temporary segments for adding data
- ↳ **Parallel (partitioned tables) mode** adds data above HWM of each partition
- ↳ Serial Direct-Load requires APPEND hint
insert /*+ APPEND */ into empnew
select * from emp; commit;
- ↳ Parallel Direct-Load requires PARALLEL table attribute or hint - APPEND hint is optional
- ↳ Local indexes on partitioned tables are rebuilt (global indexes are not supported)
- ↳ Regular indexed tables are not supported
- ↳ Exclusive lock on underlying table

Noorali Sonawalla
noorali@sunrisesys.com

21

©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

Parallel DML Update/Delete (mainly O8)

- Update and Delete operations are parallelized only by partitions
- Not applicable within a partition or on a non-partitioned table
- Each partition can be updated or deleted by one parallel server process
- Each parallel server process may process one or more partitions
- Decision to parallelize update / delete is independent of the query portion
- Use “alter session enable parallel dml”
- Parallel DML uses different locking, transaction and recovery mechanism
- More rollback segments needed
- Set CLEANUP_ROLLBACK_ENTRIES high to speedup rollback

Noorali Sonawalla
noorali@sunrisesys.com

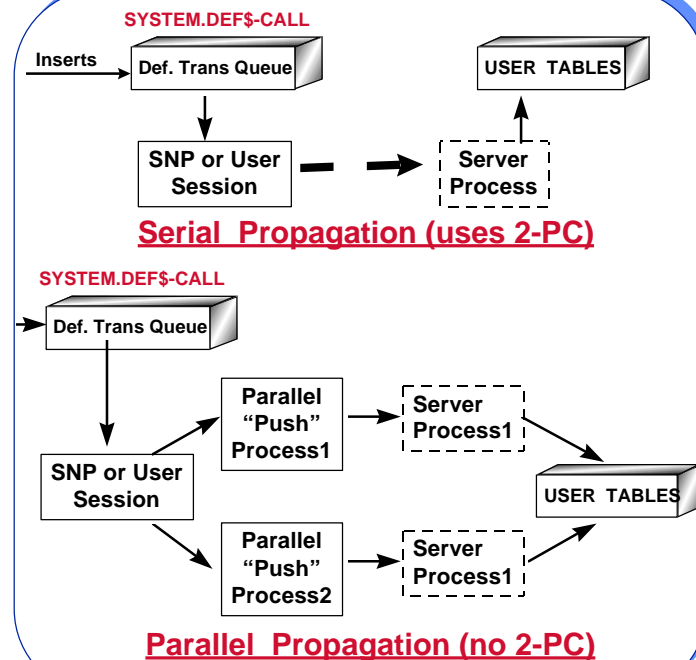
22

©Sunrise Systems, Inc., 1996 - 1997
(732) 603-2200

Parallel Propagation in Symmetric Replication (v8)

- In Master-to-Master Replication, transactions are queued in a common set of tables called Deferred Queue
- At periodic intervals, these transactions are pushed to other master sites
- In v7, the “push” mechanism between 2 sites is serialized
- in v8, this “push” mechanism can be parallelized
 - p Uses the same basic mechanism of a Parallel Coordinator and multiple Parallel Server Processes which “push” the deferred queues in parallel - maintaining transaction consistency

Parallel Propagation in Symmetric Replication (v8)



How to Monitor PQO

- In unix, Query Processes are numbered as p000, p001, etc..
- At OS level, use “ps -ef | grep ora_p0” to list active Query Processes and monitor overall IO and Memory closely
- Internal Dynamic Performance Tables
 - ↳ **V\$PQ_SYSSTAT** (system level statistics)
 - ↳ **V\$PQ_SESSTAT** (session level statistics)
 - ↳ **V\$PQ_SLAVE** (statistics for each active parallel query server process)
 - ↳ **V\$PQ_TQSTAT** (new in v7.3 - statistics for all parallel queries and parallel query operations for the session)

Managing PQO

- **Overriding the Default Degree of Parallelism (DOP)**
 - ↳ In 7.1 and 7.2, default DOP could be set at Instance, Table or Query levels. In 7.3, Instance level option is not available (**PARALLEL_DEFAULT_SCANSIZE** and **PARALLEL_DEFAULT_MAX_SCANS** are obsolete)
 - ↳ In v7.3, Oracle checks the number of CPUs and Disks to determine default DOP
 - ↳ Parallelism for a table is considered by the optimizer only if it is set at the table level or specified in a query hint
 - ↳ Once a table is a candidate for parallelism, the DOP is taken from query hint or table definition or from default value - in this order
- **Number of Query Servers allocated depend on available resources**
- **DOP = 1 implies serial processing - with no query servers (except for replication !)**

Managing PQO

→ Rewriting SQL

- ↳ Oracle can parallelize joins more effectively than subqueries
- ↳ Convert subqueries, especially correlated subqueries, into joins - at times, the optimizer does that
- ↳ Use Explain Plan to determine which operations are being parallelized
- Oracle cannot return the final results to a user in parallel - it can insert into a table in parallel. Parallelizing online browsing of a table scan will hold up system resources
- Limit Parallelism to tables and queries that really need it and can be monitored
- Keep a close watch on overall Memory, CPU and IO resources
- Understand and use Explain Plan

Summary

- What is Parallel Query Option (PQO)
- What can be Parallelized
- Internal Workings of PQO
- Impact of Striping on PQO
- Other Parallel Operations (v7 & v8)
- Performance Monitoring and Tuning Issues
- Summary