Tuning Database Reorganizations for Maximum Speed

Gil Asherie & Heather Compher

Gil.Asherie@quest.com; Heather.Compher@quest.com

Space Management & Reorganization Products

Quest Software

Oracle Open World 2000

Agenda

- Why Reorganize?
- Available Reorganization Methods
- Tuning Tips
- Case Studies

Why Reorganize?

- Performance, availability and manageability
  - Condense data in tables
  - Tidy up indexes
  - Recover wasted disk space
  - Relocate objects
  - Restructure objects

Available Reorg Methods

- Oracle Export-Import
- SQL to unload, SQL*Loader to reload
- SQL & PL/SQL scripts
- Third-party products

Data Movement

- DB -> File System -> DB
- DB -> Temp location in database -> DB
- DB -> New location in DB

Reorg of Table EMP

- Create temp table EMP_ (with optimal storage)
- **Copy data from original table EMP to EMP_**
- Drop EMP
- Rename EMP_ to EMP
- **Recreate indexes**
- **Reapply constraints**
- Recreate triggers and other dependencies on EMP

Three Types of Insert

- Singleton insert
  - insert into… values (:a1, :a2, :a3)
- Internal array insert
  - insert into… select * from…
  - Import
  - Only works with small rows (significantly less than a block)
- Direct path block insert
  - SQL*Loader Direct = Yes
  - create table… as select…

“Create” & “Copy” steps

- **Create...; Insert...**

```
create table EMP_ (col1, col2, …)
    tablespace USER_DATA
    storage (initial 10M Next 10M …);
 insert into EMP_ (col1, col2, …)
    select col1, col2, …
    from EMP;
```

- **Create... as Select...**

```
 create table EMP_
    tablespace USER_DATA
    storage (initial 10M next 10M …)
    as
    select col1, col2, …
    from EMP;
```

The NOLOGGING Option

- Turns off writing to redo logs for table/index builds
- 30-50% performance boost
- Invalidates Oracle Standby Database
- After reorg with NOLOGGING Option, a hot backup of affected tablespace is recommended
- For tables, only works with direct path load !!!!!!!!!!!!
- For index, only works DURING the Create Index

Insert and NOLOGGING

- Singleton insert/update – **always writes to log**
  - – insert into… values (:a1, :a2, :a3)
- Internal array insert – **always writes to log**
  - – insert into… select * from…
  - – Import direct = yes **(yes, always writes to log!)**
- Direct path block insert – **does not write to log**
  - – SQL*Loader Direct = Yes
  - – create table… as select…

The NOLOGGING Option

- Syntax for Tables:

```
create table EMP_
  nologging
  tablespace USER_DATA
  storage (initial 10M next 10M …)as
  select col1, col2, …
  from EMP;
```

- Syntax for indexes:

```
create index EMP$NAME on EMP (Name)
  nologging
  tablespace USER_INDEX
  storage (initial 10M next 10M …);
```

Parallel Query Option (PQO)

- Syntax for building tables and indexes
- How PQO Works
- Choosing a Degree of Parallelism
- Tuning the Query Server Pool
- Extent Size Considerations

PQO Syntax for Table Builds

- Alter session enable parallel DML;
- Parallel clause in Create Table Doesn’t Help!

```
create table EMP_
     tablespace USER_DATA
     parallel (degree 4 instances default)
     storage (initial 10M, next 10M, …)
     as
     select
     col1, col2, …
     from EMP;
```

- **This parallelizes future access to the data, but not the table build itself!**

PQO Syntax for Table Builds

- Instead, use **parallel hint in subquery**

```
create table EMP_
   unrecoverable
   tablespace USER_DATA
   storage (initial 10M, next 10M, …)
   as
   select /*+ Parallel(EMP, 4, default) */
   col1, col2, …
   from EMP;
```

- Both read of the data from the source table (EMP) and the write into the new table (EMP_) will be done in parallel

PQO Syntax for Index Builds

- Syntax for Indexes (parallel create clause)

```
 create index EMP$NAME on EMP (Name)
   parallel (degree 4 instances default)
   nologging
   tablespace USER_INDEX
   storage (initial 10M next 10M …);
alter index EMP$NAME noparallel;
```

- Without Alter statement cost-based optimizer gets confused!

How PQO Works

- PQO asks Oracle to use multiple processes for table/index builds
- Parallel Degree N for Tables --> N+1 processes
  - N Parallel Slaves
  - 1 Query Coordinator
- Parallel Degree N for Indexes --> 2*N+1 processes
  - N table scanning processes
  - N row Sorting processes
  - 1 Query Coordinator
  - Each sorting process may consume up to SORT_AREA_SIZE of memory

Choosing a Degree of Parallelism

- Parallel degree for Tables <= 2 * (# of CPUs)
  - To avoid CPU time contention
- Parallel degree for Indexes <= # of CPUs
  - Index creations should become CPU intensive if sorting is optimally done in memory

Query Server Pool

- Common set of parallel query server processes available in an instance
- To tune, use init.ora parameters:
  - **parallel_min_servers**: number of processes started when instance starts (eliminates overhead of frequent process startups and shutdowns)
  - **parallel_max_servers**: maximum number of process in query server pool. Recommended:

    2 * max_degree * num_of_reorgs

Query Server Pool

- To monitor contention for parallel query servers:

```
select Statistic, Value
from V$PQ_SYSSTAT
order by Statistic;
```

- If value of statistic “Servers Busy” is high, increase parallel_max_servers

Extent Sizes and PQO

- Number of parallel processes will affect extent allocation!
- When building tables or indexes, *each* degree of parallelism will result in the allocation of MINEXTENTS
- For example, creating a table with:

  PARALLEL degree 4; MINEXTENTS 2;

  INITIAL 20 MB; NEXT 20 MB

  will produce: 4 * 2 * 20MB = 160MB total allocation!

Extent Sizes and PQO

- To minimize over-allocation, choose smaller extent sizes
- Multiple extents (within reason) should not pose a problem
  - See Oracle Whitepaper #711: “The performance for DML is largely independent of the number of extents in the segments”
  - #711 outlines strategy for using multiple equally-sized extents to eliminate free space fragmentation at tablespace level
- Check for adequate freespace for both table and indexes!

ALTER SESSION Parameters

- ALTER SESSION can set certain parameters dynamically for reorganizing session, without affecting other users
- Consider:
  - db_file_multiblock_read_count
  - Sorting Parameters

db_file_multiblock_read_count

- Controls number of data blocks read for each read request during a full table scan (FTS)
- Significant performance boost is properly tuned, for example:

  OS read buffer = 64KB

  db_block_size = 4KB

  db_file_multiblock_read_count = 8

  – During FTS, each I/O operation will read:

    4KB * 8 = 32KB

  – Resetting db_file_multiblock_read_count = 16 gives:

    4KB * 16 = 64KB

db_file_multiblock_read_count

- Goal:

  db_block_size * db_file_multiblock_read_count

  = max OS read buffer

  - 64KB on older UNIX systems
  - 256K on NT

- In any case, db_file_multiblock_read_count cannot be larger than db_block_buffers / 4

Sorting Parameters

- **sort_area_size** (in bytes): maximum amount of memory for each sort
  - When using parallelism on index builds

    Total Sort Area = degree * sort_area_size

    - Oracle allocates Sort Area dynamically in 8K increments
  - Goal is:

    Total **sort_area_size** used = size of largest index reorg'd

    - If this requires too much memory, try using 50% or 25% of this amount plus 10% to minimize sort runs written to disk

Sorting Parameters

- **sort_direct_writes**: allows Oracle to bypass buffer cache when writing sort runs to temporary tablespace (Oracle 7.3.4 and 8.0.x)
- Can improve performance by factor of three!
- To use, set:

  sort_direct_write = true

  sort_write_buffers = 8

  sort_write_buffer_size = 65536
- **sort_direct_writes** obsolete in 8i
  - Sorts always use direct writes and automatically configure the number and size of direct write buffers

MTS Considerations

- For reorganizations, use a dedicated connection.
- This will use sort_area_size from the PGA instead of the SGA
- Set up dedicated connection in your tnsnames and use that service name for the job

Case Studies

Restructuring an OraApps DB

- Oracle Applications 10.7, Oracle 7.3.4
- HP-UX, 12 CPUs, 3.4 GB RAM
- EMC Model 3930 - 4 channel, 5 GB cache, SCSI card
- Relocated all tables (Over 3000 tables, 60 tablespaces) from older to newer EMC array
- Data volume restructured: 208 GB

Tuning Tips Used

- Enabled Sort_direct_writes
- Increased sort_area_size from 2M to 60M
- Increased sort_area_retained_size from 1M to 30M
- Tuned parallel query servers
- Implemented PQO degree 4 for large objects
- Unrecoverable

Results

- Trial run - 69 hrs
- Live run - 16 hrs
- Throughput 13GB / hr
- Cut run time of batch job from 12 to 5 1/2 hrs
- Optimized datafile size
- Regained 60GB disk - 29% Total
- Backups 50% faster
  - Due to faster disk array and smaller database size

Large SAP Table Reorg

- SAP R/3 version 4.0B - 13,887 tables
- SUN ES6000
- 20 CPUs
- 11GB RAM
- Oracle 8.0.5
- COEP: table 28.1 GB, indexes 19.5 GB

Tuning Tips Used for COEP

- Used Quest LiveReorg
- PQO - set to 8
- Degree 6 optimal for this configuration
- Tuned db_file_multiblock_read_count to stripe width - 256K
- NOLOGGING
- Enabled sort_direct_writes

Results

- Total runtime 5:50hrs: Tables - 2:05, Indexes - 3:45
- Total downtime 4 seconds!
- Throughput 8.2 GB / hr
  - Heavy tablespace & data file creation was running concurrently
- Regained 16.6 GB disk - 35% Total
  - Regained 12.1 GB on table
  - Regained 4.5 GB on indexes

Speed of PQO

- SAP Table CE1CPPA - 6GB
- Table copy *without* PQO - 3:14hrs
- Table copy *with* PQO - 1:35hrs

Tuning Tips Used for VBFA

- SAP table VBFA - 6 GB, 16 GB indexes
- Tuned sort_area_size
  - With 32 KB => Index builds took 3:36 hrs
  - With 32 MB => Index builds took 1:18 hrs

Conclusions

- Consider SQL-based reorgs
- Understand Oracle’s advanced options
- Tune your reorganizations for required performance
- When terminal velocity is not enough, consider LiveReorg