An Oracle White Paper
February 2014

Analysis of SAP HANA High Availability Capabilities

ORACLE

Introduction

In today's business world, High Availability (HA) is an extremely important consideration for any enterprise-level IT architecture. A technology platform missing on critical HA features is not enterprise-ready.

HANA has been touted by SAP as the modern platform for real-time analytics and applications. The SAP HANA documentation [1] even states that "*SAP HANA is fully designed for high availability*.". However – as the analysis in this whitepaper shows, this assertion is not correct!

This whitepaper provides an in-depth analysis of the HA features available with SAP HANA Support Package Stack (SPS) 07 [2]. It also provides a comparison between these SAP HANA HA features and HA features available with the Oracle Database, as part of Oracle Maximum Availability Architecture (MAA) [3].

As this white paper demonstrates – as far as HA is concerned, there are serious drawbacks in HANA's technical architecture, and HANA lacks several key capabilities that are absolutely necessary to implement a robust HA architecture. SAP HANA is not enterprise-ready.

The intended audience of this whitepaper are members of IT managerial and technical teams interested in finding out whether SAP HANA is viable for any of their critical applications, and/or who want to get a better understanding of how the HANA architecture compares with well-established high availability technologies.

SAP HANA – An Architectural Overview

SAP uses the term "SAP HANA" to designate the SAP In-Memory Appliance which is aimed at data analytics. It is a combination of hardware and software, and it is delivered as an optimized appliance in cooperation with SAP's hardware partners for SAP HANA. SAP also uses the term to indicate the SAP in-memory database, which is a hybrid in-memory database that combines row-based, column-based, and object-based database technology. The following diagram provides a high-level overview of the SAP HANA architecture.

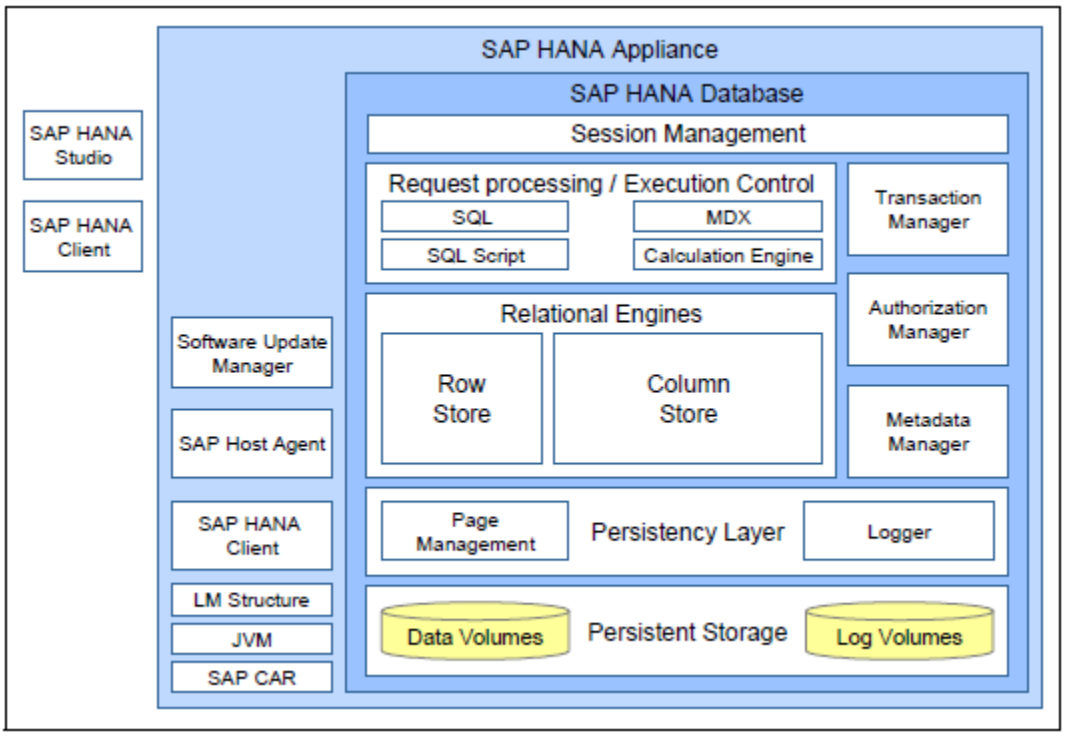

Fig: 1: SAP HANA Architecture [5]

SAP HANA Database

The SAP HANA database consists of two database engines:

- The column-based store, storing relational data in columns, optimized for holding data mart tables with large amounts of data, which are aggregated and used in analytical operations.
- The row-based store, storing relational data in rows. This row store is optimized for write operations and has a lower compression rate, and its query performance is much lower compared to the column-based store.

The engine that is used to store data can be selected on a per-table basis at the time of creation of a table (and changed subsequently). Tables in the row-store are loaded into memory at startup time, whereas tables in the column-store can be either loaded at startup or on demand, during normal operation of the SAP HANA database.

Both engines share a common persistency layer [6], which provides data persistency across both engines. Changes to in-memory database pages (equivalent to Oracle blocks) are made persistent through savepoints written to the data volumes on persistent storage. This is done automatically, at least every 5 minutes, and this is customizable. Besides, every transaction committed in the SAP HANA database is persisted by the logger of the persistency layer in a log entry written synchronously to the log volumes (similar to Oracle redo logs) on persistent storage.

The SAP HANA database supports SQL (JDBC/ODBC), MDX (ODBO), and BICS (SQL DBC). BICS is a SAP HANA-specific SQL Script language that is an extension to SQL that can be used to push down data-intensive application logic into the SAP HANA database (similar to Oracle stored procedures).

SAP HANA Appliance

The SAP HANA appliance consists of the SAP HANA database, as described above, and adds components needed to work with, administer, and operate the database. It contains the installation files for the SAP HANA Studio, which is an Eclipse-based administration and data-modeling tool for SAP HANA, and the SAP HANA client, a set of libraries required for applications to be able to connect to the SAP HANA database. The Software Update Manager (SUM) for SAP HANA is the framework allowing the automatic download and installation of SAP HANA updates from SAP Marketplace and other sources using a host agent.

SAP partners with several hardware vendors such as Cisco, Dell, IBM, HP, and Fujitsu, to provide the infrastructure needed to run the SAP HANA software in an Intel Xeon-based appliance model.

Analysis: SAP HANA High Availability (HA) Features

The HA analysis of SAP HANA can be done across the following three elements.

1. Scalability support
2. Disaster Recovery (DR) support
3. Backup & Recovery support

The following sections will look into each of these in greater detail.

SAP HANA: Analysis of Scalability Features

HANA implements scalability through the use of multiple servers in one SAP HANA cluster [7], using a shared nothing approach for the underlying data, which has to be partitioned across these servers (hosts). SAP often refers to this layout as the distributed SAP HANA database / system. This cluster typically hosts N active servers with M standby servers, accessing a shared file system. Each server is either 4 CPU/512GB or 8 CPU/1TB, and the largest certified configuration is a cluster of 56 servers [8].

Distributed HANA System: Name, Index and Statistics Servers

The Index / Name / Statistics system components are integral to a distributed HANA system and deserve special mention. In a single-host system, all these components are installed on a single SAP HANA database instance on one host. In a distributed SAP HANA system, they are installed on multiple database instances on different hosts.

- Name server – The name server owns the information about the topology of the SAP HANA system. In a distributed system with instances of the SAP HANA database on multiple hosts, the name server knows where the components are running and which data is located on which server.
- Index server – The index server contains the actual data stores and the engines for processing the data.
- Statistics server – This statistics server collects information about status, performance and resource consumption from all components belonging to the SAP HANA system. Monitoring clients such as the SAP HANA studio access the statistics server to get the status of various alert monitors. The statistics server also provides a history of measurement data for further analysis.

The Index Server is the core database engine – following is the architecture diagram for the Index Server:

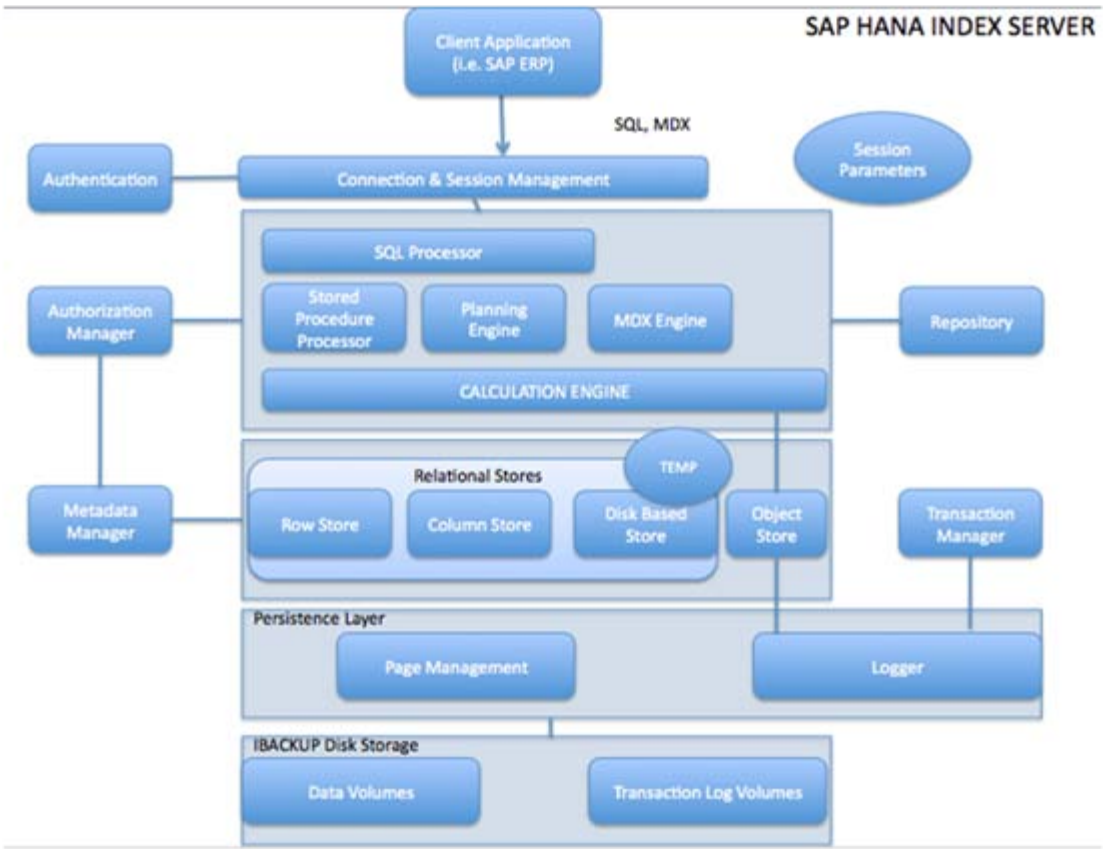

Fig: 2: SAP HANA Index Server Architecture [9]

Setting up a distributed HANA system involves a complex configuration of Name Server and Index Server roles for the hosts implementing the distributed system. It works as follows [10].

- Name Server

  Up to 3 hosts can be configured as MASTER Name Servers (using a "MASTER 1, MASTER 2, MASTER 3" naming designation), while others have to be configured as "SLAVE". However, in steady state, only one of the "MASTER n" configured hosts has its actual Name Server role as "MASTER" while all other hosts have their actual Name Server roles as "SLAVE". The MASTER Name Server is responsible for assigning storage/data volumes to active Index Servers at the time they start up. [*Note: From SAP Collateral, the functionality provided by the SLAVE Name Server is not clear.*]

  If the MASTER Name Server host fails, one of the remaining hosts configured as a "MASTER n" Name Server becomes the active MASTER Name Server.

- Index Server

  During setup, hosts can be configured as "WORKER" or "STANDBY" Index Servers. In steady state, one WORKER host gets the actual Index Server role as "MASTER" – this is assigned on the same host as the MASTER Name Server. Other WORKER hosts get the actual Index Server role of "SLAVE". The STANDBY Index Server gets the actual Index Server role of "STANDBY". The MASTER Index Server provides metadata for the other SLAVE Index Servers. The SLAVE Index Server is an active database server, is assigned a data volume, and accepts database connections. A host configured as a STANDBY Index Server is not used for database processing – all database processes run on this host, but they are idle and do not allow SQL connections.

  If an active Index Server fails, the active MASTER Name Server assigns its volume to one of the hosts in the STANDBY Index Server role.

Storage Layout: Single-Instance HANA System

To better understand how a distributed HANA system with multiple hosts persists data and accesses storage, it helps to see what the storage layout of a single-node HANA system looks like. The following diagram shows this layout from an IBM systems/storage standpoint.

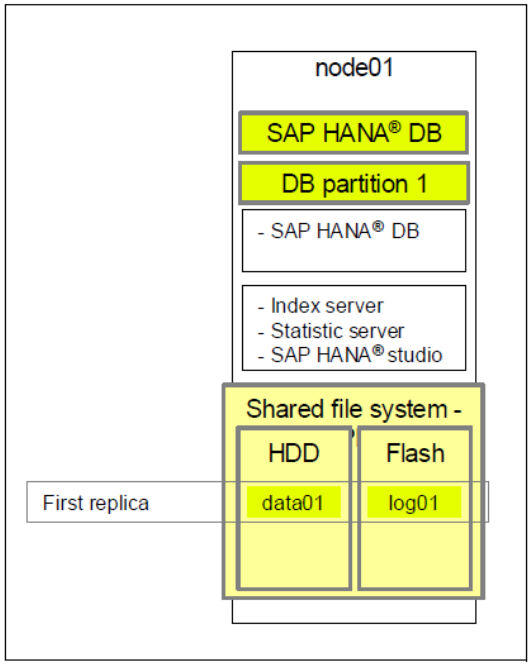

Fig: 3: Storage Layout: Single Node SAP HANA System [5]

In the above diagram, IBM's General Parallel File System (GPFS), which is a shared-disk clustered file system providing concurrent file access to applications executing on multiple nodes of clusters, is implemented on two types of storage:

- The data storage (on SAS disks), here referred to as HDD, holds the savepoints.
- The log storage (on SSD drives or PCIe Flash devices), here referred to as Flash, holds the database logs.

This single node represents one single SAP HANA database consisting of one single database partition. Both the savepoints (data01) and the logs (log01) are stored once (that is, they are not replicated), denoted as "first replica" in the above diagram.

Storage Layout: Distributed HANA System

The scale-out distributed HANA system differs from a single server solution in a number of ways:

- The solution consists of a cluster of building blocks, interconnected with two separate 10 Gb Ethernet networks, one for the SAP HANA application and one for the file system communication.
- The SAP HANA database is split into partitions, forming a single instance of the SAP HANA database.
- Each node of the cluster holds its own savepoints and database logs on the local storage devices of the server.
- The standby nodes run the SAP HANA application, but do not hold any data or take an active part in the processing. If one of the active nodes fails, a standby node will take over the role of the failed node, including the data (that is, the database partition) of the failed node. This takeover process is automatic, but orchestrated through the MASTER Name Server.
- The GPFS file system spans all nodes of the cluster, making the data of each node available to all other nodes of the cluster, including the standby node.

The following diagram illustrates this solution, showing a 4-node configuration as an example, in which the node04 serves as the standby.

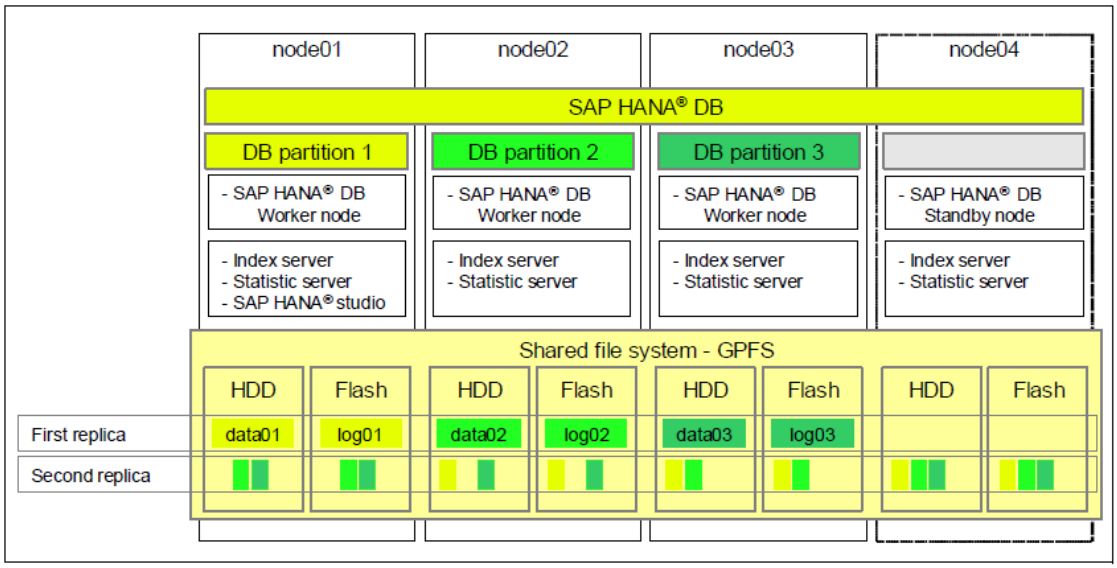

Fig: 4: Storage Layout: Distributed SAP HANA System [5]

The SAP HANA software distributes application requests internally across the cluster to the individual worker nodes, which process the data and exchange intermediate results, which are then combined and sent back to the requestor. Each node maintains its own set of data, persisting it with savepoints and logging data changes to its database log.

With respect to the system components described previously, only the Name Server service on the active MASTER host persists data. SLAVE Name Sever hosts communicate with the MASTER, but do not persist data. The Index Server service on all hosts except STANDBY hosts persists data. The Statistics Server service can run only on one host and persists data on this host.

A distributed filesystem such as GPFS combines the storage devices of the individual nodes into one file system, making sure that the SAP HANA software has access to all data regardless of its location in the cluster, while making sure that savepoints and database logs of an individual database partition are stored on the appropriate locally attached storage device (disk and flash) of the node on which the partition is located.

Distributed HANA System: Node Failover

To be able to take over the database partition from a failed node, the standby node has to load the savepoints and database logs of the failed node to recover the database partition and resume operation in place of the failed node. Using the previous diagram as an example, assume node03 experiences a problem and fails. The master node (node01) recognizes this and directs the standby node, node04, to take over from the failed node. To recreate database partition 3 in memory to be able to take over the role of node03 within the cluster, node04 reads the savepoints and database logs of node03 from the GPFS file system, reconstructs the savepoint data in memory, and re-applies the logs so that the partition data in memory is exactly like it was before node03 failed. After this is complete, Node04 is in operation as an active Index Server, and the database cluster has recovered. Data access in non-failed nodes can continue, although the extent of brown-out is not clear from HANA collateral.

After fixing the cause for the failure of node03, it can be reintegrated into the cluster as the new standby system.

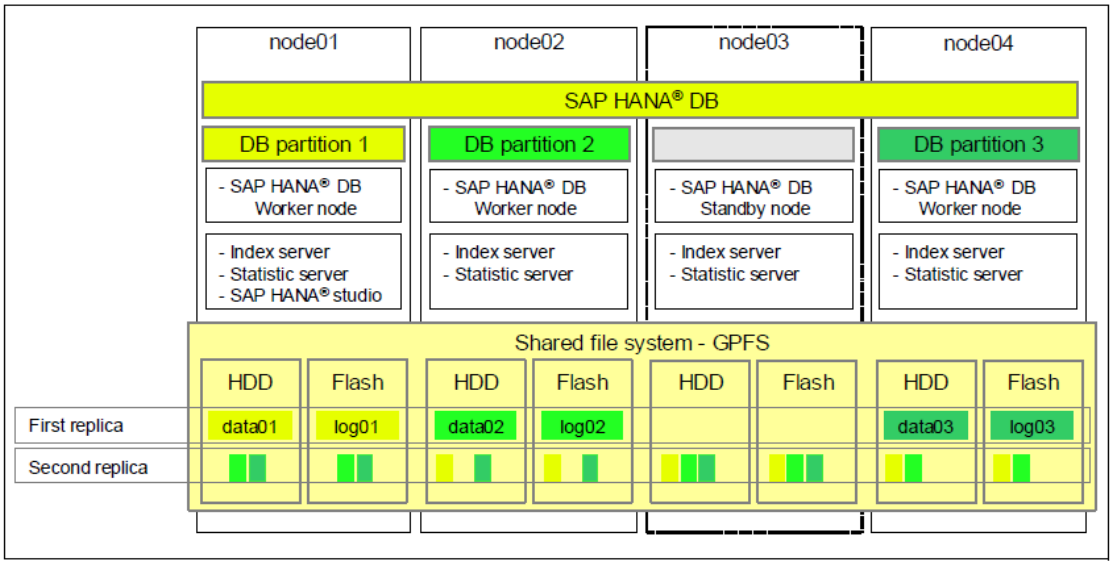

Fig: 5: Failover: Distributed SAP HANA System [5]

When a node has an unrecoverable hardware error, the storage devices holding the node's data might become unavailable or even destroyed. To mitigate this, the GPFS file system can be configured to replicate the data of each node to the other nodes, to prevent data loss if one of the nodes (with its local storage) goes down. The replication is done by GPFS synchronously, i.e. each write operation only finishes when the data has been both written locally and replicated.

HA Analysis: Analyzing the HANA Clustering Technology

As evident from preceding sections, HA support for a distributed HANA system is indeed very complex, which is largely a result of SAP's adopting a shared nothing approach for the underlying data. Note that the underlying storage system holding the data is shared among the multiple hosts within the HANA cluster. However – in steady state, each node has exclusive access to its data partition. This leads to the complexities, summarized as follows.

Insight #1: SAP HANA – NO HA
Inferior Clustering Technology

1. **Complex**

   The system model of a SAP HANA cluster, comprised of Name, Index and Statistics Server processes, is very complex – much more so than Oracle Real Application Clusters (RAC). This complexity is largely because of the data partitioning required to make this shared nothing approach work. This complexity is exacerbated because of the multiple configuration and actual roles that each of these nodes can assume for these systems components – e.g. MASTER, SLAVE, WORKER, STANDBY, each with a different connotation. For a large HANA cluster – if there is one, this could lead to a manageability nightmare.

2. **Not Active-Active**

   The HANA cluster is not an active-active cluster, unlike Oracle RAC. It resembles legacy cold cluster technology. Classic cold cluster technology can be implemented in N+1 topology, with N active nodes and 1 node waiting as the standby. This is worse. The reason is that because of data partitioning, for best HA the HANA cluster has to be implemented in N+M topology, with N active nodes and multiple – i.e. M inactive/standby nodes (corresponding to the M data partitions) waiting for the fault to happen and wasting precious system resources.

3. **Poor RTO, with an Inferior Recovery Technology**

   HANA uses a primitive, multi-decade-old recovery architecture which leads to very high MTTRs that are simply unacceptable in today's mission-critical environments. After a HANA node failure, one of the standby nodes assumes the role of an active Index Server. This node has to read the savepoints and database logs of the failed node from the shared file system, reconstruct the savepoint data in memory, and then re-apply the logs so that the partition data in memory is exactly like it was before the other node failed. Only then the node is ready to accept application connections. This is very time-consuming, and during this entire process the corresponding data partition is offline, leading to serious application downtime. RAC has none of these problems: if a node fails, applications simply reconnect to any of the surviving nodes and continue processing.

4. **Inefficient, with a Shared-Nothing Model Unsuitable for OLTP**

   HANA suffers from the classic deficiencies of a shared-nothing partitioned model. Unless the application itself is partition aware (which would be a poor app design anyway), to make this data partitioning work, the data access has to be a multi-hop process, in which the MASTER Name Server needs to do a lookup and forward incoming connections to the right Index Server hosting the corresponding data partition. This increases data access latency, and also leads to poor load balancing, especially when a local data set is accessed heavily. This makes HANA a poor choice for most OLTP workloads, adding prohibitive workload for short transactions. In contrast, for RAC, application requests can be forwarded to any node running that service, and data access can be efficiently load-balanced both from a connect-time and run-time standpoint.

5. **Restrictive**

   The HA topology of the HANA cluster is severely restrictive. According to SAP HANA documentation [1], "Note: Host roles for failover are normally configured during installation. … You can only switch the configured roles of hosts; you cannot increase or decrease the number of worker hosts and standby hosts in relation to each other." This is a serious limitation. It means that if a 5 node system is installed with 3 Worker Nodes (N1, N2, N3) and 2 Standby Nodes (N4, N5), the roles of N3 and N4 can be switched (for example), but N4 now can't be made to be a Worker Node. Such operational restrictions point to the inherent design flaws of the HANA architecture.

Adding / Removing Hosts

Hosts in the HANA cluster can be added through a network configuration that involves public network (for apps to communicate with the host) and private network (for hosts to communicate with each other). However, adding hosts to the HANA cluster has a serious impact to the overall availability of the cluster. According to SAP documentation [1]:

> *To ensure that you can recover your system to a point in time after you added the host, we recommend that you stop productive operations until you have added the host and performed a new, complete data backup.*

This implies that the entire HANA Database has to be quiesced before hosts can be added to the HANA cluster, which demonstrates that HANA scale-out comes with a significant downtime cost.

Insight #2: SAP HANA – NO HA
Downtime During Scale-Out

SAP touts HANA's scale-out capability through the use of multiple hosts for load-balancing in the HANA cluster. However, before adding nodes to support this scale-out, the entire database has to be quiesced and a full backup has to be taken, a serious impact to application availability. This is clearly an inferior solution to Oracle Real Application Clusters (RAC), which has none of these drawbacks.

Removing hosts has a similar pre-requisite [1]:

> *After you remove a host from your system, you must perform a data backup to ensure that you can recover the database to a point in time after you removed the host.*

However, this is a more complex operation. For example, the SAP Documentation [1] states the following:

> *The primary reason for changing the configured roles in the Configure Hosts for Failover Situation dialog box is to prepare for the removal of a host. In this case, you would change the configured role of the name server host to SLAVE and the configured role of the index server host to STANDBY before stopping the database instance on the host and removing the host.*

Besides, the tables on the Index Server of this host must be moved to the Index Servers on the remaining hosts in the system. Only then the host can be removed.

That brings us to the complex issue of Data Re-Distribution / Data Re-Partitioning, which we discuss next.

Data [Re]Distribution in a Distributed HANA System

Because of SAP HANA's shared nothing model, the underlying data must be partitioned / distributed among the various Index Server nodes of the HANA cluster.

Partitioning can be done primarily by two ways:

1. Different tables can be assigned to different index servers, which run on different hosts (database partitioning).
2. A table can be split in a way that different rows of the table are stored on different index servers (table partitioning).

With respect to table partitioning, the various methods available are hash partitioning, partitioning by range or value, or round-robin. Partitioning a table based on combinations of these is also supported.

Table [re]distribution is a related topic. When a table is created in a distributed HANA system, it must be assigned to one index server. This feature is called table distribution. By default, new non-partitioned tables are distributed across available index servers using a round-robin approach. For example, if there are three available index servers A, B, and C (including the master), the first table created will be located on server A, the next one on server B, the next on server C, and so on. It is also possible to specify explicitly that a table or a partition be created on a specific index server.

While tables / table partitions are assigned (partitioned) to specific Index Servers on a particular host at their time of creation, this assignment needs to be changed under various scenarios, such as adding / removing hosts, or when current table distribution / partitioning is no longer optimal. This feature is called table redistribution.

SAP itself acknowledges that table distribution / partitioning can become inefficient with time. Quoting the SAP HANA Administration Guide [1]:

- *Redistributing tables may also be useful if you suspect that the current distribution is no longer optimal.*
- *During productive operation, you may discover that the initial assignment of tables and table partitions to index servers is no longer optimal, for example, frequently joined tables are located on different servers.*
- *In a distributed SAP HANA system, partitioned tables are distributed across different index servers. The location of the different partitions can be specified manually or determined by the database when the table is initially partitioned. Over time, this initial partitioning may no longer be optimal, for example, if a partition has grown significantly.*

This leads to the periodic need to do table redistribution / repartitioning, and repartitioning in particular is an expensive, manual operation, as acknowledged by SAP [1]:

> *Performing a partitioning operation on a table in the above ways can be costly for the following reasons:*
>
> - *It takes a long time to run, up to several hours for huge tables.* ●
> - *It has relatively high memory consumption.* ●
> - *It requires an exclusive lock (only selects are allowed).* ●
> - *It performs a delta merge in advance.* ●
> - *It writes everything to the log (required for backup and recovery).*
>
> *Recommendation – (Re-)partition tables before inserting mass data or while they are still small. If a table is not partitioned and its size reaches configurable absolute thresholds, or if a table grows by a certain percentage per day, the system issues an alert.*

SAP argues that administrators should do table redistribution instead of table repartitioning, e.g. – see SAP Documentation excerpt [1] below.

> *Although it is possible to (re-)partition tables and merge partitions manually, in some situations it may be more effective to use the redistribution operation available for optimizing table partitioning (for example, if a change of partition specification is not*

required). Redistribution operations use complex algorithms to evaluate the current distribution and determine a better distribution depending on the situation.

This redistribution has to be done manually, and is bound to be a time consuming operation for large data sets.
Note: The SAP documentation is not clear at this time regarding the availability impact is for such data redistribution.

Insight #3: SAP HANA – NO HA
Scale-out Architecture Requires Expensive Data Redistribution

SAP's reliance on shared nothing approach means that the underlying data has to be partitioned across separate Index Servers and periodically, to optimize data access performance or after changes in the node cardinality, this data set has to be redistributed among available nodes. For large-scale datasets, this can be a grossly inefficient and expensive operation. In contrast, Oracle RAC, in which the nodes have equal access to all data, has none of these limitations. Nodes can be dynamically added / removed with no impact on overall data availability and layout.

SAP HANA: Analysis of Disaster Recovery Features

HANA's Disaster Recovery (DR) support hinges on two capabilities:

- Storage replication: Continuous replication (mirroring) between primary storage and backup storage over a network (may be synchronous). This solution is provided by storage partners (e.g. HP, HDS), and was the only DR support available in earlier HANA releases.
- System replication: Continuous (currently: synchronous) update of secondary system by primary system, including in-memory table loading. This capability was first introduced in SPS05.

Storage Replication

SAP relies on specific certified hardware partners to offer a storage-level replication solution, which delivers a copy of the persisted volumes or file-system to a remote, networked storage system. For synchronous replication, the SAP HANA transaction only completes when the locally persisted transaction log has been replicated remotely. SAP suggests synchronous storage replication can be used only where the primary-DR site distance is 100 kilometers or less. In any case, this solution requires a reliable, high bandwidth and low latency connection between the primary site and the secondary site.

Refer to [11] and [12] for examples on how this configuration looks. As evident from these references, while the storage-bits are propagated to the remote storage arrays, no servers can be mounted on those remote arrays while mirroring is in progress.

Among the hardware vendor partners, HP has made a big deal [13] about validating a hardware solution to provide DR for HANA. The following diagram presents a high-level view of the HP solution for HANA-DR [14].

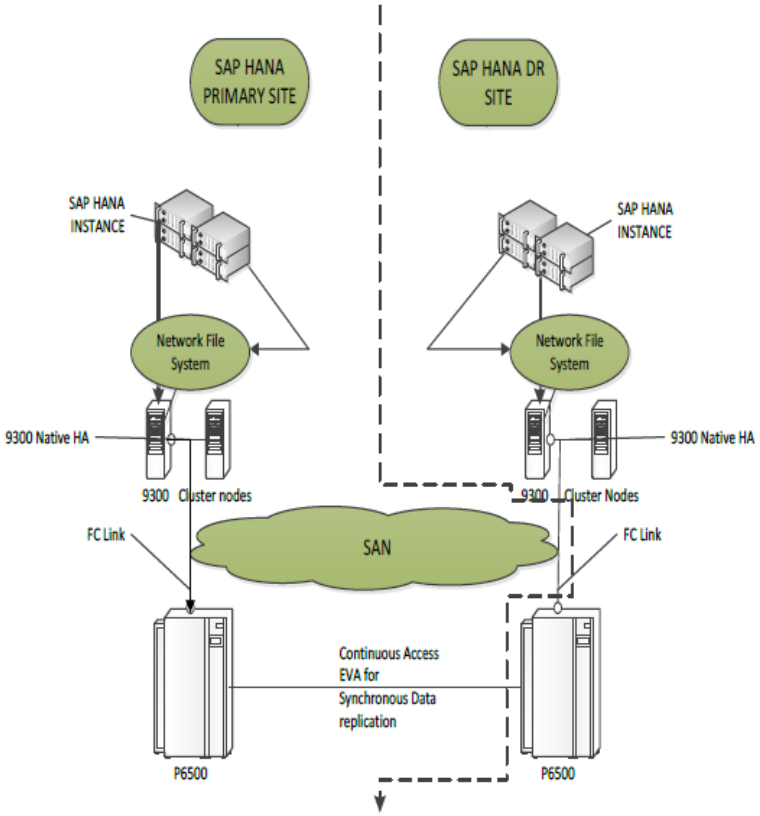

Fig: 6: SAP HANA DR With HP-based Storage Mirroring

<u>Insight #4</u>: SAP HANA – NO HA
DR Support Recommends Storage Mirroring, an Inadequate DR Solution

HANA's reliance on hardware storage partners for the underlying storage mirroring solution shows that HANA's own built-in DR solutions are still incomplete. As is well known in the database industry, a DR solution based on storage mirroring is fraught with risks. The remote storage volumes can't be used while mirroring is in session, which means that not only are there idle DR resources, but also that there is no assurance that the mirrored bits are valid. Bit corruptions originating on the primary storage array will be propagated to the remote array rendering both useless. RTO can be hours because the failover process involves cold start of servers on un-validated storage volumes. Also, a completely different set of technology best practices need to be integrated with overall DR guidelines, leading to manageability and operational inefficiencies.

System Replication

System replication [15], which is new in SPS 05, is SAP's attempt to provide a native DR solution similar to Oracle Data Guard. It still has several major deficiencies that are worth pointing out. Let's look first at the basic capabilities.

System Replication requires a secondary standby system that is configured as an exact copy of the active, primary system. Thus – if there are 'N' worker hosts in the primary, there has to be 'N' worker hosts on the secondary. This also means that for scale-out purpose, hosts must be added equally to both primary and secondary sites. Note that adding hosts on the primary requires shutdown and un-registration of the secondary. The .ini file configuration must be identical for both systems. Any changes made manually, or by SQL commands on one system must be manually duplicated on the other system.

Each service instance of the primary system communicates with a counterpart in the secondary system. The secondary system can be local or remote.

The instances in the secondary system operate in recovery mode. In this mode, all secondary system services constantly communicate with their primary counterparts, to receive and persist data and logs. It does not immediately apply the logs. Instead, periodically, incremental data snapshots are transmitted asynchronously from time to time from the primary system to the secondary system. If the secondary system has to take over, it applies only that part of the log that represents changes that were made after the most recent data snapshot.

In addition to snapshots, the primary system also transfers status information regarding which table columns are currently loaded into memory. The secondary system correspondingly preloads these columns. The secondary system is idle otherwise – it cannot accept user requests or queries.

The various methods of replication supported by System Replication are:

- Synchronous: Whenever logs are persisted in the primary system, they are also sent to the secondary system. The primary system delays committing the transaction until it receives a reply that the log is persisted in the secondary system. This is the zero data loss solution.
- Synchronous in-memory: The primary system commits the transaction after it receives a reply that the log was received by the secondary system, but before it was persisted. The transaction delay in the primary system is shorter than the previous case, because it only includes the data transmission time and no disk I/O latency. This option provides better performance, but is slightly more vulnerable to data loss.
- Asynchronous (new in SP06): The primary system sends redo log buffers to the secondary system asynchronously. This implies that the primary system commits a transaction when it has been written to the log file of the primary system and sent to the secondary system through the network. It does not wait for confirmation from the secondary system. This option provides better performance because it is not necessary to wait for log I/O on the secondary system guaranteed. However, it is more vulnerable to data loss.

If the connection to the secondary system is lost, or the secondary system crashes, the primary system times out and resumes operations.

In the event of a failure that justifies full system failover, an administrator instructs the secondary system to switch from recovery mode to full operation. The secondary system becomes the primary system by replaying the last transaction logs, and then starts to accept queries.

After a takeover operation is complete, and the data center is back in operation, the former primary now has to be registered as the secondary with the active primary system. The former primary has to be completely initialized before it can be used as a secondary – ref. the Administration Guide [1]:

> *This is the same procedure as is used for setting up a normal secondary described in Configuring the Secondary System. The data, that is already available in the persistence cannot be reused, instead a complete initialization will be carried out. This means a full replica will be shipped until the "old primary" is in sync again.*

To support failover with client libraries, the administrator has to provide a list of hostnames from the primary and secondary systems. If a host is not available, the client times out and uses the next host from the list. In the case that none of the hosts are available is a connection error generated.

HA Analysis: Analyzing HANA's DR Capabilities

HANA's DR support is quite primitive and pales in comparison with Oracle Data Guard.

Insight #5: SAP HANA – NO HA
Primitive Disaster Recovery Capabilities

1. **Inflexible**

   If the primary system has 'N' worker hosts, secondary system also needs 'N' worker hosts. Similarly, if 'N' worker hosts need to be added to the primary, the same number of must be added to the secondary. Not only that, the secondary system should be shut down (implies no DR protection) while the hosts are added on the primary. Data Guard has no such restriction.

2. **Idle DR Resources**

   Unlike Active Data Guard, the secondary system can operate only in recovery mode. The secondary system can't be open read-only even after stopping the recovery process. Naturally, there is no such capability as Data Guard Snapshot Standby in which the standby can be leveraged for Dev/Test. The secondary system cannot be used for backups. There is no assurance that the secondary system will actually work – unless you do the failover, and it may be too late then! System Replication is more than a decade behind Data Guard in terms of DR capabilities.

3. **No Support for Zero Data Loss for Asynchronous Standbys**

   System Replication asynchronous standbys, but there will be data loss if failover ensues to that standby. Data Guard eliminates that exposure through Data Guard Far Sync in Oracle Database 12c. SAP HANA has no such capability.

4. **Poor RTO**

   There is no support for automatic failover, such as Data Guard Fast-Start Failover. Failover time is also delayed by pending log apply (since the secondary system does not continuously apply logs). In addition, client failover in System Replication requires an iteration over a list of hostnames. If a host is unresponsive, clients will be stuck on a TCP/IP timeout. There is no automatic client notification facility such as Fast Application Notification (FAN) in Oracle.

5. **Only Basic Role Transition Capabilities**

   System Replication supports only a Data Guard Failover-like capability. There is no support for something like Data Guard Switchover, which assures zero data loss and is extremely helpful for minimal downtime system upgrade/refresh projects. Not only that – after a takeover, the old primary in System Replication has to be completely initialized from a backup – a time/network consuming affair, with the new primary being un-protected while this occurs.

6. **Lacking Several Core DR Features**

   System Replication is at least a decade behind Oracle Data Guard in terms of overall DR capabilities. Features such as Data Guard Far Sync, Rolling Upgrades, Reader Farms, Network Compression, Automatic Block Corruption Repair, Lost Write Detection, Multiple Standbys, Flexible Cascaded Standbys, etc. have made Data Guard the benchmark DR solution. System Replication with HANA is essentially a primitive disk backup solution where the disk happens to be located at a different storage array up to a limited regional distance.

SAP HANA: Analysis of Backup & Recovery Features

SAP HANA uses in-memory technology, but it fully persists two types of data to storage: transaction logs, and data changes in the form of savepoints ([1], [16]).

A transaction redo log is used to record a change at commit time. Upon an outage, the most recent consistent state of the database can be restored by restoring the savepoint and replaying the changes recorded in the log, redoing completed transactions and rolling back incomplete ones.

At a savepoint, all the changed data are written to storage, in the form of pages (i.e. blocks). When starting up the system, logs can be processed from the last savepoint position. Savepoints are coordinated across all processes and instances of the database to ensure transaction consistency. By default, savepoints are performed every five minutes, but this can be configured.

HANA supports two kinds of backups and both can be done only when the database is online:

- Backing up the savepoints in the form of full data backups, which can be used to restore a database to a specific point in time
- Smaller periodic log backups to recover from fatal storage faults to minimize the loss of data.

HANA also supports third party backup products by exposing a backup interface, called *Backint for SAP HANA Interface*, supported only for Linux at this time. These third party tools must use this interface to communicate with HANA for backup and recovery purpose. Several Backint communication processes may be executing at the same time to enable backups of a distributed HANA system. The backup destination used for these third party tools is a fixed location (for both data and log backups) and cannot be changed. This destination really serves as a pipe to determine where the backups are eventually written.

For recovery, the data backups, the log backups, and the log area are used. At the beginning of a recovery, all the data and log backups to be used must be either accessible in the file system or available through the third-party backup tool. When the data backup has been successfully recovered, the log entries from the log backups and the log area are automatically replayed. To recover the database to a particular point in time, a data backup and all the log backups up to the point in time for recovery are needed. Once started, a database recovery cannot be interrupted. If a recovery fails, the complete recovery must be repeated.

HANA uses a Backup Catalog, which is stored in the HANA database itself and gets backed up and versioned as part of the log backup. It contains various pieces of information, such as:

- Backups performed for a database
- The start and finish times of the backups
- Whether a backup is still running
- Whether a backup was successful or not
- Volumes that were backed up
- Log backups and which part of the log they contain
- Backup destinations and their sizes
- The destination type (File or Backint)
- An external backup ID (if a third-party backup tool is used)

With this information, the Backup Catalog can:

- Determine whether a recovery is possible
- Determine which data and log backup to use to recover the database
- Determine which backups are no longer needed

HA Analysis: Analyzing HANA's Backup & Recovery Capabilities

Similar to DR capabilities, HANA's Backup & Recovery support is also very basic and significantly lags Oracle RMAN and Oracle Secure Backup.

Insight #6: SAP HANA – NO HA
Very Basic Backup & Recovery Capabilities

1. **Barebones – Full Backup Only**

   HANA's Backup & Recovery Capabilities are really barebones – basically supporting full data backups (through savepoints) and redo log backups – to disk, either natively, or using third party backup tools. This means every data backup is a time and network consuming full backup – there is no support for incremental backups. It is really unthinkable that a technology touted as modern doesn't have incremental backup support.

2. **No Granularity Beyond Data Files and Logs**

   For HANA, backup and recovery always apply to the whole database – it is not possible to backup and recover individual database objects. For example, unlike RMAN, it is not possible to recover individual data files or individual tables.

3. **No Integrated Tape Backup / Limited Media Management Support**

   Unlike RMAN + Oracle Secure Backup, HANA allows backups only to a specified disk destination. To do tape backups, third party backup products must be used, but they are supported only for Linux.

4. **Poor Design Around Backup Catalog – Significant Data Loss Exposure**

   Unlike RMAN, the HANA Backup Catalog is located inside the HANA database itself. This means that if the HANA database gets corrupted / destroyed, all relevant backup information will be lost, existing backups cannot be retrieved, and the administrator has to resort to the last valid copy of that Catalog leading to potentially very significant data loss. Note that the SAP HANA Admin Guide [1] doesn't describe possible database salvage scenarios when the Backup Catalog is lost.

5. **Inefficient Recovery Process**

   For HANA, a database recovery process, once started, cannot be interrupted. If a recovery fails, the complete recovery must be repeated. This is extremely inefficient – especially for large databases. Unlike HANA, Oracle RMAN-based restore is optimized such that it can be interrupted anytime, and upon resumption, it will only restore the files that were not restored in the previous attempt. This also applies to the Oracle recovery process (available since Oracle version 7).

6. **Lacking Several Enterprise-level Backup & Recovery Features**

   HANA's Backup & Recovery capabilities are at least a decade behind Oracle RMAN and Oracle Secure Backup. Glaring features that are missing are: backup encryption, backup compression, block recovery, incrementally updated backups, optimized backups, parallel backup channels, media management integration. HANA's support of only primitive backup & recovery capabilities demonstrate that HANA itself is not enterprise ready.

SAP HANA: Summary Analysis of all HA Features

Besides support for Scalability, Disaster Recovery, and Backup & Recovery, a well-rounded HA solution must include support for a variety of other capabilities that address various aspects of unplanned and planned downtime. For example – human errors are one of the biggest causes of downtime – when an administrator accidentally deletes important database tables or removes important database files, a well-rounded HA solution must include the capability to efficiently undo such errors. HANA has no such capability. Similarly – reducing various aspects of planned downtime – e.g. software or hardware upgrades, technology refreshes, etc. should also be supported. HANA has none.

In contrast, Oracle Maximum Availability Architecture (MAA), integrated with the Oracle Database, provides customers with a comprehensive and integrated set of industry-leading HA technologies that can be deployed at minimal cost, help avoid downtime, enable rapid recovery from unplanned failures, and minimize the impact of planned outages.

Following is a list of critical HA features that comprise Oracle MAA, along with the benefits they provide, and also how the equivalent solutions from SAP HANA compare.

<u>Insight #7</u>: SAP HANA – NO HA
Lacks the Depth and Breadth of Oracle Maximum Availability Architecture (MAA)

Oracle MAA Feature	Benefit	Corresponding HANA Feature	Notes
Oracle RAC	Scalability through active-active clustering, Server HA	HANA Distributed System	Based on legacy cold cluster technology, expensive data redistribution needed
Application Continuity	Replays application context automatically on surviving nodes	Missing	
• Flashback Technologies • Flashback Query • Flashback Versions Query • Flashback Transaction Query • Flashback Transaction • Flashback Table • Flashback Drop • Flashback Database	Quickly and efficiently unwind human errors	Missing	
Database block checking, database block checksum, lost write protection, automatic block repair	Integrated protection from Data Corruptions	Missing	
Automatic Storage Management (ASM)	Integrated volume management, database-optimized protection from disk failure	Missing	
Recovery Manager (RMAN)	Integrated backup and recovery	HANA Backup & Recovery	Basic backup & recovery support
Oracle Secure Backup	Tape backup solution	Missing	
Oracle Active Data Guard	Comprehensive data protection, disaster recovery and data availability	HANA System Replication	Basic disaster recovery support
Oracle GoldenGate	Flexible active-active replication	Missing	
Oracle Site Guard	Failover orchestration across sites and system stacks	Missing	
Global Data Services	Failover and load-balancing of Application Services across local and remote data centers	Missing	
Online System Reconfiguration	Reduction of planned downtime for system refreshes (hardware, network, storage, data centers, etc.)	Limited	Limited support – e.g. production operations need to be stopped before adding hosts
Rolling Upgrades (RAC, Data Guard, GoldenGate)	Reduction of planned downtime for software upgrades	Limited	
Edition-based Redefinition	Online Application Upgrades	Missing	
Online Table Redefinition & Online Operations	Changing database structure without application downtime	Missing	
Maximum Availability Architecture (MAA)	Integrated set of HA Best Practices		**Missing**

Customer References

Oracle's HA solutions have shown remarkable customer adoption and market success, and continue to be a critical differentiator when choosing a database technology that can support the 24x7 uptime requirements of today's businesses. Oracle MAA has been deployed at thousands of highly critical customer sites around the world, running global economic operations related to the top banks, retailers, manufacturers, telcos, etc.

A focused list of customers who have implemented various Oracle HA solutions, along with detailed case studies, are available in the Oracle Technology Network public site [17]. These success stories about Oracle MAA in action at some of the best names in various industry verticals across the world attest to Oracle's unparalleled technical superiority in the area of high availability.

Another fact – quite ironical in this context, is what is pointed out by a research note from IDC [18] that over 80% of SAP's ERP customers use Oracle Database.

In contrast – information related to SAP HANA's HA-focused customer case studies has been extremely limited. For example, RTO (Recovery Time Objective) is an important consideration for any HA implementation – however, there hasn't been any publicly available material highlighting SAP HANA enabling fast failovers to their standby system in the event of a disaster and enabling business continuity. This may be contrasted with the Oracle Data Guard failover time reported [19] by Intel – ranging between only 2 to 30 seconds!

High Availability: Something to Remember

Databases are the beating heart of an enterprise. When databases suffer downtime – because of unexpected failures, or even because of any planned maintenance activity – the outcome usually is not good for the company. Business is lost, goodwill is wasted, and such downtime-causing incidents make national news – in a negative manner

In contrast to HANA, Oracle MAA is a significantly matured and hardened product architecture based on HA-related customer feedback and experience that Oracle has acquired for more than a decade. While the architecture itself is robust, MAA has addressed numerous availability-related corner cases, knowledge of which can only be built through this decade-long experience.

Any pragmatic business with availability as one of its main IT focus areas would be extremely ill-advised to take the HANA route.

Conclusion

Successful enterprises understand the vital importance of maintaining highly available technology infrastructures to protect critical data and information systems. At the core of many mission critical information systems is the database system, responsible for the availability, security, and reliability of the information technology infrastructure. If this database system lacks any of these fundamental capabilities, it is simply not enterprise-ready – regardless of other features it may have.

Building on decades of innovation, Oracle Database offers the industry's gold standard for integrated high availability and data protection that ensure customers can maximize their application uptime in the event of both planned maintenance activities and unexpected failures. In contrast – as this document demonstrates, SAP HANA, even though touted as the next-generation database architecture, lacks many of the fundamental high availability capabilities needed to run today's 24x7 enterprises.

Final Insight: SAP HANA – NO HA
SAP HANA is not Enterprise Ready

With very primitive HA features, SAP HANA falls significantly short of the availability demands of today's 24x7 businesses. SAP HANA cannot be deployed at enterprise scale.

References

1. SAP HANA Administration Guide: http://help.sap.com/hana/SAP_HANA_Administration_Guide_en.pdf
2. SAP HANA Documentation: http://help.sap.com/hana_appliance/
3. Oracle Database High Availability: http://www.oracle.com/goto/availability
4. Oracle Database In-Memory Option: http://www.oracle.com/us/corporate/features/database-in-memory-option/index.html
5. IBM Redbook: *In-memory Computing with SAP HANA on IBM eX5 Systems*, http://www.redbooks.ibm.com/redbooks/pdfs/sg248086.pdf
6. SAP Collateral: SAP HANA In Data Centers, Slide: "SAP HANA Persistence" - http://www.saphana.com/servlet/JiveServlet/download/2010-9-12161/SAP%20HANA%20in%20Data%20Centers%20(SPS6%2B).pdf
7. SAP Collateral: SAP HANA In Data Centers, Slide: "Scale Out" - http://www.saphana.com/servlet/JiveServlet/download/2010-9-12161/SAP%20HANA%20in%20Data%20Centers%20(SPS6%2B).pdf
8. SAP Collateral: SAP HANA In Data Centers, Slide: "SAP HANA Scalability" - http://www.saphana.com/servlet/JiveServlet/download/2010-9-12161/SAP%20HANA%20in%20Data%20Centers%20(SPS6%2B).pdf
9. Blog: *SAP HANA – Core Architecture*, http://en.community.dell.com/techcenter/b/techcenter/archive/2012/09/28/sap-hana-core-architecture.aspx
10. SAP HANA Administration Guide: "*Monitoring Host Status and Auto-Failover Configuration*" http://help.sap.com/hana/SAP_HANA_Administration_Guide_en.pdf
11. SAP Collateral: SAP HANA In Data Centers, Slide: " SAP HANA Disaster Recovery: Storage Replication" - http://www.saphana.com/servlet/JiveServlet/downloadBody/4351-102-2-9243/SAP%20HANA%20in%20Data%20Centers%20(SPS7%2B%20iFG).pdf
12. SAP Collateral: *SAP HANA – High Availability*, Section "Storage Replication", http://www.saphana.com/servlet/JiveServlet/downloadBody/2775-102-4-9467/HANA_HA_2.1.pdf
13. HP AppSystems for SAP HANA, http://h17007.www1.hp.com/us/en/converged-infrastructure/converged-systems/appsystems/sap-hana.aspx
14. HP Collateral: *Disaster-Tolerant solution for HP AppSystems for SAP HANA*, http://h20195.www2.hp.com/V2/GetPDF.aspx%2F4AA1-4846ENW.pdf
15. SAP Collateral: *SAP HANA – High Availability*, Section "System Replication", http://www.saphana.com/servlet/JiveServlet/downloadBody/2775-102-4-9467/HANA_HA_2.1.pdf
16. SAP Collateral: *SAP HANA – High Availability*, Section "Backups", http://www.saphana.com/servlet/JiveServlet/downloadBody/2775-102-4-9467/HANA_HA_2.1.pdf

17. Oracle HA Customer Case Studies: http://www.oracle.com/technetwork/database/features/ha-casestudies-098033.html

18. IDC Insight: *SAP and Sybase: A Marriage Made In Database Heaven*, June 2011, http://www.sybase.com/files/White_Papers/IDC_SAP_Sybase_DatabaseHeaven_report.pdf

19. *High Availability & Disaster Recovery with Oracle Data Guard Fast-Start Failover – An Implementation at Intel*, Oracle OpenWorld 2011, http://www.oracle.com/technetwork/database/features/availability/13441-intel-515449.pdf

SAP HANA: Analysis of HA Capabilities
February 2014
Author: Oracle HA Development

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together