Manage Concurrent Parallel Execution without Tuning

Jean-Pierre Dijcks
Oracle
Redwood City, CA, USA

Keywords:

Workload management, data warehouse, database, parallel execution, SQL Tuning

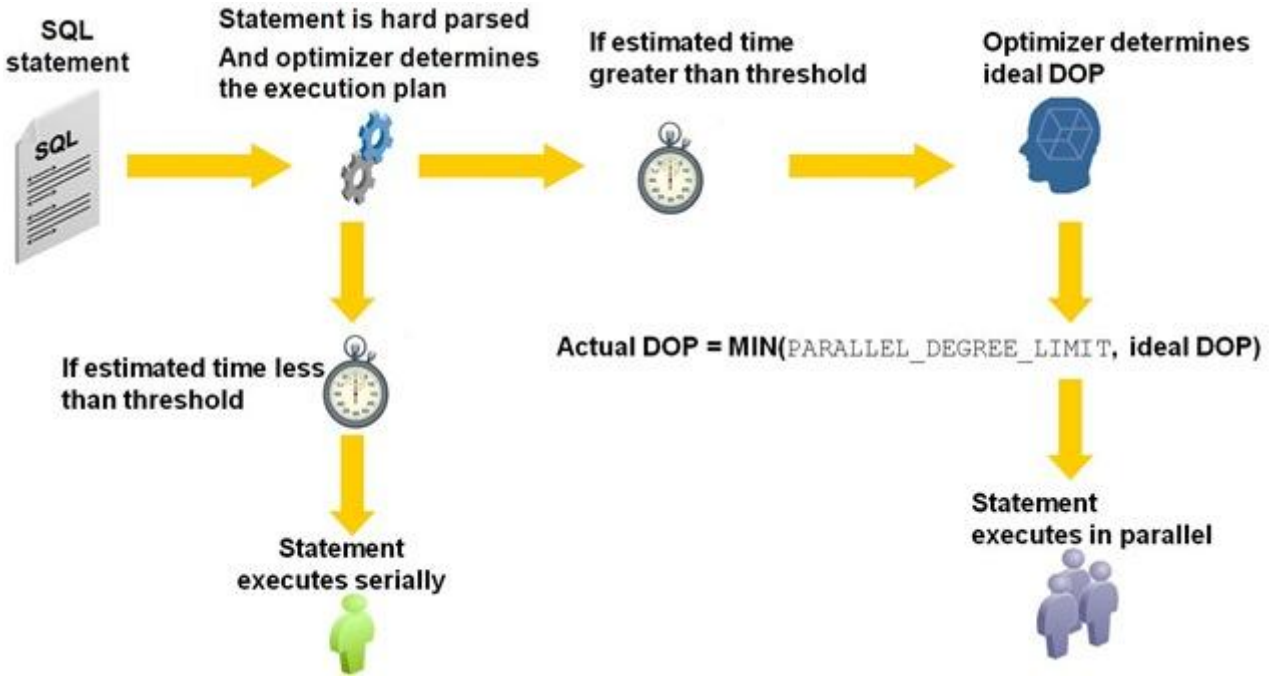
Introduction

Some of the information of this paper can be found in this great paper on OTN, but I wanted to also share a little bit on the practical side of really applying this... One of the goals we set out for in 11g Release 2 of the database is to make life simple and easy when using parallel processing.

Automatic DOP

To make this all simpler for you, we are doing things like Automatic DOP which is - as the name says - a way of having Oracle determine the degree of parallelism based on a set of criteria and some initialization parameter settings. For the casual reader, this is only available as of 11g Release 2 of Oracle Database, so don't go look for the parameters in an older release.

To enable the features in 11g Release 2, use the parallel_degree_policy parameter (by default this stuff is off - parameter is set to manual). For Auto DOP, setting this to limited is sufficient. If you want more functionality (in-memory parallel processing and parallel statement queuing parallel_degree_policy should be set to auto).
As the name says and the above description implies, Auto DOP, or Automatic Degree of Parallelism is Oracle determining the parallel degree with which to run a statement (DML, DDL and queries) based on the fastest possible plan as determined by the optimizer. That means that the database parses a query, calculates the cost and then calculates a DoP to run with. The cheapest plan may be to run serial, which is also an option. The following illustrates the decision making process:



The threshold that is prominently mentioned above is set by parallel_min_time_threshold. The default of this parameter is 10 seconds. If you want to run more statements in parallel, make sure to reduce that number so that more plans qualify for parallel evaluation.

The other parameter mentioned in the above is the parallel_degree_limit. Parallel_degree_limit is the maximum DOP that can be used. By default this is set to the value of "CPU". In this case CPU means Default DOP, which in turn is calculated as the total CPU count of the system (if this is a cluster, it includes all CPUs in the cluster)multiplied with the value of the parameter parallel_threads_per_cpu. On a 32 CPU system (cluster or single machine) this is 32 * 2 = 64.

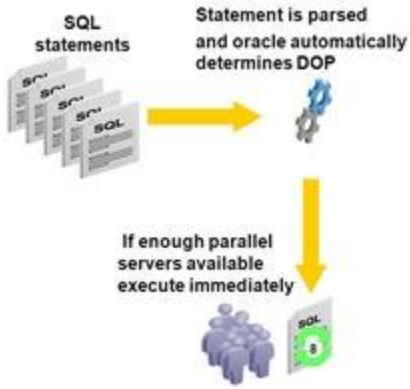
The DOP that we run the statement is the minimum value of the computed DOP (or ideal DOP) and that parallel_degree_limit parameter. Within the explain plan you will see the actual DOP and whether or not we capped it. If we cap it you see something like " degree of parallelism is 64 because of parallel threshold".

Should you choose to go for auto DOP you may see many more statements running in parallel, especially if the threshold is low. BTW low is relative to the system and is not an absolute quantifier. In other words, on some systems 0.1 second is low, on others 20 seconds is low...

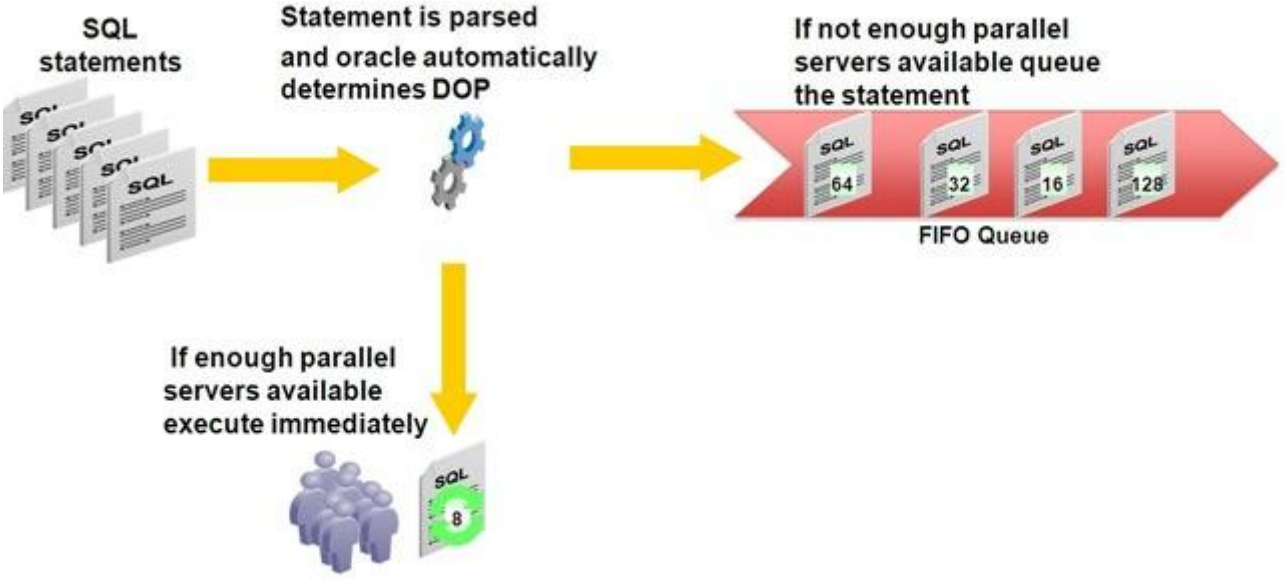
Parallel Statement Queuing

Because of the expected behavior of more statements running in parallel it becomes more important to manage the "scarce resources" of parallel processes available. That means that the system should be smart about when to run a statement and verify the requested number of parallel processes are around. Requested number of processes in this is the DOP for that action.

The answer to this is Parallel Statement Queuing. The long and short of parallel statement queuing is that a statement runs when its requested DOP is available. E.g. when a statement requests as DOP of 64, it will not run if there are only 32 processes currently free to assist this customer. As with the annoying telephone systems, the statement will be placed into a queue. However, Oracle does enforce a strict First In - First Out queue, never so sure about the phone systems on that one...
In the picture below you can see this in action. Let's say a statement requests to be run with a DOP of 8. Enough processes are around and available, and Oracle just runs the statement with DOP of 8.



Not all that interesting. Now if a statement comes in that wants to run with a DOP of 128 and there are no 128 parallel processes available to start working, the statement gets queued. There is a threshold before this is starting, but more on that later.



The statement requesting DOP 128 is being queued and is first in the queue as shown above. Subsequent statements requesting DOPs of 16, 32 and 64 are nicely queued as well, behind

our first arrival. All running statements will of course run and complete. Once the 128 processes are around, our first in the queue will run and the queue will clear out as more processes come available.

You will only get parallel statement queuing once you reach a certain threshold (and if parallel_degree_policy is set to auto). That threshold is called parallel_server_target. The default value for parallel_server_target is set to 4 times the default DOP. On our system above with 32 CPUs and a default DOP of 64, the threshold is 256. This means that as soon as 256 parallel processes on this box are busy, queuing starts to happen.
A random example could be that there is a session running with DOP of 128, one is running with 64 and four statements are running with DOP of 16. Now if the above picture occurs, e.g. a statement requires DOP of 128, that statement is queued.
To avoid an arbitrary number of parallel processes to be running on a system, which may overload that system, the parameter parallel_max_servers provides a hard upper boundary. That boundary is not displaced by parallel_server_target.

I often get the question what to set the parallel_server_target to... and of course the answer was: that depends. You don't want to set parallel_server_target to the same value as parallel_max_servers, because queuing is not really benefiting you then. You also do not want to set it too low. With the above numbers and 32 CPUs, 32 is probably on the low end.
Queuing will cause wait events on the system. These wait events are "ENQ JX SQL statement queue" to indicate that a statement is queued in the parallel statement queue and "PX Queuing: statement queue" to indicate that a statement is in the queue and that it is the next one being executed. E.g. it is the one with DOP 128 in the above picture. The nicest way to see this is by using the Enterprise Manager SQL Monitor, but you can also use the V$SQL_PLAN_MONITOR view:

```
SELECT s.sql_id, s.sql_text
FROM v$SQL_MONITOR m, v$SQL s
WHERE m.status='QUEUED'
AND m.sql_id = s.sql_id;
```

If you need to bypass the queue, use the NO_STMT_QUEUING hint. This will bypass the queue. This is also a reason to leave some headway between the moment the queuing starts and the maximum parallel server processes you allow. If queuing starts to late and you need to run that crucial query, Oracle may downgrade it to a lower DOP than requested because you bump your head onto the maximum allowed number of parallel processes.
If you do not want to go to auto on parallel_degree_policy you can use the hint STMT_QUEUING, and you will get queuing on the system. Do make sure you have the other parameters set so we start queuing appropriately.

One more thing...
In our best practices we recommend that you set parallel_execution_message_size to 16KB allowing sufficiently large message buffers for communication between producers and consumers and between parallel processes in general.

And by the way... as a small update on the post on in-memory parallel execution, make sure that you size the interconnect of the machine appropriately when using in-memory parallel execution. This is key to getting the most out of the in-memory execution.
Then throw compression into the mix and you are going to get some real cool performance numbers out of the parallel stuff... but more on that some other time.

Now by the time you all read this, 11.2.0.2 is out for the general public. And within 11.2.0.2 you will need to learn and start using Database Resource Manager to manage the queues. As you will see in the DOAG session, we have now expanded the queuing to have a parallel statement queue PER consumer group in DBRM.

Contact address:

Jean-Pierre Dijcks
500 Oracle Parkway, M/S 4op7
Redwood City, CA, 94065

Phone: +1 650 607 5394
Email jean-pierre.dijcks@oracle.com
Internet: blogs.oracle.com/datawarehousing