Oracle Data Pump Internals

Dean Gagne
Oracle USA
Nashua NH USA

Keywords:

Oracle Data Pump, export, import, transportable tablespace, parallel, restart

Introduction

Oracle Data Pump was a new feature introduced in Oracle Database 10g Release 1. The Oracle Data Pump export (expdp) and import (impdp) utilities are the replacements for the original export (exp) and import (imp) utilities. Oracle Data Pump enables very fast bulk data and metadata movement between Oracle databases. It is high speed, uses parallel threads, and is supported by command line clients (expdp and impdp) as well as a Web-based Oracle Enterprise Manager interface. Some of the original key enhancements over the original utilities are:

- Parallel execution
- Object selection
- Restart capability
- Network operations (export/import over a db link without using a dumpfile.)

Parallel Operations:

Oracle Data Pump users can specify the maximum number of parallel threads that the Oracle Data Pump utility can use. This value represents the maximum number of processes that the MCP can start in order to complete the transaction. This number does not include the MCP processes, which is described below. In some cases, the value specified for parallel is too large and Data Pump will only start the necessary additional threads to complete the task.

When a Data Pump job starts, a minimum of 2 processes are created. The first process is called the MCP (Master Control Process). Its function is to verify all parameters and the job description. It also controls and assigns work items to the Worker processes. The second process that gets started is the Worker process. This process will open communications with the Metadata API to export or import ddl and open communications with the Data layer to unload or load data. When the parallel parameter is not specified, or if a value of 1 is specified, then these are the only 2 processes that will be started for the job. If a parallel value greater than 1 is specified, then the additional processes will be created. These additional processes will be a combination of Worker processes and parallel execution (PQ) slave processes.

Parallel Export Operations:

When an Oracle Data Pump export job that contains both metadata and data is started with a parallel value of greater than 1, the first Worker process will export all of the metadata. The first task that this Worker process will perform is an estimate of what data needs to be exported. The information collected at this time is called ‘TABLE_DATA’ objects. These objects describe each subpartition of a

subpartitioned table, each partition of a partitioned table, and each table if it is not partitioned or subpartitioned. Once the estimate phase is complete, the MCP will use the remaining parallel processes to export the data associated with these TABLE_DATA objects, while the first Worker process will continue to export the metadata. The MCP process will determine which TABLE_DATA objects to assign to worker processes and it will also decide what value of parallelism each worker process will be assigned. If the worker is assigned a parallelism value larger than 1, then parallel execution slaves (PQ slaves) will be used to export the data in the TABLE_DATA object. When PQ slaves are used, the worker process is the query coordinator and it does not account towards the total parallel value.

Below are two examples of what could happen when exporting with parallel set to 6. The first example is with lots of data and the second example is with minimal data.

Example 1: Processes started with an export job with Parallel = 6 – lots of data
(data consists of 2 large subpartitions and 1 small subpartition)

- MCP – does not apply to the parallel 6 value
- Worker1 – Unload metadata – 1 parallel process
- Worker2 – Unload subpartition user1:tab1:subpart1 - 2 degrees of parallelism
    - Worker2_PQ1 – Unload data – 1 parallel process
    - Worker2_PQ2 – Unload data – 1 parallel process
- Worker3 – Unload subpartition user1:tab1:subpart2 - 1 degree of parallelism
- Worker4 – Unload subpartition user1:tab1:subpart3 - 2 degrees of parallelism
    - Worker4_PQ1 – Unload data – 1 parallel process
    - Worker4_PQ2 – Unload data – 1 parallel process

Example 1 summary: The sum of the processes doing work (not the MCP and not the worker processes acting as query coordinators) = 6. In this test case, you will see that there are 4 workers, but 2 of them are query coordinators. You will also see that there are 4 PQ slaves.

In some export jobs, there may not be enough data to use all available parallel processes. This is shown in example 2 below.

Example 2: Processes started with an export job with Parallel = 6 – very little data
(data consists of 1 small table)

- MCP – does not apply to the parallel 6 value
- Worker1 – Unloading metadata – 1 parallel process
- Worker2 – unloading small table user1:table1 – 1 process

Example 2 summary: only 2 workers will be seen even though parallel 6 was specified.

For export jobs, it is strongly suggested that you specify enough dumpfiles for the maximum number of parallel threads specified. Since only one process can write to a dumpfile, if you don't specify enough dumpfiles, the number of parallel threads running will be restricted to the number of dumpfiles that have been specified.

It is also suggested that different I/O controllers be specified when specifying dumpfiles. This will avoid potential I/O bottlenecks when multiple threads are all writing data to dumpfiles at the same time.

Parallel Import Operations:

Import jobs use parallel differently. Since there is a dependency issue with objects, some objects need to be created in certain orders. Tablespaces need to be created before users, users before tables, tables before indexes, etc. Because of the dependency issues, most metadata is created serially. There are some exceptions. Package bodies are created in parallel and indexes are built using parallel execution slaves. A typical import that included both data and metadata will follow these steps:

1. Create metadata in dependency order and stop after tables are created (all done serially)
2. Import data using up to max parallel.
   Note: The parallel algorithm for importing data is the same for exporting data. PQ slaves are used if beneficial, and processes are used up to the maximum parallel value specified.
3. Create metadata in dependency order until package bodies. (serially)
4. Create package bodies using parallel worker processes
5. Create metadata in "dependency order" until indexes (serially)
6. Create indexes serially, but use pq slaves to build them.
7. Finish importing remaining metadata. (serially)

The packages bodies are divided up to the maximum parallel value specified. For example, if you have 20 package bodies (assuming all are about the same size when exported) and parallel=5 was specified, the MCP would start 5 Worker processes and each would be assigned 4 package bodies.

For index creation, each index is created serially, but the parallel value in the index would be modified to the parallel value of the Data Pump job. If your index was created like:

```
CREATE UNIQUE INDEX "HR"."EMP_EMP_ID_PK" ON "HR"."EMPLOYEES"
 ("EMPLOYEE_ID")
  PCTFREE 10 INITRANS 2 MAXTRANS 255
  STORAGE(INITIAL 16384 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 505
  PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM" PARALLEL 3 ;
```

Then if you did your import with parallel=22, the create index would be executed as these 2 statements

```
CREATE UNIQUE INDEX "HR"."EMP_EMP_ID_PK" ON "HR"."EMPLOYEES"
 ("EMPLOYEE_ID")
  PCTFREE 10 INITRANS 2 MAXTRANS 255
  STORAGE(INITIAL 16384 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 505
  PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM" PARALLEL 22 ;
```

This would use up to 22 PQ Slaves to build the index as fast as possible. Once built, the following command would be executed.

```
ALTER INDEX "HR"."EMP_EMP_ID_PK" PARALLEL 3;
```

This restores the original parallel value to your index. Once Data Pump is complete, the goal is to restore the original settings, so the parallel value of 3 needs to be restored. While running Data Pump import, Oracle may choose to use less than 22 pq slaves to build the index, but Data Pump gives

Oracle the option to use up to 22. The actual value is determined by looking at the index, the associated table, how much data is in the table, etc.

Data Pump Data movement methods:

Oracle Data Pump uses three different methods to move data. The method chosen is decided by many different factors. These factors include the types used in the table definition, the method chosen by the user, whether or not a link is specified, whether or not a filter or a remap is specified, the size of the data in the table, and the total amount of data in the Data Pump job. The three methods available are Direct Path, External Table, and Insert as Select over a network link.

The types used in the table definition can eliminate certain methods. External tables does not support moving longs or long raws, so if the table has a long or long raw, external tables cannot be chosen. When importing into pre-existing partitioned tables, the partitioning scheme can be different due to changed character set from the source to the target. In this case, external tables would be used. Direct path is not allowed.

The customer could specifically choose the access method by using:

ACCESS_METHOD=EXTERNAL_TABLES

In this case, Data Pump will use external tables. If it is not possible to move data using this method, an error will be reported. This would be true if one of the tables in the job had a long column. In that case, the data associated with the table with the long column would not be imported and an error message would be reported back to the user.

If a network link, a query clause, or a remap_data transform is specified, the only possible solution would be to use external tables or Insert as Select over the network link.

If the job is a network import, the only solution is “Insert as Select”. The data will be inserted into a local table directly by selecting from the remote table.

If none of the above scenarios are present and the data can be loaded using any possible method, then the Master Control Process (described above) will choose the best method based on table size, job size, and the parallel value specified. If there is only 1 table and the table is large, the MCP will choose external tables with the maximum parallel value. This allows the worker process to use PQ slaves to move the data. If the job contains many smaller tables, the overhead cost of setting up external tables would not be advantageous. It would be much faster to use direct path and allow multiple workers to import data for single tables. Internal testing shows that Direct Path is approximately 2 times faster than external tables so if the table is small enough, it is possible that a direct path load or unload could finish before the external tables processes are even configured.

When data is being exported, it is desirable to export the larger tables first, especially when a parallel value of greater than 1 is specified. If this was not done this way, and if the largest table were to be exported last, and if it is not possible to use parallel threads to export that table, the remaining parallel processes will be idle while the data in this table is exported. If the same table were to be exported first, the remaining parallel threads could be busy exporting data from other tables, keeping them all busy and reducing the overall time for the export job to complete. With this algorithm, if a small table was last and could not make use of parallel threads, the remaining parallel threads would still be idle, but since the table is small, it would complete much faster.

Oracle Data Pump Master Table

When an Oracle Data Pump job is running, it is using a "Master Table" to keep track of where it is in a particular job. During export, when object information is written to the dump file, the location and a brief description is inserted into the master table. When export jb is completely exported, the contents of the master table are written to the dump file. During import, the first information imported is the master table. From here, the Data Pump packages can determine what was exported and determine what needs to be imported.

Some of the benefits of having this table is increased speed when performing an import operation and only importing a subset of the objects. In the exp/imp dumpfiles, the dumpfile was sequential. Let's say you exported 2 tables and 2 indexes. The dump file would look something like:

```
CREATE TABLE "SCOTT"."FOO" ...
CREATE TABLE "SCOTT"."BAR" ...
CREATE UNIQUE INDEX "SCOTT"."BAR_IND" ON "SCOTT"."BAR"...
CREATE UNIQUE INDEX "SCOTT"."FOO_IND" ON "SCOTT"."FOO"...
```

If all you wanted to import was the information for SCOTT.BAR (both table and index), then the imp utility would have to read the complete dumpfile to find the ddl associated with scott.bar and scott.bar_ind. With Data Pump, the xml is written to the dumpfile and a row is inserted into the master table indicating the object type, object schema, object name, dumpfile location. So, for Data Pump, the master table would have rows similar to this:

object_type	object_schema	object_name	dumpfile_location
TABLE	SCOTT	FOO	1234
TABLE	SCOTT	BAR	1235
INDEX	SCOTT	FOO_IND	1236
INDEX	SCOTT	BAR_IND	1237

Since scott.bar and scott.bar_ind are going to be imported, the Data Pump process just reads the information at block 1235 and 1237. The information for scott.foo and scott.foo_ind is never read. This improves performance of import jobs when filters are applied.

During a Data Pump job, the contents of the master table are kept current. This is beneficial in case a job is stopped. In order to restart a stopped job, the Data Pump packages query the master table to see where to restart.

Oracle Data Pump Export Restart

If an Oracle Data Pump export job is ever stopped, it can usually be restarted. A few jobs cannot be restarted. Some of those are: jobs using Transportable Tablespace, and jobs that have not completed the estimate phase. For export jobs that can be restarted, the key factor in getting it restarted is the existence of the master table.

During an export, while exporting metadata, the Worker process keeps track of what it is working on. It does this by entering rows in the master table. For example: if users are being exported, when the Worker process gets the metadata for the first user, it creates a row called a 'TYPE COMPLETION" row in the master table indicating that USERS are in progress of being exported. Some of the information in this type completion row is the object type, the current time is stored in a column called

start_time, etc. It then creates an OBJECT row for each type (shown above in the TABLE/INDEX example). When all of the users are exported, the Worker process will update the type completion row with a completed_time. This is repeated for each type of object exported.

If an export job were to be stopped (either manually, system failure, or other reason) the master table is not deleted. When this job is restarted, the Worker process will look for the type completion rows to see what had a start_time but did not have a completed_time. For example:

```
object_type   start_time                  completion_time
tablespace    12-sep-2012:9:04.01         12-sep-2012:9:05.23
user          12-sep-2012:9:05.27
```

In this situation, the Worker process detects that users were being exported. In this situation, Data Pump would drop all of the rows that describe users, drop the type completion row that describes that users were in progress, and remove the xml stored in the dumpfiles that describe users. It would then configure the metadata api to start after TABLESPCE, since tablespace is the last complete object type.

If there were no type completion rows with a null completion_time as in this example:

```
object_type   start_time                  completion_time
tablespace    12-sep-2012:9:04.01         12-sep-2012:9:05.23
user          12-sep-2012:9:05.27         12-sep-2012:9:07.34
```

the worker process would detect that nothing was in progress so no rows would have to be dropped, no XML would be removed from the dump files, and the metadata api would be configured to start after the 'USER' type.

If a job is stopped using the command line interface, be careful on which option you select. The option of stop will stop the job and allow it to be restarted. The option of kill will stop the job and will remove the master table. Once the master table is removed, the job can no longer be restarted.

Oracle Data Pump Import Restart

If an Oracle Data Pump import job is ever stopped, it can be restarted as long as the master table is present. The import restart code also uses the master table when restarting. Instead of looking at the type completion rows, it looks at the object rows. There are status and state fields in the object rows to detect which objects were imported and which object still need to be imported. Here is an example of the TABLE/INDEX example shown above. In this example, the table scott.foo and scott.bar were exported along with their indexes. At the beginning of import, the rows describing these objects would look like this:

```
Type  object_schema object_name  processing_state  processing_status
TABLE SCOTT         FOO          R                 C
TABLE SCOTT         BAR          R                 C
INDEX SCOTT         FOO_IND      R                 C
INDEX SCOTT         BAR_IND      R                 C
```

The state field of 'R' indicates that the objects were Retrieved (or exported). The status of "C" indicates that the objects are Current (or in good standing). If an import job was started and then stopped after scott.foo was imported and before scott.bar was imported, the rows would look like this:

```
Type  object_schema object_name  processing_state  processing_status
TABLE SCOTT         FOO          W                 C
TABLE SCOTT         BAR          U                 C
INDEX SCOTT         FOO_IND      R                 C
INDEX SCOTT         BAR_IND      R                 C
```

This shows that the table scott.foo was 'W' written or imported and the status of "C" shows it is current or in successful. For Table scott.bar, the import started to import this object, but never finished. The state of 'U', unknown, shows this. The object is still C current so it has not failed. When Data Pump import restarts, it will find this row and the import job will continue from here. The table Scott.bar will be imported again. After the table is create, the import job will continue with the remaining objects. In this case, those would be the 2 indexes.

Contact address:

Name

Dean Gagne
1 Oracle Drive
Nashua, NH 03062
U.S.A.

Phone: +1(603)897-3202
Email dean.gagne@oracle.com
Internet: www.oracle.com