Oracle Maximum
Availability Architecture

Oracle Database In-Memory

High Availability Best Practices

ORACLE WHITE PAPER | SEPTEMBER 2015

Table of Contents

Introduction

Oracle Database In-Memory (Database In-Memory) introduces dramatically better performance for analytic queries. Because the In-Memory column store has been seamlessly integrated into Oracle Database, all of the high availability benefits that come from the Maximum Availability Architecture (MAA) are inherited when implementing Database In-Memory. With generic MAA best practices as described in the HA documentation or MAA white papers located in www.oracle.com/goto/maa, Oracle Database In-Memory can be configured to tolerate instance and node failures, protect and repair from data corruptions quickly, to fail over in the case of cluster, database failures or disasters in a timely manner and to address various planned maintenance activities with near zero database downtime. This paper focuses on additional best practices to achieve high availability and high performance application services for Oracle Database In-Memory during planned maintenance activities and failures.

Best Practices

Adherence to the following best practices will result in a highly available In-Memory implementation:

- Apply Recommended Software Optimizations for InMemory MAA
- Configure and Manage Connections for High Availability
- Duplicate In-Memory column store Across Nodes On Engineered Systems
- Configure Tables to be In-Memory prior to Partition Exchanges
- Follow Standard MAA Best Practices

The best practices in this paper assume that you are implementing these recommendations on an Oracle Real Application Cluster (RAC) database. Oracle RAC provides your application with cluster and database services needed to support high availability and is the foundation for the rest of the best practices described below.

Apply Recommended Software Optimizations for Database In-Memory MAA

Refer to My Oracle Support (MOS) 2045279.1 for the latest Database In-Memory MAA software prerequisites and recommendations.

Configure and Manage Connections for High Availability

A high availability strategy must include a way to notify client sessions when a cluster resource is no longer available or when a resource is back on-line. Oracle provides Fast Application Notification (FAN) and Fast Connection Failover (FCF) features to quickly notify clients when an instance or database fails, starts up or changes database roles. Furthermore, the use of a connection pool such as Oracle Universal Connection Pool (UCP) can provide additional features such as runtime load balancing and Application Continuity.

Oracle Database In-Memory in a RAC environment

Oracle Database In-Memory (Database In-Memory) enables data to be populated into memory in a new in-memory column format. It does this using an In-Memory column store (IM column store), which is a new component of the Oracle Database System Global Area (SGA), called the In-Memory Area. Data in the IM column store does not reside in the traditional row format used by the Oracle Database; instead it uses a new column format. Just as a tablespace on disk is made up of multiple extents, the IM column store is made up of multiple In-Memory Compression Units (IMCUs).

Only objects with the INMEMORY attribute are populated into the IM column store. A table will be automatically loaded into the IM column store when the PRIORITY sub-clause of the INMEMORY attribute is set to any value other than NONE. This will cause a table to be populated into memory immediately after the database instance starts. If the PRIORITY sub-clause is not specified it will default to NONE, meaning the object won't be populated into the IM column store until it accessed for the first time.

Each node in a RAC environment has its own In-Memory column store. By default, a table specified with the INMEMORY attribute is automatically distributed across all of the nodes in a cluster. How an object is distributed across the cluster is controlled by the DISTRIBUTE sub-clause. By default, Oracle decides the best way to distribute the object across the cluster given the type of partitioning used (if any). Ideally the data should be evenly distributed across the RAC nodes. If there is a data skew in your (sub)partitions, its recommend that you specify DISTRIBUTE BY ROWID RANGE to distribute by rowid ranges.

Since data populated in-memory in a RAC environment is affinitized to a specific RAC node, parallel server processes must be employed to execute a query on each RAC node against the piece of the object that resides in that node's IM column store. The query coordinator aggregates the results from each of the parallel server processes together before returning them to the end user's session. In order to ensure the parallel server processes are distributed appropriately across the RAC cluster, you must use Automatic Degree of Parallelism (AutoDOP), so the query coordinator is aware of the data affinity or the data location.

AutoDOP is enabled by setting the `parallel_degree_policy` parameter to AUTO. With AutoDOP the optimizer automatically determines the degree of parallelism used for a query. This setting should not be confused with the table decoration or hints that permit you to manual specify the degree of parallelism for an object or a SQL statement.

If a RAC node were to fail, a portion of the data will no longer be in-memory and will have to be read from the buffer cache, flash or disk. This can result in an increase in response times until the missing portion of the data can be populated into the In-Memory column stores on the remaining nodes.

When running on an Oracle Engineered System such as Exadata or SuperCluster and high availability with consistent response time expectations is paramount, tables should be mirrored in-memory across nodes to ensure that all of the data needed by an application is already stored in-memory and doesn't need to be re-populated when a node goes down, leading to performance impact until data is repopulated in the surviving instances.

To do this, simply specify the *DUPLICATE* sub-clause of the INMEMORY attribute as follows:

```
SQL> ALTER TABLE customers INMEMORY PRIORITY NONE DUPLICATE ;
```

This will ensure that a mirrored copy of each piece of the table (IMCU) will be stored in-memory on another node in the cluster. Note, PRIORITY NONE was specified in the above command using DUPLICATE to ensure that tables are only populated on demand via a triggering query and repopulation occurs on the desired node.

To protect against multiple instance failures, it is possible to populate an object into the IM column store on each node in the cluster by specifying the DUPLICATE ALL sub-clause of the INMEMORY attribute. This will provide the highest level of redundancy and provide linear scalability, as queries will be able to execute completely within a single node.

```
SQL> ALTER TABLE customers INMEMORY DUPLICATE ALL;
```

This will ensure the table is duplicated across all nodes in a cluster and any DML made to the table will be simultaneously reflected in the IM column store on all nodes. Since all IMCUs are available on all nodes, AutoDOP is no longer required to ensure affinity between the IMCUs and parallel slaves, so `parallel_degree_policy` can left at the default MANUAL. The DUPLICATE ALL subclause also allows manual control of the degree of parallelism via hint or table property (including allowing the optimizer to decide) knowing that parallel slaves will always find an IMCU on the instance they are running against.

Please note that the use of the DUPLICATE or DUPLICATE ALL sub clause on a non-Engineered system will not have any effect.

Best Practices for Using DUPLICATE and DUPLICATE ALL

The best practices for using the DUPLICATE and DUPLICATE ALL options involve the use of FAN Managed Services and proper settings for session parameters.

FAN Managed Services

FAN enabled clients and properly configured services are a key factor for minimizing impact to applications when planned or unplanned outages occur. When using DUPLICATE or DUPLICATE ALL it is recommended to avoid application impact ("brownouts") during repopulation by preventing client connections to the instance that has been restarted and will repopulate data. This is done by disabling and stopping the service for an instance that has gone down. We refer to this as "managed" connection service in this discussion and Table 1 below. The steps needed to manage a service are:

1. Use SRVCTL to stop and disable the services on the failed instance for analytic clients using In-Memory column store, e.g.:

   ```
   $ srvctl stop service –service prod_dbim_fan –database dbm –instance dbm1
   $ srvctl disable service -service prod_dbim_fan –database dbm –instance dbm1
   ```

2. Perform maintenance on the instance or node

3. Ensure the IM column store is being populated on the newly restarted instance and wait for all tables to be populated. Repopulation can be triggered by performing a full table scan against tables you marked with the In-Memory attribute. We refer to this query as a triggering query because it triggers the repopulation of the IM column store. The session running the triggering query must have its *parallel_instance_group* set properly (see Table 1 and *Session Parameter Settings* below).

You can monitor how many bytes of a table or partition have yet to be populated across instances with this query:

```
SELECT v.inst_id, v.segment_name name, v.partition_name
, v.populate_status status
, v.bytes/1024/1024 bytes_mb
, v.bytes_not_populated/1024/1024 bytes_not_pop_mb
, (v.bytes-v.bytes_not_populated)/1024/1024 bytes_populated_mb
FROM gv$im_segments v
WHERE v.owner = '&owner_name'
ORDER BY 1
```

When using DUPLICATE ALL, the column "`bytes_not_populated`" will be zero when the segment is completely In-Memory If using the DUPLICATE sub-clause, use the following query:

```
SELECT v.segment_name name, v.partition_name,
2*max(v.bytes)/1024/1024 tot_mirrored_bytes_mb
, sum(v.bytes-v.bytes_not_populated)/1024/1024 tot_bytes_populated_mb
FROM gv$im_segments v
WHERE v.owner = '&owner_name'
GROUP BY  v.segment_name, v.partition_name
ORDER BY 1
```

When the segments are entirely in-memory, you will see the TOT_MIRRORED_BYTES_MB column equal the TOT_BYTES_POPULATED_MB column.

4. Use SRVCTL to enable and start the services using In-Memory on the newly restarted instance

   ```
   $ srvctl enable service -servcce imquery –database dbm –instance dbm1
   $ srvctl start service –service imquery –database dbm –instance dbm1
   ```

5. Clients will begin using the new instances as their connection pools load balance or new connections are made

Some of the above steps may be automated with scripts for convenience and efficiency. MOS note 1927000.1 shows how to script a FAN callout in Oracle RAC environments to automatically disable a service when an instance or node goes down.

Session Parameter Settings

The DISTRIBUTE sub-clause affinitizes the data populated into the IM column store to a specific node, so *parallel_degree_policy = AUTO* must be set by sessions using parallel execution to ensure processes are sent to the appropriate node containing the data in-memory. This is true even if the DUPLICATE sub clause is specified. If the use of *parallel_degree_policy=AUTO* is acceptable to the application, then Oracle recommends the use of DUPLICATE because it reduces memory usage while still offering high availability.

The DUPLICATE ALL option populates complete copies of the data on each instance, so a session may have *parallel_degree_policy* set to MANUAL, LIMITED, or AUTO and still benefit from the In-Memory column store since the parallel execution processes will always find the data they need regardless of which node the parallel execution processes are running on.

The *parallel_instance_group* parameter is used to control which instances participate in parallel execution. *The* service assigned to the *parallel_instance_group* parameter and used by the application's sessions may be different

than the one used for connections (the service referred to as a *managed connection service* as introduced in the previous section). This is done to maintain proper affinity of parallel execution processes during repopulation. See the table for specific recommendations on when to use different services for the connection and *parallel_instance_group* parameter.

TABLE 1: CONFIGURATION SETTINGS FOR DUPLICATE / DUPLICATE ALL

Duplicate mode	Parallel degree policy	Table or Query DOP setting	Parallel instance group service used by the application	Parallel instance group service used during repopulation	Benefits
DUPLICATE	AUTO	AUTO*	Including all instances and different from connection service	Including all instances and different from the connection service	» Negligible impact for single instance failure » Maximizes IM capacity
DUPLICATE	LIMITED or MANUAL	NOT RECOMMENDED – NO PQ AFFINITIZATION POSSIBLE			
DUPLICATE ALL	AUTO, LIMITED, or MANUAL	AUTO or MANUAL	Same as the "managed" connection service	Including all instances and different from the connection service	» Tolerates multiple instance failures » Allows use of parallel_degree_policy = LIMITED or MANUAL

**When the parameter parallel_degree_policy is set to AUTO no table or query DOP should be set. The Optimizer should have full control over the DOP used.*

A couple of examples will illustrate the recommendations in the table above (both examples assume the use of RAC on a Full-Rack Exadata Engineered System):

1. Using DUPLICATE on an Exadata Full-Rack System

» Managed connection service uses `prod_dbim_fan` service

- Application clients connect with this service; the service is altered during outages by a custom script (as discussed earlier) to ensure clients don't connect to an instance that went down. The service is altered when the instance is available again and repopulation on the restarted instance is finished.
- The service is configured to send connections to any of the eight database nodes unless there is an outage. Immediately after an outage, the managed connection service disables and stops the service for the failed instance. In-flight queries may be affected, but new connections will use the remaining instances enabled for the service.
- Two copies of IMCU are present across the eight instances since tables placed in the IM column store are set to use the *DUPLICATE* option. This permits better use of memory for In-Memory objects.

» Repopulation service uses *prod_dbim* service

- This is defined to include all eight instances in the cluster; they all participate in parallel queries; this service is not changed during outages.
- To repopulate the IM column store, a session will have the `parallel_instance_group` value of *prod_dbim* when executing the triggering query on the newly restarted instance once it has become available. This will cause the IM column store to be repopulated on the restarted instance such that it contains the same IMCUs it had before the crash (adjusting for changes that may have occurred since).

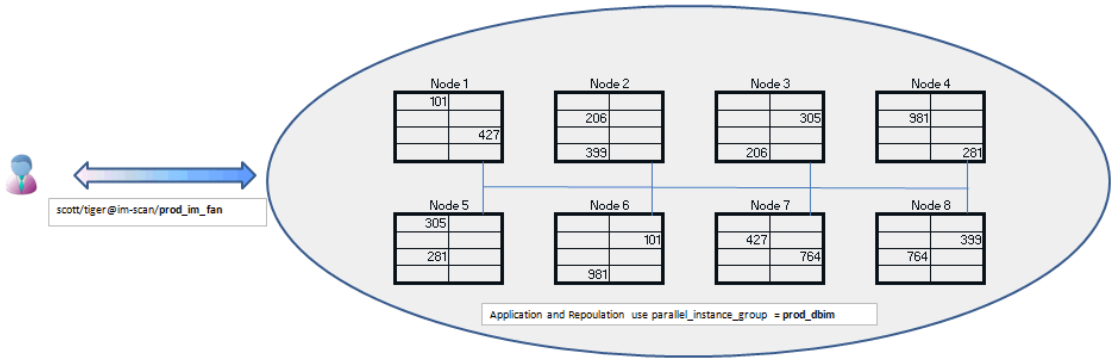
» The application uses the following for its database sessions:

- `parallel_degree_policy`= `AUTO` (required with DUPLICATE to affinitize slaves to IMCUs)
- `parallel_instance_group` = `prod_dbim` (differs from service used by clients to connect)

» The managed connection service will be enabled and started for the restarted instance once its IM column store is repopulated.

» Connections will resume on the newly started instance once the connection service is enabled and started.

The illustration below uses numbers to represent IMCUs. IMCUs with the same number represent copies of each other. Notice that there is a primary and secondary copy of each IMCU in the cluster but Oracle can read from either side of the mirror at any time.



This shows clients connecting with the "prod_im_fan" service that is managed so that a failed instance is disabled from the service.

The application uses the "prod_dbim" service for parallel_instance_group parameter.

A session executing the triggering query to repopulate a failed instance's IM column store will also use the "prod_dbim" service.

The application MUST set parallel_degree_policy=AUTO to ensure parallel execution processes are affinitized to instances properly.

2. Using DUPLICATE ALL on an Exadata Full-Rack System

» Each node will contain a copy of all IMCUs since In-Memory tables are set to use the *DUPLICATE ALL* option. This means that more memory is used but the cluster will tolerate the loss of more than one node without performance impact to the application. It will also permit the application to use *parallel_degree_policy* set to `MANUAL`, instead of `AUTO`, if this is needed.

» Managed connection service uses *prod_dbim_fan* service

- Application clients connect with this service; the service is altered during outages to ensure clients don't connect to an instance that went down until after repopulation is finished as in the previous example. After an instance failure, the managed connection service disables and stops the service from running on the failed instance. In-flight queries may be affected, but new connections will use the remaining instances that have a mirror copy of the failed instance's IMCUs.
- The application uses `parallel_instance_group` set to the managed connection service so parallel execution processes will not be spawned on the newly restarted instance during repopulation.

» Repopulation service uses *`prod_dbim`* service

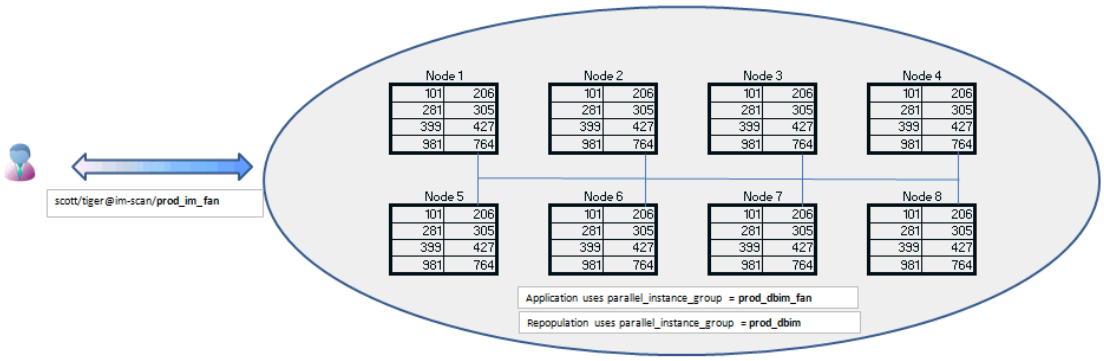
- This contains all instances desired to participate in parallel queries; this service is not changed during outages.To repopulate the IM column store, a session with the *`parallel_instance_group`* value of *`prod_dbim`* executes a triggering query on the newly restarted instance that is now available again. This service includes all instances including the one that needs to be repopulated. The application continues to use the managed connection service for its `parallel_instance_group` *(*`prod_dbim_fan`*)* to ensure parallel execution processes are affinitized to IMCUs on available instances during repopulation.

» *The a*pplication uses the following for its database sessions:

- *`parallel_degree_policy`* = AUTO or MANUAL
- Query or table DOP setting can be AUTO or MANUAL
- *`parallel_instance_group`* = *`prod_dbim_fan`* (notice this is the same as the connection service)

The managed connection service will be enabled and started for the newly restarted instance once its IM column store is repopulated. Connections will resume on the newly started instance.

The illustration below uses numbers to represent IMCUs. IMCUs with the same number represent copies of each other. Notice each node has a complete set of IMCUs.



This shows clients connecting with the "prod_im_fan" service that is managed so that a failed instance is disabled from the service.

The application uses the "prod_dbim_fan" service for the parallel_instance_group parameter to ensure parallel execution processes don't try to run on an instance that is repopulating.

The application may set parallel_degree_policy=AUTO, MANUAL, or LIMITED

A session executing the triggering query to repopulate a failed instance's IM column store will use the "prod_dbim" service

If you aren't using Engineered Systems you will not benefit from the In-Memory high availability features provided by DUPLICATE or DUPLICATE ALL but you can obtain performance and capacity benefits using parallel query when you set *parallel_degree_policy* to AUTO on a RAC system.

Configure Tables for In-Memory Storage prior to Partition Exchanges

Database In-Memory works at the partition level, not just at the table level. When performing partition exchanges, it is recommended to first specify the In-Memory attribute on the non-partitioned table to be exchanged into the partitioned table and ensure it is populated into the IM column store. When the partition is exchanged, the data will already be in in-memory and usable without waiting for its data to be populated. You can exchange a partition in a table populated into the IM column store at any time without needing to quiesce the workload just as you have been able to do with a non-In-Memory table in the past. In the example command below, you would first alter the standalone table, lineorder_tab, to have its in-memory attributes set, ensure it is populated in-memory, and then use the following command to do the partition exchange operation:

```
ALTER TABLE lineorder EXCHANGE PARTITION p1 WITH TABLE lineorder_tab;
```

Follow Standard MAA Best Practices

The best practices listed above are to be applied in addition to the MAA best practices that are already published in the High Availability Best Practice documentation and other MAA papers found on OTN at www.oracle.com/goto/maa.

MAA Outage Testing Using In-Memory Tables

Outage Testing Matrix

The following matrix lists various outage scenarios that were tested for clients using In-Memory tables and the observed impacts to the client application.

MAA OUTAGE MATRIX WITH IN-MEMORY TABLES

Outage	Outage Simulation Process	Application Impact and Observations
Planned DB Node Maintenance	1. Start workload with all instances having pre-loaded the In-Memory column store 2. Stop service normally on an instance 3. Disable service for the affected instance 4. Ensure all connections moved to instances on other nodes 5. Shutdown instance or node 6. Perform maintenance on node 7. Restart instance or node 8. Run query to load In-Memory column store for restarted	• Minimal impacts to the application in Oracle RAC during reconfiguration (if the maintenance calls for the instance to be stopped and restarted)

	instance 9. Wait for all tables to load In-Memory on affected instance 10. Enable service for affected instance 11. Start service for affect instance 12. Observe clients using the restarted instance	
Stop Service FORCE	1. Ensure the application is configured for Application Continuity and application is able to trap and retry after ORA-12805 or ORA-40 errors 2. Start workload 3. Stop service FORCE on an instance 4. Start service on again on the instance 5. Observe load returns to the restarted service over time	• Brief response time spike at time of failure due to: • Termination of query • Re-run / replay of query when application continuity is used • No need to re-load the In-Memory column store since the instance did not fail or get restarted

Database Instance Failure	1. Ensure application is able to trap and retry ORA-12805 or ORA-40 after errors
	2. Start workload
	3. Kill SMON on one instance
	4. Immediately disable the service for the affected instance
	5. Wait for instance to restart
	6. Run queries to force In-Memory tables to load – wait until tables are loaded
	7. Enable the service back on the affected instance
	8. Observe load returns to the restarted instance over time
		• Brief response time spike at time of failure due to:
		  • Termination of query
		  • Oracle RAC reconfiguration
		  • Re-run / replay of query when application continuity is used
		• Brief response time spike when the failed instance rejoins the cluster

Database Node Failure	1. Ensure application is able to trap and retry ORA-12805 or ORA-40 after errors
	2. Start workload
	3. Remove power from the database node
	4. Immediately disable the service for the affected instance
	5. Power on and wait for instance to be restarted
	6. Run queries to force In-Memory tables to load – wait until tables are loaded
	7. Enable the service back on the affected instance
	8. Observe load returns to the restarted instance over time
		• Brief response time spike at time of failure due to:
		  • Termination of query
		  • Oracle RAC MISCOUNT delay (1 min typically); This wait is reduced to a few seconds on Engineered Systems at 12.1.0.2 BP7 or higher
		  • Oracle RAC reconfiguration
		  • Re-run / replay of query when application continuity is used
		• Brief response time spike when the failed instance rejoins the cluster

Database Failure
Data Guard Failover
	1. Start workload
	2. Ensure service is disabled on the standby database (and enabled on the primary)
	3. Shutdown abort ALL nodes in the cluster
	4. Wait for standby database to open
	5. Run queries to force In-Memory tables to load – wait until tables are loaded
	6. Enable the service on the new primary database
	7. Observe load is occurring on the new primary database
		• Response time spike at time of failure due to:
		  • Termination of query
		  • Standby Open
		  • Load of In-Memory tables in cache
		  • Re-run / replay of query when application continuity is used
		Note:The business and the administrator have to weigh the trade-offs between very fast database and application recovery time objective (RTO) when you allow clients and applications to connect immediately after the database starts up as primary VS longer downtime or RTO when waiting for the in-Memory tables to be pre-loaded before allowing application to connect. The latter ensures the fast in-memory query response immediately after connecting to the new primary database.

Please note the following regarding the above tests:

- The test application was configured to use the UCP connection pool in 12.1.0.2
- The remaining nodes had sufficient CPU and memory capacity to take on the load from the instance that was shut down for maintenance. If this is not the case in your environment, you will need to leverage Oracle's Resource Management features and accept an impact to your performance SLAs.

Conclusion

Oracle Database In-Memory provides game-changing performance improvements while retaining the reliability and high availability benefits of the Oracle Database. Implementing the best practices found in this paper will help you maximize availability when implementing Database In-Memory.

References

Oracle Technology Network: Oracle Database In-Memory Page: http://www.oracle.com/us/products/database/options/database-in-memory/overview/index.html

Oracle Database In-Memory Blog, https://blogs.oracle.com/In-Memory/tags/rac

Shell Script to Automatically Disable a Service Based on FAN Events, http://support.oracle.com, MOS note ID 1927000.1

Graceful Application Switchover in RAC with No Application Interruption, http://support.oracle.com, MOS note ID 1593712.1

Client and Application Failover Validation Matrix, http://support.oracle.com, MOS note ID 1617163.1

Client Failover Best Practices for Highly Available Oracle Databases - Oracle Database 12c, http://www.oracle.com/technetwork/database/features/availability/oracle-database-maa-best-practices-155386.html

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

ORACLE

Connect with us

blogs.oracle.com/oracle

facebook.com/oracle

twitter.com/oracle

oracle.com

Integrated Cloud Applications & Platform Services

Oracle Database In-Memory: High Availability Best Practices
September 2015
Author: Hector Pujol
Contributing Authors: Maria Colgan; Richard Exley; Andy Rivenes; Lawrence To