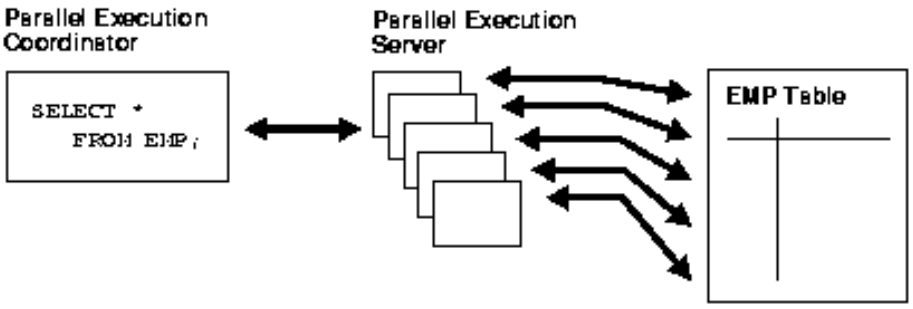
Alter session to force parallel query doesn't work, Why?

Sometimes DBAs need to run a specific Query's in parallel to using CPU Utilization and run faster but parallel setting doesn't work although we force it!



(Degree of Parallelism | DOP)?

The degree of parallelism (DOP) is the number of parallel execution servers associated with a single operation. Parallel execution is designed to effectively use multiple CPUs.

Don't enable parallelism for small objects

Small tables/indexes (up to thousands of records; up to 10s of data blocks) should never be enabled for parallel execution. Operations that only hit small tables will not benefit much from executing in parallel, whereas they would use parallel servers that you want to be available for operations accessing large tables.

Best practices when using object sizes as the main driving factor for parallelism are commonly aligning the DOP with some kind of step function for parallelism, e.g.

Objects smaller than 200 MB will not use any parallelism

Objects between 200 MB and 5GB are using a DOP of 4

Objects beyond 5GB are getting a DOP of 32

Needless to say that your personal optimal settings may vary - either in size range or DOP - and highly depend on your target workload and business requirements only.

Type of execution:

1) Query

You can enable parallel execution and determine the DOP in the following priority order:

Hint / session / table

The DOP is limited by the Oracle Database Resource Manager (DBRM) settings.

For example, if your resource plan has a policy of using a maximum DOP of 4 and you request a DOP of 16 via a hint, your SQL will run with a DOP of 4.

2) Hint

Oracle will make use of Parallel Loading when “Parallel” hints is used in a query block SQL Statement. The requested DOP for this query is DEFAULT.

```
select /*+ parallel(c) parallel(s) */ c.state_province , sum(s.amount) revenue from customers c , sales s

where s.customer_id = c.id

and s.purchase_date between to_date('01-JAN-2023','DD-MON-YYYY') and to_date('31-DEC-2023','DD-MON-YYYY')

 and c.country = ‘IRAN’ group by c.state_province ;
```

This method is mainly useful for testing purposes, or if you have a particular statement or few statements that you want to execute in parallel, but most statements run in serial.

The requested DOP for this query is 16 for the s table (sales)

```
Select /*+ parallel(s, 16) */ count (*) from sales s;
```

3) Session

a) Enable

The PARALLEL parameter determines whether all subsequent query statements in the session will be considered for parallel execution.

Force: If no parallel clause or hint is specified, then a DEFAULT degree of parallelism is used.

```
Alter session force parallel query;
```

This force method is useful if your application always runs in serial except for this particular session that you want to execute in parallel. A batch operation in an OLTP application may fall into this category.

```
ALTER session enable parallel query;
```

b) Disable

```
Alter session disable parallel query;
```

```
SELECT DISTINCT px.req_degree "Req. DOP",  px.degree "Actual DOP"FROM v$px_session px

WHERE px.req_degree IS NOT NULL;

Req. DOP           Actual DOP
----------         ---------
16                 16
```

4) Table

a) Enable

ALTER TABLE <table name> PARALLEL 32;

ALTER TABLE <table name> PARALLEL (DEGREE 32);

ALTER TABLE <table name> PARALLEL (DEGREE DEFAULT);

Use this method if you generally want to execute operations accessing these tables in parallel.

Tables and/or indexes in the select statement accessed have the parallel degree setting at the object level. If objects have a DEFAULT setting then the database determines the DOP value that belongs to DEFAULT.

For a query that processes objects with different DOP settings, the object with the highest parallel degree setting accessed in the query determines the requested DOP.

b) Disable

ALTER TABLE table_name NOPARALLEL;

- Get

SELECT table_name, degree FROM user_tables WHERE table_name='MyTableName';

DML

To enable parallelization of Data Manipulation Language (DML) statements such as INSERT, UPDATE, and DELETE, execute the following statement.

Alter session enable parallel DML;

Problem:

Sometimes DBAs need to run a specific Query's in parallel to using CPU Utilization and run faster but parallel setting doesn't work although we force it!

Force should mean that some behavior would always happen (when possible), right?

Let's doing a test:

SQL> CREATE TABLE MYTABLE AS SELECT * FROM dba_objects;

SQL> CREATE INDEX MYINDEX ON MYTABLE (owner);
SQL> begin
DBMS_STATS.GATHER_TABLE_STATS(ownname => NULL, tabname =>'MYTABLE', ESTIMATE_PERCENT =>
75,METHOD_OPT => 'FOR ALL COLUMNS SIZE AUTO', CASCADE => TRUE, no_invalidate=>false);
end;
/

Now let's "force" the parallel query in my session, run the query and check the execution plan:

SQL> ALTER SESSION FORCE PARALLEL QUERY PARALLEL 2;

```
SQL> SELECT SUM (object_id) FROM MYTABLE WHERE owner LIKE 'S%';
SUM (OBJECT_ID)
--------------
    979900956
SQL> select * from table (dbms_xplan.display_cursor);
-------------------------------------------------------------
| Id | Operation | Name | E-Rows |E-Bytes| Cost (%CPU)|
-------------------------------------------------------------
|   0 | SELECT STATEMENT                          |       |       |       |  186 (100)|
|   1 | SORT AGGREGATE                            |       |     1 |    12 |       |
|   2 |   TABLE ACCESS BY INDEX ROWID| MYTABLE     |  6741 | 80892 |   186   (0)|
|* 3 |    INDEX RANGE SCAN | MYINDEX              |  6741 |       |    18   (0)|
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
  3 - Access ("OWNER" LIKE 'S%')
      Filter ("OWNER" LIKE 'S%')
```

The output shows a regular, serial execution plan!

Ohhhh !!, let's increase the "forced" parallelism from 2 to 3 and run exactly the same query again:

```
SQL> ALTER SESSION FORCE PARALLEL QUERY PARALLEL 3;
SQL> SELECT SUM (object_id) FROM MYTABLE WHERE owner LIKE 'S%';
SUM (OBJECT_ID)
--------------
 979900956
SQL> select * from table (dbms_xplan.display_cursor);
-------------------------------------------------------------------------------------------------------
| Id | Operation            | Name     | E-Rows |E-Bytes| Cost (%CPU)|    TQ |IN-OUT| PQ Distrib |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |        |       |       |  128 (100)|       |     |          |
|   1 | SORT AGGREGATE           |        |     1 |    12 |           |       |     |          |
|   2 |   PX COORDINATOR         |        |       |       |           |       |     |          |
|   3 |    PX SEND QC (RANDOM) |:TQ10000 |     1 |    12 |           | Q1,00 | P->S | QC (RAND)  |
|   4 |     SORT AGGREGATE       |        |     1 |    12 |           | Q1, 00 | PCWP |          |
|   5 |      PX BLOCK ITERATOR   |        |  6741 | 80892 |   128   (0)| Q1, 00 | PCWC |          |
|* 6 |       TABLE ACCESS FULL  | MYTABLE      |  6741 | 80892 |   128   (0)| Q1, 00 | PCWP |          |
-------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
  6 - Access (:Z>=:Z AND :Z<=:Z)
      Filter ("OWNER" LIKE 'S%')
```

Now the query will get a parallel plan!

The reason for this behavior is that the FORCE parallel query syntax doesn't really *force* Oracle to use a parallel plan, but rather just reduces optimizer cost estimates _for full table scans _ (the higher the parallelism, the lower the FTS costs – Jonathan already has details about this in his blog entry, so I won't replicate this).

But the optimizer is still free to choose some other, non-parallel execution plan if that has a lower cost than the best parallel one!

So what happened above is that with "forced" parallel degree 2, the parallel full table scan plan must have had a higher cost than the serial index range scan (186), but once I increased the parallelism "factor" to 3, then the final cost of the parallel full table scan plan ended up being lower (128) than the best serial plan found.

This is a good example showing that both the PARALLEL hints and the FORCE PARALLEL session settings really just adjust a narrow set of optimizer cost computation inputs and don't really fix the resulting execution plan. If you really want to fix an execution plan, you need to tie optimizer "hands" in every aspect with a full set of hints just like the stored profiles do. That way, even if there is a lower cost plan available, the optimizer doesn't know about it as you've prohibited it from doing any calculations other than your hints direct it to.

Note that when testing this, your mileage may vary, depending on how much data you have in your test table (or rather in the optimizer stats for that table) plus system stats.

Enabling Parallel SQL Execution

You enable parallel SQL execution with an ALTER SESSION ENABLE PARALLEL DML|DDL|QUERY statement. Subsequently, when a PARALLEL clause or parallel hint is associated with a statement, those DML, DDL, or query statements will execute in parallel. By default, parallel execution is enabled for DDL and query statements.

Forcing Parallel SQL Execution

You can force parallel execution of all subsequent DML, DDL, or query statements for which parallelization is possible with the ALTER SESSION FORCE PARALLEL DML|DDL|QUERY statement. Additionally you can force a specific degree of parallelism to be in effect, overriding any PARALLEL clause associated with subsequent statements. If you do not specify a degree of parallelism in this statement, the default degree of parallelism is used. Forcing parallel execution overrides any parallel hints in SQL statements.

-- 12c ADAPTIVE:

--select * from table (dbms_xplan.display_cursor (null, null,'ALLSTATS LAST +PEEKED_BINDS +PARALLEL +PARTITION +COST +BYTES +ADAPTIVE'))

-- 19c HINT_REPORTING:

-- select * from table (dbms_xplan.display_cursor (null, null,'ALLSTATS LAST +PEEKED_BINDS +PARALLEL +PARTITION +COST +BYTES +HINT_REPORT'))

Done.