|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | **What DataPump And Oracle RDBMS Parameters Can Significantly Affect DataPump Performance ? (Doc ID 1611373.1)** | [To Bottom](https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=526491383263372&id=1611373.1&_adf.ctrl-state=2m17dlafk_21)  [To Bottom](https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=526491383263372&id=1611373.1&_adf.ctrl-state=2m17dlafk_21) |  |        |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **In this Document**   |  |  | | --- | --- | |  | [Goal](https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=526491383263372&id=1611373.1&_adf.ctrl-state=2m17dlafk_21#GOAL) | |  | [Solution](https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=526491383263372&id=1611373.1&_adf.ctrl-state=2m17dlafk_21#FIX) | |  |  |  | | --- | --- | |  | [References](https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=526491383263372&id=1611373.1&_adf.ctrl-state=2m17dlafk_21#REF) |       **Applies to:**  Oracle Database - Enterprise Edition - Version 10.1.0.2 to 11.2.0.4 [Release 10.1 to 11.2]  Information in this document applies to any platform.  **Goal**  To provide a list of DataPump and RDBMS specific parameters that can significantly affect the speed of DataPump API based operations.  **Solution**  Data Pump Performance Affecting Parameters  At the Data Pump API level:  **ACCESS\_METHOD=...**  In some cases the access method chosen by the Data Pump API is not the fastest access method for your data set.  In such a case, you will not know this unless you explicitly try each of the available access methods for this parameter.  In other words, trial and error testing is the only way to know which will work best for your data set.  The current choice in access methods to choose from are DIRECT\_PATH and EXTERNAL\_TABLE.  **CLUSTER=N**  In a RAC environment it can improve the speed of Data Pump API based operations.     Note that since this parameter is limited to Data Pump API operations only, in a RAC environment, I would first recommend setting this parameter to N, and only if setting it to N has no effect would I also test setting the database parameter named PARALLEL\_FORCE\_LOCAL to a value of TRUE since PARALLEL\_FORCE\_LOCAL could have a wider scope of effect than just Data Pump API based operations.  **DATA\_OPTIONS=DISABLE\_APPEND\_HINT**  This impdp only parameter, under very specific conditions, can be safe to use to possibly reduce the time of importing data.  All of the conditions listed below must be met to be safely using the DISABLE\_APPEND\_HINT option of the DATA\_OPTIONS parameter ...  *Condition 1:*  The import operation is importing data into pre-existing tables, partitions or sub-partitions.  The term "pre-existing" is to mean in this case that the objects already exist at the time that the import job is started.  *Condition 2:*  The number of pre-existing objects being imported into is very small (Presumed to be 10 or less in number).  *Condition 3:     Only select statements will be issued by other RDBMS sessions against objects to be imported into while that import job is running.*     Note that the Data Pump specific import parameter value of DISABLE\_APPEND\_HINT for the parameter named DATA\_OPTIONS is available only as of version 11.2.0.1 and higher of the Data Pump API.     Note that the time saving benefit of using the DISABLE\_APPEND\_HINT value is only expected to be present if the wait times for locks to be released by other sessions are going to be significantly long.  **ESTIMATE**  The ESTIMATE parameter has two mutually exclusive possible assigned values to chose from, one is BLOCKS and the other is STATISTICS.  The BLOCKS approach at estimating the size of the data set that is being exported is expected to consume much more time to complete than using the STATISTICS approach for doing the same thing.  Presumably, the BLOCKS approach to estimating the size of the data set to be processed is more accurate than the STATISTICS approach to do the same thing.  Prior to 11.2.0.4 or with unpatched 11.2.0.3 versions of the RDBMS the ESTIMATE=STATISTICS effectively did the STATISTICS approach which is expected to be much less time consuming to perform then the BLOCK approach and then due to a bug the BLOCK approach was then executed in addition to the STATISTICS approach, even if only the STATISTICS approach is being chosen.  The end result was that if you chose the STATISTICS approach then the time to perform the estimate was actually longer than chosing the BLOCKS approach since when using BLOCKS approach only the BLOCKS approach was used performed.  Since 11.2.0.4 or with patched 11.2.0.3 versions of the RDBMS this ESTIMATE=STATISTICS bug is fixed and for faster Data Pump API based export operations, I would advise that ESTIMATE=STATISTICS be used, if the accuracy of the estimated size of the export dump file set is not a major concern.   EXCLUDE=COMMENT  In some cases, where there is no need to reference the comments on columns and object types by the RDBMS client software that is being used by the end users, this information can be omitted to reduce the amount of data to be processed by the Data Pump operation which in turn will reduce the time spent performing that Data Pump based operation.  If in doubt, then do not exclude COMMENT.   EXCLUDE=STATISTICS  If not needing to use the mutually exclusive INCLUDE parameter, then excluding the generation and export of statistics at export time will shorten the time needed to perform any export operation. The DBMS\_STATS.GATHER\_DATABASE\_STATS procedure would then be used at the target database once the import operation was completed.  Data Pump operations can hang, sometimes indefinitely, when concurrent statistics generation for a same table is being performed by both the Data Pump engine and any other RDBMS session.  For long running Data Pump operations (1 hour or more), consider disabling the database's automatic statistics gathering task.    To temporarily disable the 11g RDBMS automated statistics generating task so that the Data Pump operation does not encounter contention with this task ...    As user SYS execute the following commands ...    exec DBMS\_AUTO\_TASK\_ADMIN.DISABLE(client\_name => 'auto optimizer stats collection', operation => NULL, window\_name => NULL);    Then start the Data Pump operation and once the Data Pump operation has completed re-enable that task using the command below ...    exec DBMS\_AUTO\_TASK\_ADMIN.ENABLE(client\_name => 'auto optimizer stats collection', operation => NULL, window\_name => NULL);    To temporarily disable the 10g RDBMS automated statistics generating task so that the Data Pump operation does not encounter contention with this task ...    As user SYS execute the following commands ...    exec sys.dbms\_scheduler.disable ('GATHER\_STATS\_JOB');    Then start the Data Pump operation and once the Data Pump operation has completed re-enable that task using the command below ...    exec sys.dbms\_scheduler.enable ('GATHER\_STATS\_JOB');  **NETWORK\_LINK=...**  Performance wise, the use of this parameter further limits the effective degree of parallelism that can be used by the Data Pump API and unless your network throughput and network latency are better than a local drive, using NETWORK\_LINK will be expected to be significantly slower than using export dump files for the purposes of moving data.  With respect to Data Pump API performance, because of it`s tendency to be significantly slower than a dump file based Data Pump API operation, I would only advise the use NETWORK\_LINK as a last resort. I would consider using a portable or shared drive to write dump files to before considering the use of NETWORK\_LINK as a means of transfering data to and from databases.     Note that rules for distributed transactions apply when using a database link, so the use of NETWORK\_LINK means that the rules of distributed transactions apply to it`s use as well which means that distributed timeouts could be a concern for large data sets being moved.  **PARALLEL=Number\_of\_CPUs**  If there is more than one CPU available and the environment is not already CPU bound or disk I/O bound or memory bound and multiple dump files are going be used (ideally on different spindles) in the DUMPFILE parameter, then parallelism has the greatest potential of being used to positive effect, performance wise.  If the Data Pump parameter named PARALLEL is set to N where N > 1, then I would strongly recommend, for best use of parallelism, that the number of dump files defined by the Data Pump parameter named DUMPFILE be no less than N in quantity.  Please refer to [Note 365459.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&sourceId=1611373.1&id=365459.1) - Parallel Capabilities of Oracle Data Pump  **NOTE:**  The PARALLEL parameter is an upper limit on the number of concurrent Data Pump Worker processes that the Data Pump API based job can use, but the Data Pump API may choose to use a lower number of Data Pump Worker processes than the value assigned to that parameter.  Depending on which bottlenecks are in your host environment, it is possible that a Data Pump API based operation would be overall faster using a PARALLEL value that is less than the number of CPUs available to the Oracle RDBMS instance that is spawning the Data Pump Worker processes.  QUERY  The use of the QUERY parameter is expected to add significant overhead to any Data Pump API based operation.  The overhead is expected to be proportional to the number of rows in the table(s) queried.  **REMAP\_\***  The use of any REMAP\_\* parameter is expected to add significant overhead to any Data Pump API based operation.  The overhead is expected to be proportional to the number of objects to be imported or exported.  **NOTE:**  REMAP\_SCHEMA remaps only the ownership of an object for the CREATE object\_type schema.object\_name ... DLL section of a DDL statement. All other references to the old schema name within a DDL statement are left unchanged. Therefore, it is not unexpected that user defined PL/SQL code that explicitly references the old schema name within the body of the PL/SQL code will either fail to compile or may behave unexpectedly if the old schema name exists in the target database.  For PL/SQL routines, within the bodies of those routines the REMAP\_\* parameter will have no effect. All explicit references to the old name values of object types to be remapped will remain unchanged within a PL/SQL body.    At the Oracle RDBMS instance level:  **\_MEMORY\_BROKER\_STAT\_INTERVAL=999**  If resize operations are a significant time consumer in your slow Data Pump environment, then setting this parameter will reduce the frequency of resize operations which in turn will reduce the overall amount of time that resize operations delay other operations over a given time span.  Data Pump operations are particularly vulnerable to being delayed because of frequent streams pool resize operations.  This is because the Data Pump API depends a lot on the streams feature of the Oracle RDBMS to help perform export and import operations.  I would recommend a value of 999 for this parameter, if the parameter named STREAMS\_POOL\_SIZE is already set explicitely in value by the DBA (which gives the streams pool a minimum size) and frequent resize operations are occurring in your slow Data Pump environment.  **\_OPTIMIZER\_GATHER\_STATS\_ON\_LOAD=TRUE**  12c adds this new parameter that can slow down import operations when left at it's default setting of TRUE.  With 12c, if you intend on using EXCLUDE=STATISTICS for an import operation, then also set \_OPTIMIZER\_GATHER\_STATS\_ON\_LOAD=FALSE at the database level.      ALTER SYSTEM SET "\_OPTIMIZER\_GATHER\_STATS\_ON\_LOAD"=FALSE;     Perform the Data Pump based import operation with the EXCLUDE=STATISTICS setting.     ALTER SYSTEM SET "\_OPTIMIZER\_GATHER\_STATS\_ON\_LOAD"=TRUE;     It would be recommended, given the above scenario, to manually gather database statistics after that import operation has completed successfully..     EXEC DBMS\_STATS.GATHER\_DATABASE\_STATS;  **AQ\_TM\_PROCESSES=0**  When set explicitly to a value of 0, it can negatively impact the speed of Advanced Queue operations, which in turn can negatively affect Data Pump engine based operations since Data Pump uses Advanced Queueing.  Either leave this parameter unset or set this parameter to a value that is greater than 0.  **DEFERRED\_SEGMENT\_CREATION=TRUE**  For import operations only, the time spent performing space allocation for empty tables is eliminated. There is no expected significant performance benefit from setting this parameter to a value of TRUE from a data Pump API based export operation perspective.  This parameter is most useful when creating tables for an 11.2.0.2 or higher versioned environment where it is likely that a significant number of tables to be created will never be used.  **WARNING:**  Setting this parameter to TRUE will break backward compatibility, if you need to transfer table DDL metadata from an 11.2.0.2 Oracle RDBMS database or higher versioned Oracle RDBMS database to a pre-11.2.0.2 versioned Oracle RDBMS database where this functionality was not supported!  Please refer to: [Note 1087325.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&sourceId=1611373.1&id=1087325.1) - Error ORA-439 When Importing Tables Created With Enabled Deferred Segment Into Oracle 11g Standard Edition [Note 11812675.8](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&sourceId=1611373.1&id=11812675.8) - Bug 11812675 - Corrupt .dmp file / IMP-9 from direct path "exp" of no-segment table (conventional export)  Unpublished bug 8833245  - EXPORT IS SILENTLY DISCARDING NON-MATERIALIZED TABLES IN SCHEMA/DB MODE  Pre-11.2.0.2 Classic engine based import utilities, if unpatched, will not properly handle the CREATE DDL of tables that use that functionality and will likely raise an unexpected error when importing an empty table that was created while DEFERRED\_SEGMENT\_CREATION=TRUE.  Note that the database initialization parameter named DEFERRED\_SEGMENT\_CREATION when set to TRUE is not supported for use by Oracle RDBMS versions below 11.2.0.2.    **FILESYSTEMIO\_OPTION=...**  In the specific case where the RDBMS instance will be writting to an ACFS file system, given the nature of the types of write operations performed by the Data Pump API as part of an export operation, values other than NONE for this parameter can result in slow export operations.  Reference: [Bug 18253632](https://support.oracle.com/epmos/faces/BugDisplay?parent=DOCUMENT&sourceId=1611373.1&id=18253632) "WRITING EXPDP TO ACFS IS 5X SLOWER THAN WRITING ASM DISKGROUP/LINUX NATIVE FS" which was closed as not a bug.  **FIXED\_DATE=...**  Looking at note "Init.ora Parameter "FIXED\_DATE" Reference Note" (Doc ID 30724.1) it looks to me like FIXED\_DATE has never been completely supported by all features of the Oracle RDBMS, given that there are Oracle RDBMS feature compatibility bugs from 7.3.4.5 up to 11.2.  Whether they are called bugs or limitations, whether or not an RDBMS feature is fully compatible with the FIXED\_DATE parameter being set or not appears to be on an RDBMS feature-by-feature basis.  In the case of FIXED\_DATE and the Data Pump API or perhaps PL/SQL in general, the end result may or may not be a hang.    **NLS\_CHARACTERSET=... and NLS\_NCHAR\_CHARACTERSET=...**  Import operations for partitioned tables that get created and populated by the import operation will not be able to use more than one Data Pump worker for a specific partitioned table at any time, when either of these two database parameters differ between the source database and the target database. In such cases, only one Data Pump worker at a time can work with the table data as it will hold an exclusive lock on that table which in turn will prevent any other Data Pump workers from working with that same table. When no exclusive lock is held on a partitioned table it is possible for more than one Data Pump worker to be used for that partitioned table which can in some circumstances improve the speed at which data is imported into the partitioned table. See note "DataPump Import (IMPDP) In Parallel Is Slow When Importing Partitioned Tables" ([Doc ID 1681444.1](https://support.oracle.com/ui/km/DocumentDisplay.jspx?id=1681444.1)).  **NLS\_COMP=... and NLS\_SORT=...**  In some rare cases, having both of the database initialization parameters set to a value of BINARY will result in a significantly faster Data Pump API based operation.  Whether or not these parameters being set to BINARY will help performance in your environment has to be tested in your environment as the root cause of performance increase has not been investigated.  The after logon trigger listed below sets these two above mentioned NLS parameters at the session level only for an RDBMS session that logged into the database instance via an RDBMS connection that was established by a Data Pump utility.  CREATE OR REPLACE TRIGGER sys.expdp\_nls\_session\_settings AFTER LOGON ON DATABASE  DECLARE   V\_MODULE VARCHAR2(60);  BEGIN  SELECT SYS\_CONTEXT ('USERENV', 'MODULE') INTO V\_MODULE FROM DUAL;  IF UPPER(V\_MODULE) LIKE 'UDE%'  THEN   BEGIN   EXECUTE IMMEDIATE 'ALTER SESSION SET NLS\_COMP=''BINARY''';   EXECUTE IMMEDIATE 'ALTER SESSION SET NLS\_SORT=''BINARY''';   END;  END IF;  END;  /    **NOTE:  Logon triggers are ignored by Data Pump for releases < 11.2.  Please refer to:** [**Bug 5724679**](https://support.oracle.com/epmos/faces/BugDisplay?parent=DOCUMENT&sourceId=1611373.1&id=5724679) **- LOGON TRIGGER NOT FIRED FOR DATA PUMP MASTER AND WORKER PROCESSES  A few interim patches are available (for 10.2.0.4 and 11.1.0.7).**  **PARALLEL\_FORCE\_LOCAL=TRUE**  In a RAC environment it can improve the speed of Data Pump API based operations and avoid some Parallel DML bugs at the same time.     Note that the database initialization parameter named PARALLEL\_FORCE\_LOCAL is available only as of version 11.2.0.2 and higher of the Oracle RDBMS software.  **STREAMS\_POOL\_SIZE**  To avoid bug 17365043 '"STREAMS AQ: ENQUEUE BLOCKED ON LOW MEMORY" WHEN REDUCING STREAMS\_POOL\_SIZE' I would recommend setting the STREAMS\_POOL\_SIZE to the value returned by the result set of the following query.  select 'ALTER SYSTEM SET STREAMS\_POOL\_SIZE='||(max(to\_number(trim(c.ksppstvl)))+67108864)||' SCOPE=SPFILE;'  from sys.x$ksppi a, sys.x$ksppcv b, sys.x$ksppsv c  where a.indx = b.indx and a.indx = c.indx and lower(a.ksppinm) in ('\_\_streams\_pool\_size','streams\_pool\_size');  At the table DDL level:  **NETWORK\_LINK+SECUREFILES**  The NETWORK\_LINK parameter when used to move tables with LOB columns that are configured to use SECUREFILES make for an extremely slow transfer of that LOB data when compared to exporting the LOB containing tables to a file, and it has been observed that using the NETWORK\_LINK parameter to move tables with LOB columns defined to use securefiles generates a lot of undo data.  Part of the cause of the difference in speed as of 11.2.0.3 is that distributed transactions is that allocation requests are limited to one block at a time over a database link which means more round trips to get larger data sets transferred.  **SECUREFILES (Without NETWORK\_LINK)**  The storage of LOB column data using the SecureFiles storage format does not allow tables with LOB columns to be exported and imported using parallelism for RDBMS version below 12.2.  The storage of LOB column data using the BasicFiles storage format does not allow tables with LOB columns to be exported or imported using parallelism.    At the table DML level:  Contention between the Data Pump operation and another RDBMS session for access to RDBMS objects (Typically for locks on table row data).  The Data Pump engine during an export operation will wait for most types of locks on rows and tables that are held by other RDBMS sessions to be released before it will proceed to export or import rows from the associated table.  The Data Pump engine during an export operation does wait for some types of locks on rows and tables that are held by other RDBMS sessions that an export performed by the classic export utility did not wait for.  Therefore exporting a table that is being actively updated will be expected to be slower to do than exporting that same table when it is not being updated.    Things that a Data Pump operation has no control over which affect the speed of Data Pump operations:   * Compilation speed of Java Stored Procedures during an import operation. * Compilation speed of PL/SQL packages during an import operation. * Constraint creation speed (Foreign key and Primary key) during an import operation. * Index creation speed during an import operation. * The speed at which the creation DDL of an object is processed if that object's creation DDL has dependencies on remote objects. * An undersized SGA. * Any lack of operating system or hardware resources not due to an excessive use of said resources by the Data Pump operation. * Locks already held by other RDBMS sessions on RDBMS objects that the Data Pump operation also wants to have access to.   **References**  [NOTE:453895.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&sourceId=1611373.1&id=453895.1) - Checklist For Slow Performance Of DataPump Export (expdp) And Import (impdp) | | |