**PX In Memory, PX In Memory IMC?**

By: [Yasin Baskan](https://blogs.oracle.com/author/yasin-baskan) | Senior Principal Product Manager

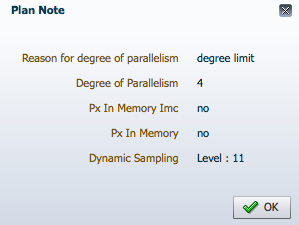
<https://blogs.oracle.com/datawarehousing/px-in-memory,-px-in-memory-imc>

In my [previous post](https://blogs.oracle.com/datawarehousing/entry/monitoring_parallel_execution_using_real) I talked about how to use SQL Monitor to monitor parallel statements. The part of that post which talks about In-Memory Parallel Execution triggered some questions and I want to provide more information on that in a separate post here.

[**In-Memory Database? In-Memory Parallel Execution? In-Memory Everywhere**](https://blogs.oracle.com/datawarehousing/px-in-memory,-px-in-memory-imc#toc_0)

Back in the 11.2 days when there was no [Oracle Database In-Memory Option](http://www.oracle.com/us/corporate/features/database-in-memory-option/index.html) (DBIM) there was no confusion about in-memory because there was only one way to use the database memory for parallel execution and that was [In-Memory Parallel Execution](https://blogs.oracle.com/datawarehousing/entry/in-memory_parallel_execution_i) (IMPX, also known as IMPQ). Starting with 12.1.0.2 DBIM boosted the performance and also the confusion about the term in-memory parallel execution.

I will not go into the details of how these two features work in this post but let us look at the plan note from the previous post and try to understand what it shows about these.

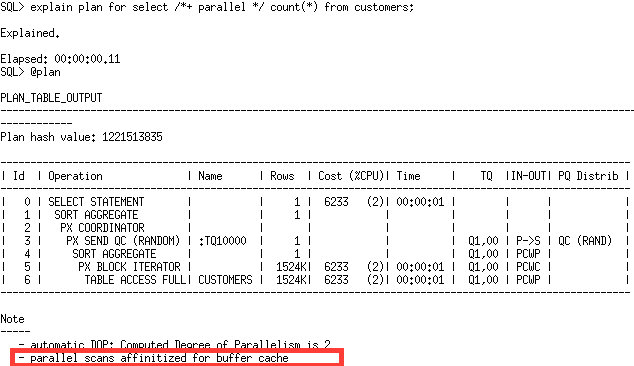


As we see there are two notes about in-memory, *Px In Memory Imc* and *Px In Memory*. Let us look at what these two mean.

[**Px In Memory**](https://blogs.oracle.com/datawarehousing/px-in-memory,-px-in-memory-imc#toc_1)

This plan note is only related to IMPX and not related to DBIM. It shows whether the statement uses IMPX or not. If you see *yes* here it means some or all of the tables in the query are accessed through the buffer cache rather than doing direct reads from disk. IMPX looks at the table sizes and the buffer cache size and decides whether to read the blocks into the buffer cache or not. Once the blocks are in the buffer cache subsequent parallel statements can read the data from there rather than going to the disk. In the case of RAC, IMPX makes sure that PX servers are allocated on the nodes where table blocks reside, as a result of this cache fusion is not used to transfer blocks between nodes which in turn makes performance better.

This plan note in SQL Monitor maps to the note *parallel scans affinitized for buffer cache* in the DBMS\_XPLAN output. Here is an example showing IMPX will be used.



The first access to the *CUSTOMERS* table will read the data from the disk to the buffer cache, subsequent accesses to this table will read data from the buffer cache. The term *affinitized* here means that specific blocks are assigned to specific nodes and PX servers on those nodes will read the data through the buffer cache.

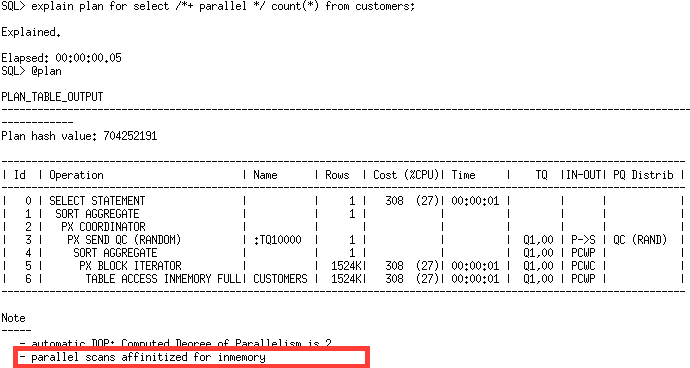
You will see this note if IMPX is used independent of whether you are on a single-instance database or a RAC database.

**So, if you see the note *parallel scans affinitized for buffer cache* in the DBMS\_XPLAN output you will see *Px In Memory=yes* in SQL Monitor.**

[**Px In Memory Imc**](https://blogs.oracle.com/datawarehousing/px-in-memory,-px-in-memory-imc#toc_2)

This plan note is only related to DBIM and not related to IMPX. IMC here means In-Memory Columnar which indicates DBIM. The IMC store provides metadata that shows which part of an in-memory table is in which node's memory in a RAC database. Parallel Execution uses this information to allocate PX servers on those nodes so that they can read the data from the IMC store without going to disk. The plan note *Px In Memory Imc* shows whether the statement accesses an in-memory table and whether PX servers are affinitized to RAC nodes. This does not indicate if DBIM is used or not, that is indicated by the operations in the plan like *TABLE ACCESS INMEMORY FULL*.

This note maps to the note *parallel scans affinitized for inmemory* in the DBMS\_XPLAN output. Here is an example showing in-memory affinity is used.



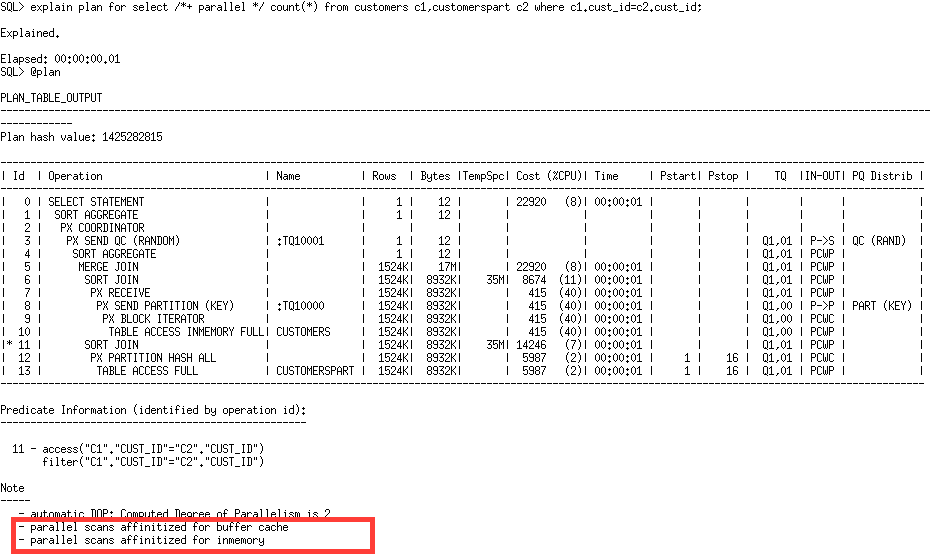
In a single-instance database you will not see this note because any PX server can read the data from the single node's IMC store and there is no concept of affinity of PX servers to nodes.

**So, if you see the note *parallel scans affinitized for inmemory* in the DBMS\_XPLAN output you will see *Px In Memory Imc=yes* in SQL Monitor.**

[**One or The Other?**](https://blogs.oracle.com/datawarehousing/px-in-memory,-px-in-memory-imc#toc_3)

It is possible to have parallel statements using both IMPX and DBIM. If your statement accesses both in-memory tables and tables determined to be read via the buffer cache you can see both notes in the plan output.

In this example, *CUSTOMERS* table is an in-memory table and since this is a RAC database PX servers are affinitized to nodes. *CUSTOMERSPART* table is not an in-memory table but Parallel Execution decided to read it through the buffer cache.



There is a lot of information about how DBIM works in the [In-Memory blog](https://blogs.oracle.com/In-Memory/), I will talk about how IMPX works in a future post. In the meantime please comment here if you have any questions related to it.