This document contains a list of the differences between the latest version of the Tuning Parallel Execution paper that is online and the version published in Select Journal. I've given page numbers for the various source documents. I have also given the initials of the reviewer who prompted the change although how I've decided to implement their suggestion is down to me and you shouldn't assume they approve of the final version!

Reviewers

Carl Dudley (mainly style improvements, so none listed here)
Tom Kyte (TK)
Jonathan Lewis (JL)
Mark Rittman (MR)
IOUG editors (IOUG)

As a general note, I'm still unhappy with some of the diagrams so will produce new versions in future.

Cheers,

Doug

Select page 5
PDF/DOC page 1
HTML px.html

- Reduced the length of the introductory paragraphs from the original UKOUG paper. (IOUG and me)

- Changed 'Parallel Query Option' to 'Parallel Query'. I wanted the line to be historically accurate, but it might confuse people that this was still an *option* (TK)

- Changed 7.1 to 7.1.6 to be more specific about the version it was introduced (TK)

- Changed 'One of the foundations of the architecture is multiple processes running in *parallel*' to ' … running *concurrently*' to avoid confusion. (JL)

- Changed 'Sensibly Oracle limits those tasks that can use Parallel Execution to those that are likely to benefit from it' to 'As Oracle's parallel capabilities have been developed, most tasks can be executed in parallel now.' (JL)

- Changed 'index full scans' to 'index fast full scans' (JL)

Select page 6
PDF/DOC page 3
HTML px2.html

- Changed the example slave SQL statement to remove the OPS$ORACLE schema owner that I'd pasted from a test session so that there were no special privileges that could muddy the waters. (JL)

- Added a mention that the SQL associated with the slaves in 10G is the same as the parent SQL statement executed by the query coordinator. This is also mentioned later, but it improved the flow of the document to include it here. (JL)

- Added a footnote to emphasise that using parallel execution implies the use of the cost based optimiser, regardless of the presence of statistics. (TK)

- Changed Figure 3. Added another box for the Query Co-ordinator to indicate that it acts as the Ranger to calculate the correct sort ranges for the slaves to sort half of the rows and changed the sort ranges from A-M and N-Z to A-J and K-Z as an additional symptom of this. I'm still thinking of how to improve this diagram further.

PDF/DOC page 4

- Changed 'For the sort, each slave takes half of the possible range of values' to 'For the sort, the QC process acts as a Ranger and divides up the sort activity by calculating the correct range of values for each slave to sort so that they'll process a reasonably equal number of rows.' (JL)

- Changed the following section significantly (JL)

  "It is essential that each PX slave in a given set must be able to communicate with *all* of the slaves in the other set. Even with a DOP of two, you can see this means 4 inter-slave connections. As the DOP increases, the number of connections will increase exponentially.

  The guaranteed maximum number of processes required for a given query is 2 x *DOP* plus the Query Co-ordinator. If there are multiple step operations in the plan, then the two sets of slaves will be re-used."

  to :-

  "Each PX slave in a given set must be able to communicate with *all* of the slaves in the other set so as the DOP increases, the number of connections will increase rapidly. As Jonathan Lewis points out in his article 'The Parallel Query Option in Real Life' (http://www.jlcomp.demon.co.uk/pqo_2_i.html), the number of inter-slave communication paths is DOP-squared. Even with a DOP of two, you can see this means four inter-slave connections, a DOP of four would need 16 connections and so on.

  The maximum number of processes required for a simple query is 2 x *DOP* plus the Query Co-ordinator. However, if there are multiple DFOs in the plan then additional PX slaves may be acquired and slave sets may be re-used. "

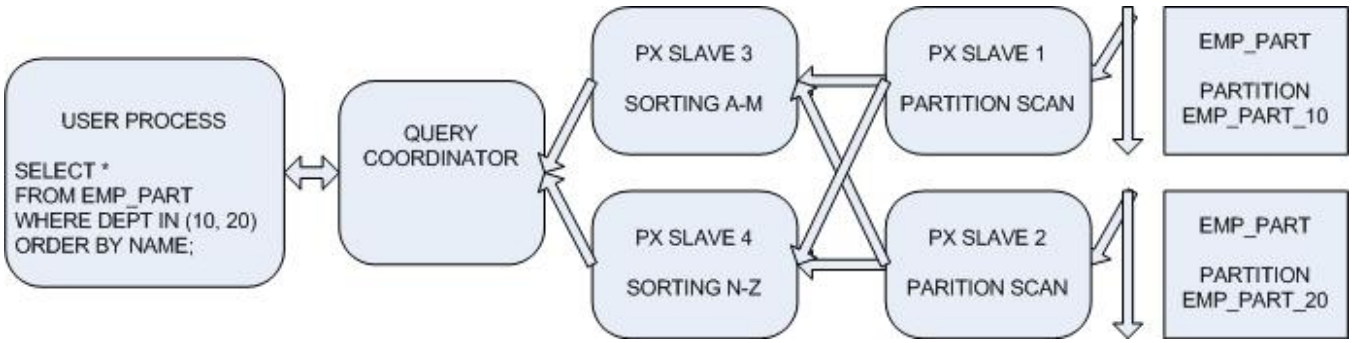
  With the following footnote :-

  Jonathan Lewis sent me an example statement that will fire up 34 slaves on Oracle 10g and 22 slaves on 9i, so things are more complicated than they might appear!

- Removed the following section and Figure 4 completely because they were completely wrong. (TK and JL)

  Finally, it's worth considering what happens when we use Parallel Execution against a Partitioned Table, because Oracle behaves differently.

  Figure 4 – Parallel Partition Scan with Sort – Degree 2



In this case, the maximum DOP that Oracle can use is the number of separate partitions that will be scanned because, instead of using block ranges, Oracle divides the work up by partitions. It's worth noting that this could lead to less than impressive results if you don't have a fairly even distribution of rows in each partition. For example, if EMP_PART_10 contained only 3,000 rows, but EMP_PART_20 contained 35,000, Oracle would still have to wait for PX SLAVE 2 to complete it's work before the sort operation could be completed, so PX SLAVE 1 might be idle for much of the query processing. This isn't necessarily a problem, but our query would be processed more efficiently if each partition had 19,000 rows.

Select page 7
PDF/DOC page 5
HTML px3.html

- Added the following caveat – "Important – I've tried to be helpful in suggesting some initial values for these parameters, but your database and application is unique, so you should use these suggestions as starting points. There are no easy answers."

- Changed some of the wording in the section on parallel_adaptive_multi_user to emphasise that it's a good solution to a tricky problem as long as you're aware of the implications. (TK)

- Changed the recommended setting of parallel_adaptive_multi_user from 'Depends' to 'True' (TK)

Select page 8
PDF/DOC page 7
HTML px4.html

- Changed the recommended range of values for parallel_max_servers from 2-10 * Number of CPUs to 2-4 * Number of CPUs (Several sources)

- In 'Other Significant Parameters' - changed 'sort areas' to 'work areas' (TK)

Select page 9
PDF/DOC page 9
HTML px6.html

- Changed the query against v$pq_tq_stat to use the following ORDER BY clause which lists the slaves in the order of activity (JL)

```
ORDER BY dfo_number DESC, tq_id, server_type DESC, process;
```

- Added the break statement to improve the report formatting (JL)

```
break on dfo_number on tq_id
```

- Added a few footnotes discussing the previous two changes

Select page 11
PDF/DOC page 12
HTML px7.html

- Added a comment explaining some of the difficulties with analysing px slaves in 10g (JL)

  "I think this makes it much easier to see at a glance what a particular long-running slave is really doing, rather than having to tie it back to the QC as on previous versions. However, it makes things much trickier when trying to access the execution plan for different slaves"

- Added a footnote with a link to Mark Rittman's PX tracing blog entry. (MR)

Select page 12
PDF/DOC page 15
HTML px9.html

- Changed the following sentence (JL)

  "The slower your I/O sub-system, the more benefit you are likely to see from PX - but shouldn't you fix the real problem?"

  To

  "Sometimes when faced with a slow i/o subsystem you might find that higher degrees of parallelism are useful because the CPUs are spending more time waiting for i/o to complete. Therefore they are more likely to be available for another PX slave (that isn't waiting on i/o) to use. This was certainly my experience at one site. However, it's also true that using PX will usually *lead* to a busier i/o subsystem because the server is likely to favour full scans over indexed retrieval. There are no easy answers here - you really need to carry out some analysis of overall system resource usage to identify where the bottlenecks are and adjust the configuration accordingly."