Invincible Supply Chain

Reference Architecture for Mission-Critical SAP® Advanced Planning & Optimization in SAP liveCache with HotStandby

IBM | SAP

High availability for SAP® Supply Chain Management
SAP Advanced Planning & Optimization
SAP liveCache
DB2 HADR
SAN Volume Controller
PowerHA System Mirror
Tivoli System Automation for Multiplatforms

Invincible Supply Chain

High Availability for Mission-Critical SAP Advanced Planning & Optimization – Design, Reference Architecture, and Proof of Concept –

IBM SAP International Competence Center (ISICC)
Walldorf, Germany

IBM Storage Lab Services, Mainz, Germany

IBM Boeblingen Lab, Rot, Germany

SAP teams:
SAP Labs, Berlin, Germany
Performance, Data Management, and Scalability, Walldorf, Germany
Solution Management, Supply Chain Management, Walldorf, Germany
Product Management, Supply Chain Management, Walldorf, Germany

21/ June 2011

1 Table of Contents

1 Table of Contents 4
2 Preface 5
2.1 Document scope 5
2.2 Special notices 5
2.3 Version information 5
2.4 Document layout and design 5
2.5 Authors of this document 6
2.6 With gratitude and acknowledgment of our sponsors 6
2.7 Project team 7
3 Introduction – Why High Availability for SAP Advanced Planning & Optimization 8
4 Benefits Summary 10
4.1 Target industries and solutions 10
4.2 HotStandby versus a failover solution 12
5 Available-to-Promise – A Day in the Life 14
6 Proof of Concept Scope 16
6.1 Application scope and goals 17
6.2 Summary of KPI results 23
7 Design Components of the Infrastructure 26
7.1 Infrastructure scope and stack 26
7.2 High-availability infrastructure design on Power servers – the basis 30
7.3 Tivoli clustering – Tivoli System Automation for MultiPlatforms 49
7.4 Tivoli clustering for IBM DB2 HADR 50
7.5 Tivoli clustering for SAP central services for ABAP (ASCS instance) 60
7.6 SAP liveCache with HotStandby design and requirements 68
7.7 Design of the PowerHA Cluster for SAP liveCache HotStandby 77
8 Summary and Conclusions 97
9 Appendix 98
9.1 Related documents and sources of further information 98
9.2 SAP liveCache versions 99
9.3 AIX, PowerHA, and Java versions 99
9.4 Tivoli SA MP and DB2 HADR versions 101
9.5 Power Systems, AIX, and storage versions 101
10 Copyrights and Trademarks 102
11 Disclaimer and Special Notices 103

2 Preface

2.1 Document scope

This document is intended for architects and implementations teams that have the responsibility of designing and implementing a highly available infrastructure stack for the SAP® Advanced Planning & Optimization (SAP APO) component in a mission-critical landscape.

2.2 Special notices

2.3 Version information

This is version 1.1 of this document including architecture, design and concept.
Online versions of this document and subsequent releases can be found at this website:
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100677

2.4 Document layout and design

This document is separated into component areas. Each clustered component of this infrastructure stack can be implemented independently of the other clusters. Therefore, the document groups the basis component and the cluster solution for each of the component areas together but provides no dependency on the sequence of installation. As an example, HADR with IBM® Tivoli® System Automation for Multiplatforms (SA MP) can be the only cluster in a supply chain management (SCM) system if high availability for all components is not deemed necessary (non-mission-critical load). Within each of the component areas, there is a sequence path.

2.5 Authors of this document

- Carol Davis, Consulting IT Specialist, SAP International Competence Center
- Elke Hartmann-Bakan, IT Specialist, SAP International Competence Center
- Jan Muench, Advisory IT Specialist, IBM, Infrastructure and Technology Service
- Katharina Probst, Developer, IBM Boeblingen Lab

2.6 With gratitude and acknowledgment of our sponsors

We thank those who have given the support and vision that allow us to provide these case studies in support of our joint customer needs.

Juergen Primsch, Vice President, SAP MaxDB & SAP liveCache, SAP

Dr. Ulrich Marquard, Senior Vice President, Performance & Scalability, SAP

Dr. Volkmar Soehner, Development Manager, SAP liveCache Applications, SAP

Bill Gilmour, Industry General Manager, Consumer Products, IBM

Emily Benner, Director, System and Technology Group, ISV Technical Enablement, IBM

Michelle Tidwell, Program Director, System Storage, ISV Enablement, IBM

Laurent Montaron, Manager - POWER and Next-Generation Platform ISV programs, IBM

Vandana Kumar, SAP Strategy and Enablement Manager, IBM

Dr. Antonio Palacin, Director of ISICC IBM SAP International Competence Center, Global IBM – SAP Alliance, IBM

Robert D. Thomas, Vice President, Business Development, Information Management, IBM

We would also like to thank Tim Main, Consulting IT Specialist and IBM Client Technical Advisor for Unilever, for his guidance throughout and the extended technical steering team whose insight into actual requirements was vital to this project.

2.7 Project team

This project and its results come from the efforts of a joint IBM and SAP team, crossing many areas of specialization and responsibility. This type of team demonstrates the level of co-innovation enthusiasm and reflects the technical collaboration between these two companies.

Name	Responsibility	Position	Company
Carol Davis	Project Design and Execution	Consulting IT Specialist	IBM
Vandana Kumar	Product Management and Delivery	Technology Business Development Executive, Systems and Technology Group	IBM
Elke Hartmann-Bakan	DB2 HADR, Tivoli System Automation	IT Specialist, Data Management	IBM
Katharina Probst	SAP liveCache and PowerHA	Developer, HADR Solutions for SAP on AIX	IBM
Jan Muench	HA Infrastructure and Virtualization	Advisory IT Specialist, Power Systems	IBM
Werner Thesing	SAP liveCache HA	Development Architect, SAP MaxDB SAP liveCache	SAP
Erika Wolf	SAP Advanced Planning & Optimization DP	SAP liveCache Applications	SAP
Anette Foellmer	SAP Advanced Planning & Optimization DP	Senior Developer, Performance and Scalability	SAP
Oliver Goos	HSS library	IT Specialist, ESCC Mainz, Systems and Technology Group, Lab Services	IBM
Blandine Alazard	HSS library	Advisory IT Specialist, ESCC Mainz/ Systems and Technology Group, Lab Services	IBM
Gerald Heisig	Available-to-Promise	Product Management, Supply Chain Management	SAP
Claus Bosch	Available-to-Promise	Solution Manager, Supply Chain Management	SAP
Wolfram Schick	Available-to-Promise	Development Architect, Supply Chain Management	SAP
Ivan Sesar	Available-to-Promise	Product Management, Supply Chain Management	SAP
Ulrich Mast	Available-to-Promise	Solution Manager, Supply Chain Management	SAP
Marc-Stephan Tauchert	Performance and Sizing	Certified SAP Technical Specialist, SAP Growth Program	IBM
Ursula Zachmann	Performance and Sizing	IT Specialist, SAP Solutions	IBM
Dr. Helmut Mueller	Landscape Discovery and Monitoring	Senior Client IT Architect	IBM

With technical contributions from:

- Walter Orb, Consulting IT Specialist, IBM SAP International Competence Center
- Maik Gasterstaedt, IT Specialist Storage, IBM SAP international Competence Center
- Andreas Schauberer, Tivoli Development, IBM Boeblingen Lab
- Hinnerk Gildhoff, Software Developer, DB2 HADR SAP, IBM Böblingen Lab
- Olaf Depper, IT Specialist, DB2 SAP, IBM Böblingen Lab
- Bernhard Buehler, IT Specialist, Systems and Technology Group, Lab Services, IBM Germany

3 Introduction – Why High Availability for SAP Advanced Planning & Optimization

SAP Advanced Planning & Optimization (SAP APO) is an advanced planning and scheduling software that provides the tools needed to optimize supply chain processes at strategic, tactical, and operational planning levels. Demand planning, supply network planning, and production planning and detailed scheduling are three of the core business scenarios addressed by this component. For companies operating worldwide, planning information must be made available across system boundaries as quickly as possible to support business efficiency. Global available-to-promise (global ATP) in SAP Advanced Planning & Optimization provides this key functionality.

The architecture of SAP APO spans two databases: the typical SAP application or ABAP™[1] database, and the memory resident database used for optimization and planning, which is the proprietary SAP liveCache. These two databases are logically interlinked at application level, but do not know each other at database level. Related but unique data exists in each of the two databases, but neither has the complete data content. If these two databases lose their data synchronization, which can result from an infrastructure failure, or the need to restore one or the other of the two, the application provides the means to resynchronize at application level. This resynchronization will normally result in the identification of data that exists in one database and for which there is no reference in the other. Restoring synchronization will require an understanding of the significance of the data and the ability to take informed decisions. In summary, the recovery driven from the application level may have an extremely long path length and require expert business skills.

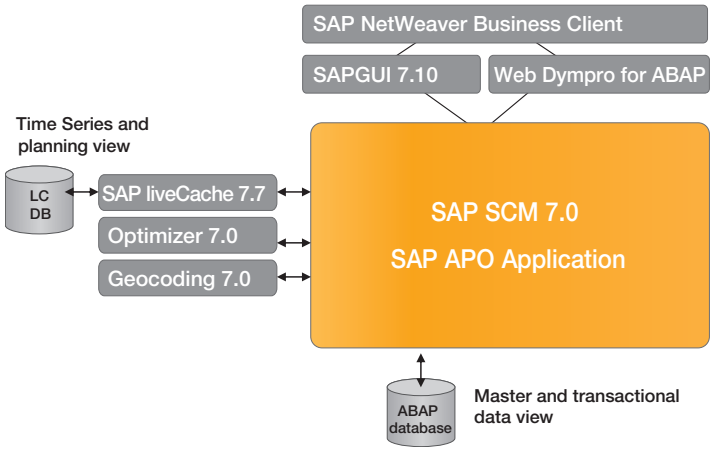


Figure 3.1: Overview of SAP APO architecture with liveCache and the SAP SCM optimizer

As failures come in many different guises, from logical errors to infrastructure disasters, this recovery is part of the operational lifecycle of any production system. The goal is to extend duration of the undisrupted production time to the maximum possible. A logical error is something the infrastructure cannot protect against, but there are many situations in which the design of the infrastructure and middleware can support the application and the business.

1 Database used for master and transactional data, customizing, and ABAP objects that provide the business content of the SAP system. ABAP is a proprietary object-oriented programming language from SAP.

The proof of concept being described in this document looks at some of these options. The focal point is a joint IBM and SAP solution around the SAP liveCache component itself to ensure high availability and very fast recovery without data loss.

As SAP liveCache is a component of the SAP Supply Chain Management (SAP SCM) application, protecting SAP liveCache make sense when the SAP SCM system itself is highly available. SAP SCM itself is based on ABAP and the concept is therefore built on the current best practices for high-availability solutions for ABAP systems. The design then extends the availability concept to cover the ABAP database, the critical SAP central services software, and the infrastructure itself – top to bottom.

This document describes reference architecture for SAP Advanced Planning & Optimization based on end-to-end high availability.

The proof of concept is based on IBM Power systems™ technology, IBM DB2®, and two alternative clustering solutions for high availability – Tivoli System Automation for Multiplatforms, and IBM PowerHA®.

The choice of DB2 database was made because the functionality DB2 HADR (which includes features such as independent data images, closely synchronized, and quick failover) fits very well in the target solution. DB2 is also tightly integrated with SAP software and with the Tivoli System Automation for Multiplatforms (Tivoli SA MP) cluster solution.

SAP delivers Tivoli SA MP clustering as part of the SAP solution for DB2 HADR. The implementation can be extended to cover the critical SAP components. This combination is used quite extensively for SAP ABAP-based systems, such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), as well as Supply Chain Management (SCM). The portion of this document covering the critical SAP components and DB2 HADR is generally applicable to ABAP systems.

SAP liveCache with HotStandby is a new feature that will be supported by the PowerHA cluster solution. As we demonstrate in this proof of concept, SAP liveCache HotStandby solution provided by PowerHA can also be integrated into an SAP SCM system that is using Tivoli clustering for the other critical SAP components and for HADR.
Additionally, the proof of concept includes a cluster solution for DB2 HADR and the other critical SAP components based on PowerHA, providing a complete PowerHA solution for situations where a mix of cluster technology is undesirable.

4 Benefits Summary

This document shows the benefits to mission-critical SAP APO business processes provided by high availability for SAP SCM. Each component of the SCM stack is looked at separately and can be implemented as a separate cluster to provide high-availability building blocks for SAP APO or other ABAP-based core systems.

The business benefits result from high availability and uninterrupted processing in online order confirmation and extremely fast return to business for production planning. In scenarios, such as service parts management, a five-minute downtime has been quoted as the limit before the business begins to suffer a loss.

The solution presented here explains how this target can be met with effective component redundancy, which also provides a cost-effective solution.

4.1 Target industries and solutions

This proof of concept focuses on the following components:

- IBM PowerVM™ technology – IBM POWER6® and IBM POWER7® processor-based servers
- IBM System Storage® SAN Volume Controller (and any SAN storage) or IBM System Storage DS8300
- IBM AIX® 6.1
- DB2 HADR
- Tivoli System Automation for Multiplatforms
- PowerHA SystemMirror for AIX

In terms of SAP software, this information applies to:

- SAP Advanced Planning & Optimization (SAP APO), including demand planning, supply and network planning, and production planning and detailed scheduling functionality
- Integrated scenarios using SAP APO for available-to-promise (ATP) and Global available-to-promise (GATP) for resource availability and planning
- SAP Service Parts Planning application and other service parts management software
- SAP NetWeaver™ technology platform for ABAP and SAP NetWeaver Business Warehouse; SAP Supply Chain Management, SAP Customer Relationship Management, SAP ERP, and other ABAP-based software.

Target industries typically using this software:

Service parts management functionality in SAP SCM

- Aerospace and defense
- Automotive
- Heavy equipment
- Industrial machinery
- Household appliances
- High-tech

SAP Advanced Planning & Optimization

- Consumer products
- Automotive
- High-tech and electronics
- Household appliances
- Discrete manufacturing (industrial machinery and components)
- Retail
- Chemicals
- Mill products and mining
- Oil and gas
- Life sciences
- Wholesale

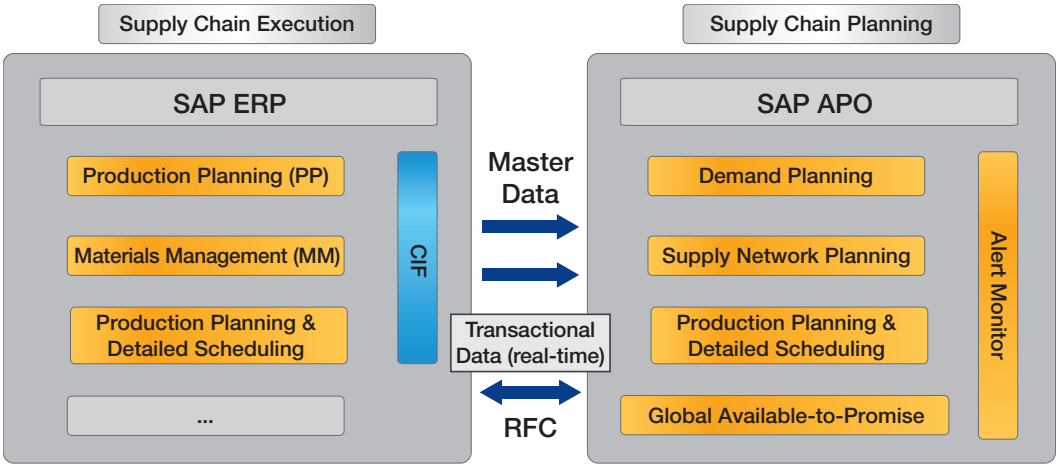


Figure 4.1.1: Diagram of an integrated SAP ERP and SAP SCM system used for ATP

4.2 HotStandby versus a failover solution

In a tightly integrated SAP landscape, the failure and subsequent recovery of a single system can have repercussions on other systems and the consistency of the data that might span more than one system. Resynchronization can be time-consuming. The objective of this high availability design is to avoid interruption as far as possible to the application level, by providing a robust infrastructure design with redundancy. Whenever the redundant hardware infrastructure cannot protect the system, for example, in the failure of the server itself, then recovery of the system moves to the next layer, provided by the clustering solution. The fastest recovery is provided by either HotStandby or by an active/passive design. This method of having two equal components with their own resources provides the more robust solution, as well as the fastest recovery.

A traditional database failover restarts with a database crash image once the disks are successfully brought back online. The database attempting to restart the service has to recover the crash image, which normally includes rolling out all “in-flight” transactional data. Depending on what was happening at the time of failure, this recovery can take a significant amount of time.

If the database is file system-based, which most are, the file systems themselves may need to be recovered before the database activity can begin. This can also take time depending on the number and size of the file systems.

In the case of SAP APO with SAP liveCache, the file systems are recovered, the crash image is used to realign data consistency by rolling back uncommitted transactions, and the SAP liveCache memory resident data cache is reconstructed. This data cache can be many gigabytes in size and also adds to the recovery time for the service.

For a SAP APO system that has gone through a crash recovery, it is recommended to run an application level consistency check to ensure the data spanning the two databases of SAP APO are consistent. This can also take some significant time depending on the number of data objects.

By this time, other processes may well have timed out.

For online ATP, the expectation is that if the system takes longer than two to three minutes to recover, the order entry side of available-to-promise will be broken. Most likely the user will lose patience and kill the session, or the process may timeout in other areas of the technical infrastructure. The result is that materials will be left in temporary assignment and are unavailable until they are either manually recovered or some expiration date is set to clean them up. The result is that it may not be possible to confirm orders that really could have been confirmed because necessary resources appear unavailable.

For production planning batch jobs running in the overnight window, this might mean that skilled application people need to be available for the recovery after a failover. This can delay the time required to restart the jobs after service recovery.

Redundant Data Images

Redundant data images (as provided by the HotStandby and active/passive databases) provide an extra failback in case of an actual data corruption at physical level or loss of a storage server.

A traditional failover, where the storage is taken over by a backup server, will use one copy of the data that is active on only one of the servers at any time. A failover server will take over the data from the failed server and rely on this data image to attempt to restart the lost functionality. If the data is corrupted, a restore of the database is necessary and archived logs will be reapplied to recover to the point in time of the failure. At this point, the resynchronization effort with other systems depends on the integration and dependencies of the business processes.

As used in this proof of concept, DB2 HADR has a complete set of redundant data – both database and logs. These can be on separate storages servers in separate sites. HADR does have to do a certain amount of rollback recovery as part of the takeover to clean up the open transactions. This is done very quickly as the database is already online and active. For large batch planning jobs where one would intuitively expect to generate a large volume of rollback activity in case of a failover, this proves not to be the case. The commit rate of the planning application jobs is very high and, therefore, the recover requirement for rollback of uncommitted data after a failover is very low.

SAP liveCache in HotStandby does no rollback recovery as it is only applying data from transactions. This is not a crash image but an active database in synchronization with the master.

SAP liveCache itself can switch roles within seconds. The cluster software builds in a short delay to validate a failure before initiating a takeover. With the delay, the service is available within the two minutes failover target with all data intact. The consistency check is not necessary.

With SAP liveCache in HotStandby, there are two redundant copies of the data, but there is a shared log. In case of a mirror corruption, you might be unfortunate to lose the log.

In this case, it is possible to drop the corrupted log and recover SAP liveCache from the last valid save point. This would be much quicker then restoring the last version and trying to roll forward from archive logs. In both cases, the data which was in the online log will be missing and a re-synchronization with the SAP APO database will be necessary – but it will be considerably faster than a recovery.

In summary, the major differences:

- Two separate versions of the data for higher data security
- Active/standby database engines, with synchronized data status to take over in seconds with data consistency
- Fast takeover avoids chain reaction of failing dependencies

5 Available-to-Promise – A Day in the Life

A day in the life of a sales person in a collaboration supply chain scenario:

A fictional, but possible, scenario to demonstrate one of the countless situations where customer satisfaction, indeed the sales itself, depends on punctual delivery and the guarantee of punctuality depends on reliable confirmation. What is the value of the theoretically fastest and most-sophisticated system if it is not there when it is mission critical?

Picture a small building supplier somewhere:
The day hang like a wet grey sock, and Harald, sipping his half-cold coffee, appeared to be bent on becoming part of a matching pair if his body posture was any indication. He sighed, and with a last longing glance, he put away the travel brochure of white sands and blue water, sweeping it with a despondent finality into a drawer. An unusual movement from the corner of his eye, caught his attention. His boss was steaming in his direction, weaving through the display arrangements of luxury baths and building materials with an astonishing focus of purpose, headed directly toward Harald and the customer service desk. His whole body language emanated a hitherto unknown tension and excitement. He was being followed by two very serious and rather tired looking men carrying document folders.

It was the order of the year! The large five-star hotel complex, going online with much fanfare and prominence, was the talk of the region. The captain of this luxury liner about to set sail had, however, caught a glimpse of a distant iceberg. The renowned firm scheduled to deliver the sumptuous Jacuzzi baths for the top category suites had informed the construction manager that a portion of their supply chain, on which this delivery depended, had unexpectedly gone into receivership. The delivery date would not be met.
The two men in tow, on this now glorious day, were the hotel's construction manager himself and his lead architect. The boss was preparing to launch the lifeboats – 62 luxury Jacuzzi baths in four different models – all of which could be met by components in their portfolio and, according to the hotel architect, integrated into the hotel's general design without major modification. The question on which all revolved was – could their firm deliver in time?

All eyes were on Harald – for both his boss (who had specifically provided Harald with training on the new order and logistics system), and the other two gentlemen, Harald was now the man of the moment.

It took nearly 40 minutes to get through the first three models and 50 of the 62 baths, each component a separate order line and the delivery of all components verified through the available-to-promise functionality of the logistics system. This system ensured that what was confirmed was either currently available or would be available to meet the delivery date, reserving materials and updating the order and production planning directly. Harald worked conscientiously and accurately – he would make no error on this order. Nevertheless, in the pauses between confirmation responses from the planning system, his concentration did allow for split-second interruptions (little dreamy flashes of tropical nature).

It happened on the third to the last order line confirmation:
In the middle of an order, a dialog box appeared on his screen! Some gibberish about "RFC connections failure" and "gateways"; he hit the Enter key again and the dreaded hourglass appeared. Something in the system had failed, leaving him with an unconfirmed and incomplete order and no idea on how he could proceed.

Now we can leave Harald, sweating in despair, in desperate debate with the help desk assuring him that they are working on it while his potential customers begin tense conversations on their mobile phones, making detailed notes and eyeing the wall clock.

Or…
We can give Harald a break and provide him the system described in the rest of this document. In this case, he only needs to keep calm, perhaps offer coffee – and provide some friendly distraction. He can be confident that regardless what may have happened, the system will be back within two to three minutes. He can then continue exactly where he was interrupted, and complete those last few order lines.

Let us be kind to Harald and say his company can meet the deadline and the 62 Jacuzzi baths are confirmed with guaranteed delivery from various locations. Let this company make its name as a reliable supplier with this large construction firm, and let Harald earn the commission of his young life. As this is fiction, we can easily do this. Nevertheless, this is a realistic and possible sales scenario.

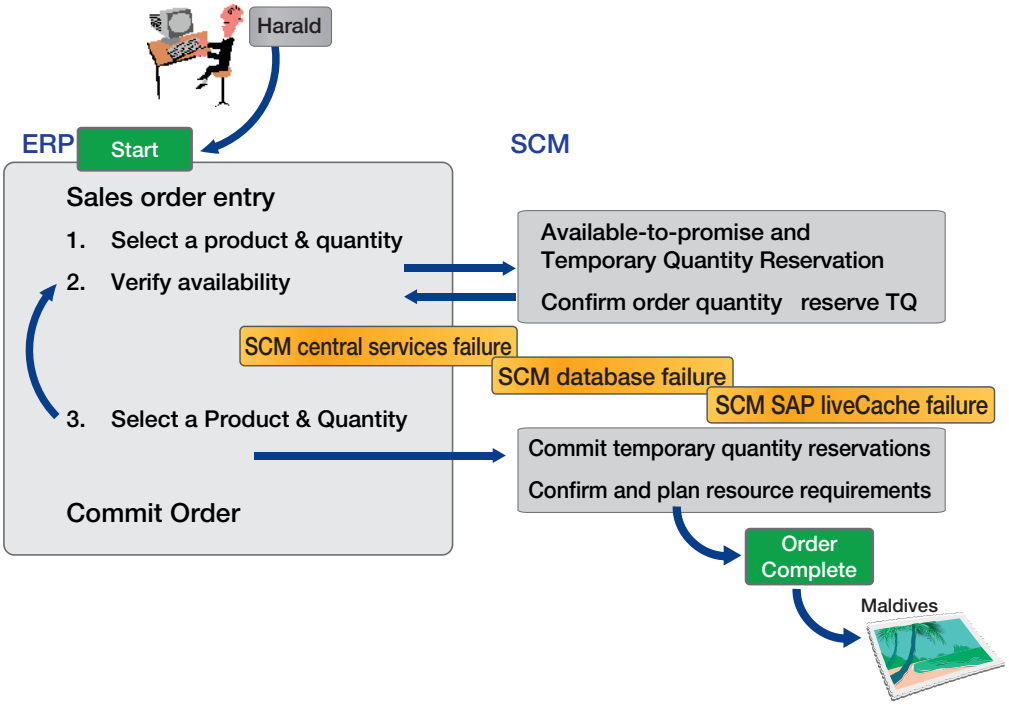


Figure 5.1 A day in the life

We can replace Harald's dilemma with of number of integrated supply chain scenario from sales to service parts planning.

Short of collecting a commission and going off to the Maldives, the team writing this document assumed the part of Harald and went through these same steps to prove the speed of the infrastructure recovery and the business process recoverability provided by the integrated order entry and logistics system using SAP ERP and available-to-promise functionality in SAP APO.

The team successfully completed the open order with business process consistency despite intervening database failure, SAP liveCache failure, server failure, storage failure. and various combinations thereof. The target of this proof of concept is to provide an end-to-end high- availability system and demonstrate the application benefits such an infrastructure can provide.

6 Proof of Concept Scope

The proof of concept covers the design and implementation of a high-availability SAP APO system on a POWER7 processor-based server using infrastructure virtualization (PowerVM). The design of the infrastructure is intended to allow full access to functionality, such as Live Partition Mobility (LPM). For this purpose, the entire solution stack is based on virtual I/O (VIO). Virtual I/O is a functionality that implements an abstraction layer between logical partitions being used as servers, and the physical I/O components of the machine. VIO Servers are special logical partitions (LPARs) that own the physical hardware adapters and provide the mapping of virtual adapters to physical adapters. Virtual adapters and devices are made available by the VIO Server to the server LPARs. This abstraction layer allows multiple server LPARs to use the same physical I/O adapter.

A highly available cluster solution will require redundant I/O adapters, and alternative paths to the network and the storage systems. The number of adapters in any physical server is limited and dedicated redundant I/O adapters per LPAR may quickly exhaust this limit. VIO functionality, therefore, provides a very cost- effective solution for I/O redundancy. Redundant I/O paths are implemented in the VIO level and can be used by all logical partitions within the system. VIO provides infrastructure flexibility, as additional LPARs can be added to the system without regard to the number available I/O slots, as there need be no requirement for additional I/O adapters. The VIO approach is also cost effective, as the number of hardware adapters (and the subsequent number of administered LAN and SAN ports) is kept to a minimum since hardware redundancy and the network and SAN access paths are implemented only once and then shared.

The middleware and mission critical SAP components (databases, central services, and SAP liveCache), are implemented as separate high-availability cluster solutions to demonstrate the building blocks that are available and the strengths of each solution. As SAP APO has a load profile that greatly benefits from processor sharing between components, the system is implemented on micropartitions in a shared processor pool.

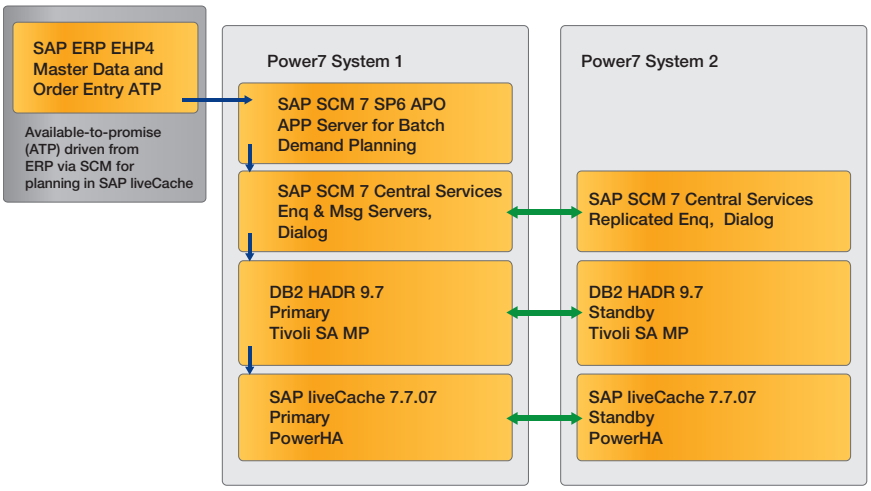


Figure 6.1: Overview of the cluster building blocks and the application components

The clusters were tested for a number of conceivable failures discussed and agreed with the Technical Steering Team made up of client teams and customer architects.

Two application scenarios were selected to drive the proof of concept at application level. These include the ATP check coming from order creation in SAP ERP (similar to the order creation in service parts management functionality) and a batch planning load represented by demand planning functionality. The application server for the batch demand planning jobs is not clustered, as multiple application servers can be configured and therefore the application server represents no single point of failure (SPOF). In the case of a component failure (database, SAP SAP liveCache, or application server), the batch jobs will need to be restarted. The jobs can normally be restarted as soon as the failed component is recovered and therefore, for batch, the focus is on the speed of component recovery. For ATP, the focus is on recovery of the application and continuous availability. For both, the target is availability and data consistency.

6.1 Application scope and goals

This section describes the business processes used to test the infrastructure from the application level. The goal was to test the response of the application to various failover situations and understand the impact on the business. For this purpose, common scenarios were selected and the approach described in detail to make the results easy to map to the normal production business.

6.1.1 Demand planning in SAP APO

The demand planning scenario used for these tests is taken from the standard benchmark for SAP APO. The benchmark scenario was updated for SCM7 by the SAP sizing and performance team for use in these tests and for reverification of the SAP APO sizing for systems running in a virtualized landscape.

For demand planning, the tests performed under load refers to the load generated by 16 parallel demand planning jobs with a total throughput of around 480,000 character combinations per hour at aggregate level.[2]

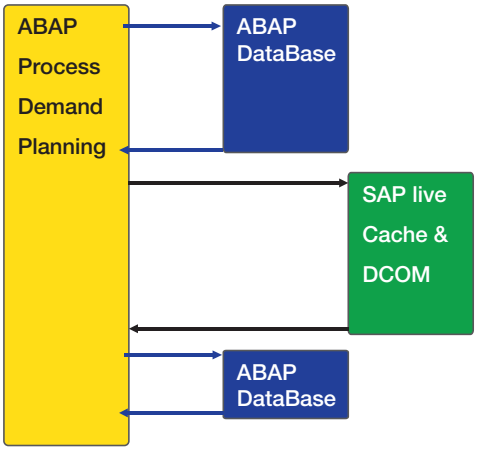


Figure 6.1.1.1: Process flow between SAP work process and the two databases

Demand planning test case

In the proof of concept, demand planning is used to simulate all three of the major batch scenarios (demand planning, supply network planning, and production planning and detailed scheduling) as they have similar behavior profiles. They are all back ground batch jobs running entirely within SAP SCM. For these load tests, demand planning is started in massive parallel, and during the batch run, component failures are initiated.

The demand planning batch jobs run entirely within the SAP SCM system, within the batch application server. In this case, there is only limited traffic between the batch application server and either the enqueue server or the message server. The traffic is primarily between the application server and the two databases: the ABAP database (DB2 HADR) and SAP liveCache.

Figure 6.1.1.2 shows the traffic between the batch application server and the cluster pairs through the cluster service IP addresses.

2 This load is simply a representative load using a portion of the total machine resources. This is neither an SAP SCM benchmark result, nor an indication of any possible high-load achievement.

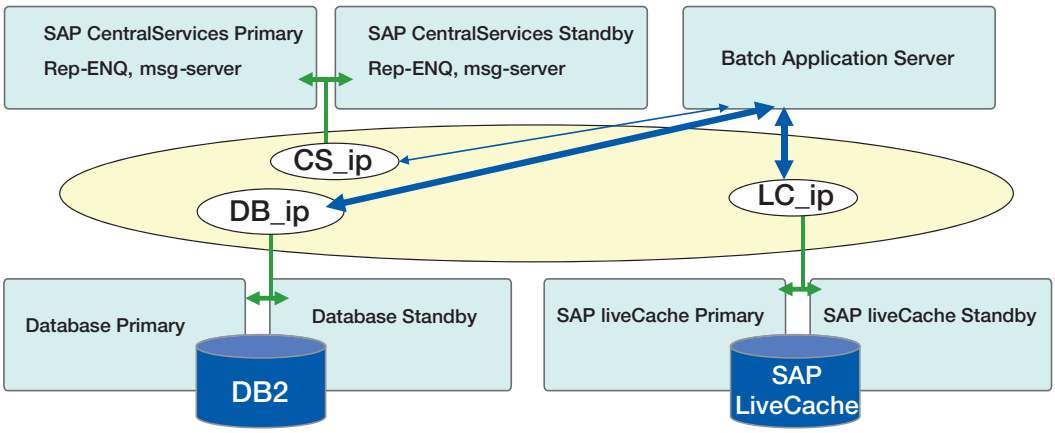


Figure 6.1.1.2: Overview of the cluster communication for demand planning

Key performance indicators (KPIs)[3]

The batch jobs are expected to cancel in the case of a component loss. They should not hang or continue processing with invalid results. They should be in the state to restart processing immediately the failed component has been recovered. Consistency checks should not indicate any inconsistency other than a possible redundant time series. It is acceptable for the KPI that such inconsistencies are found in the time series as a result of the failure. These are time series that have no related information in the database but in no way effect the quality of the results of the planning run. These "orphaned" time series do nothing more than waste a bit of memory. Our reference contacts in production planning operations confirmed that clean up of the time series is done periodically (outside the critical time window) as general practice to recover any memory.

The consistency verification routines were used periodically throughout the project to ensure that no other inconsistencies occurred.

The current KPI for complete recovery in preparation for a restart is less than or equal to five minutes.

Following a failure under demand planning load, the system should be able to restart (with operator intervention or job scheduling) the batch jobs within five minutes with data consistency.

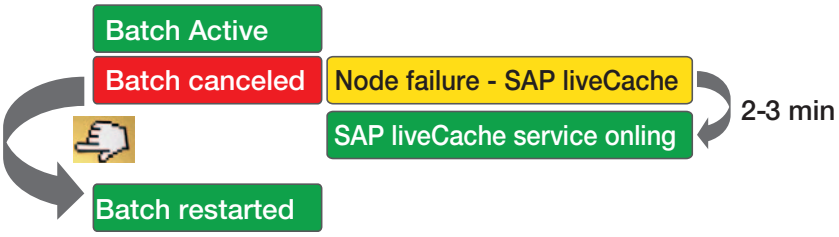


Figure 6.1.1.3: KPI for demand planning recovery

In figure 6.1.1.3, the component failure can be SAP central services, the DB2 HADR database, or SAP liveCache.

3 Indicators used to measure success.

6.1.2 SAP ERP ATP check using SAP SCM

The ATP check in SAP APO is used to ensure that delivery dates can be met before an order is confirmed. ATP functionality exists in SAP ERP as well, but it is not as rich as that provided by SAP APO. ATP in SAP APO provides the following benefits:

- Prevent overcommitment
- Manage backorders
- Enable search in multiple locations (global ATP)
- Automate a manual process
- Reduce the amount of time taken to process an order
- Provide visibility of your sales commitments for every material for which an ATP check is performed, regardless of the results
- Provide additional information to production beyond a forecast
- Allow customers to be prioritized and realign commitments of a constrained product when necessary

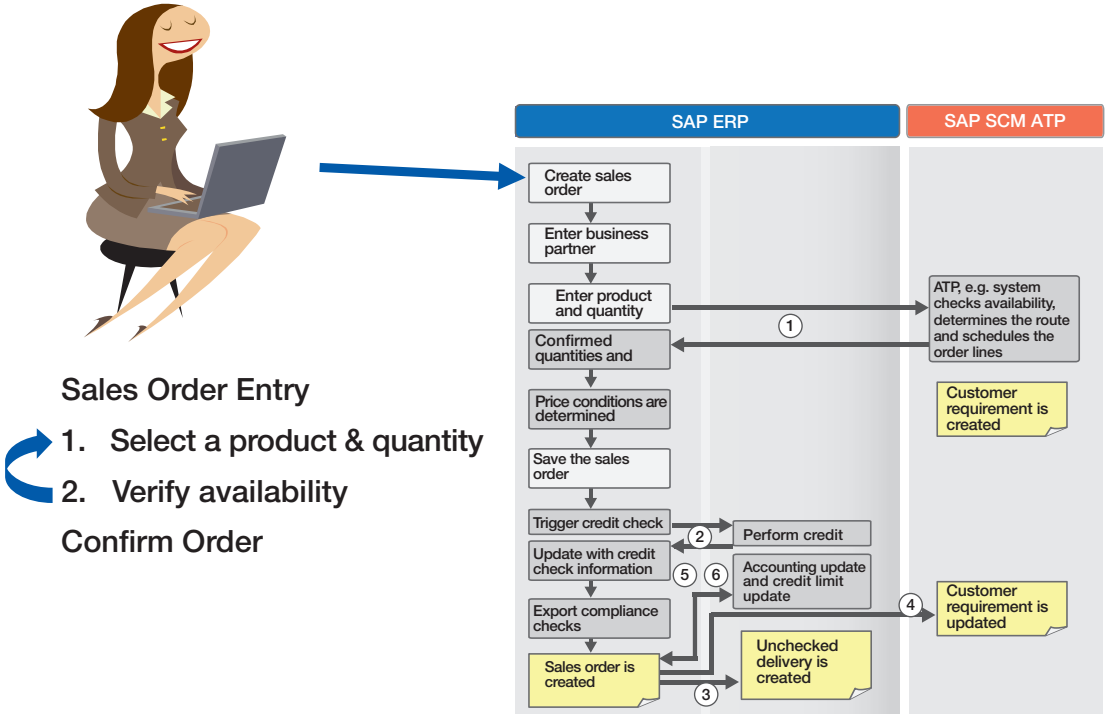


Figure 6.1.2.1: Overview of the SAP ERP order creation process with ATP to SAP SCM

The ATP check for order entry begins in either SAP CRM or SAP ERP and accesses the supply chain management planning of SAP APO through a remote function call (RFC) as part of the transaction. Figure 6.1.2.1 is an overview of the order creation process. An ATP check can be run for each order line of the customer order as necessary, and each check results in an RFC exchange between SAP ERP and SAP APO. This process is shown in more detail in the figure 6.1.2.1, but important to note is that if the SAP APO system is not available, the ATP check cannot be completed with confirmation. Orders can then only be created without the assurance of the ATP confirmation, and without product or materials reservation – this is a major depreciation in quality.

Therefore, one of the major objectives of this proof of concept is to ensure that SAP APO is there, and that this process chain is not broken by a failover of any of the components.

Figure 6.1.2.2 shows the actual multiple logical units of work (LUW) of the business process for ATP. This process spans four LUWs. The first LUW is the driving process which is creating the order and talking to the SAP GUI. The second is the RFC process created on SAP APO and tracked through the transaction identifier (trguid) exchanged between SAP ERP and SAP APO. The third is the booking process that asynchronously updated the database in SAP ERP following the order commit, and the fourth LUW is the asynchronous commit and database updates in SAP APO driven by the SAP ERP commit and transmitted over the queued core interface communication (CIF).

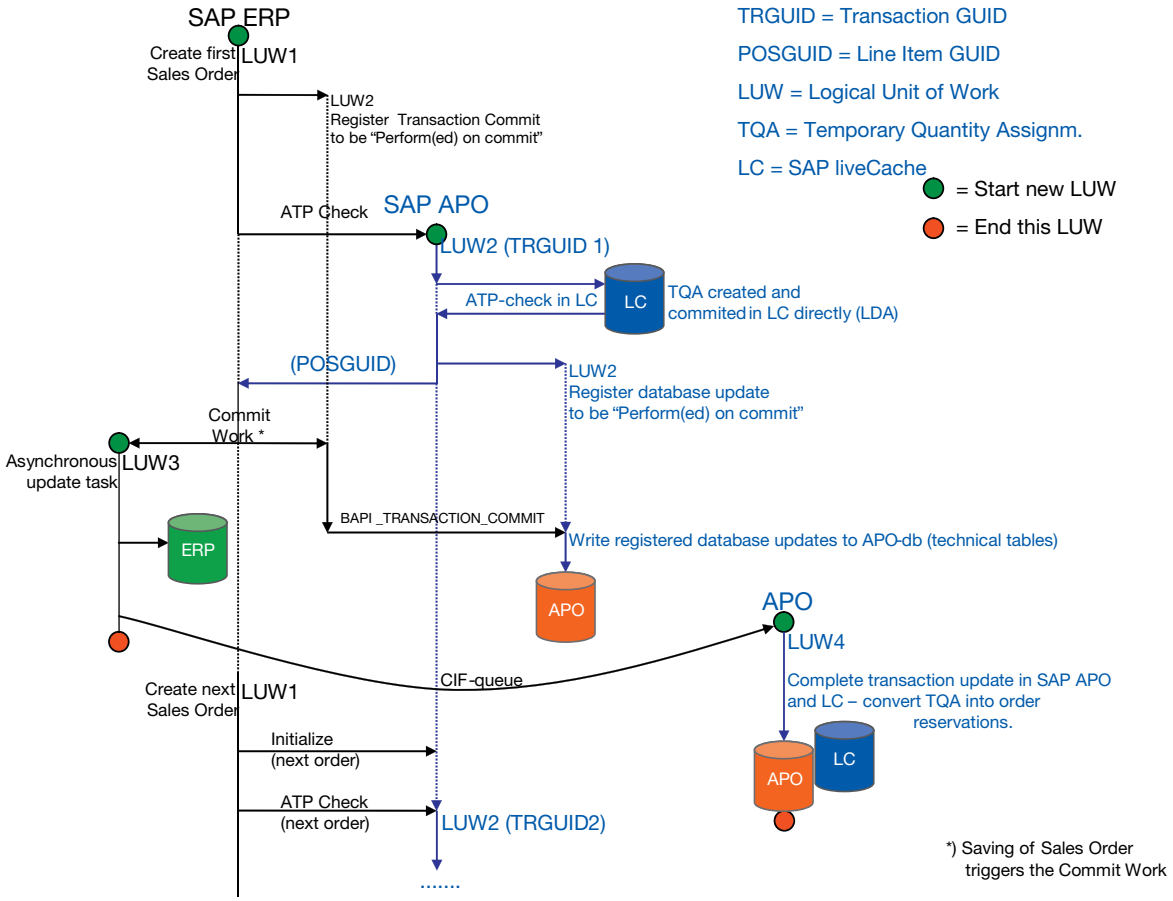


Figure 6.1.2.2: Diagram of recoverable order creation process based on multiple logical units of work

ATP test case

In the proof of concept, the functionality of this order creation business process was the focal point for ATP verification. An order was opened and several line items confirmed through ATP, resulting in temporary quantities being reserved in SAP APO. Prior to the order being commited in the SAP ERP system, a component failure was initiated in the APO infrastructure, and further attempts to continue adding line items to the open order with ATP were made during the failover activity. The target was to see whether the ERP side of the order process would fail or whether is would be able to recover and complete the processing with APO.

KPIs

The ideal result is that the order process recovers and can continue, allowing the order to be completed. An acceptable result might be that the order must be reinitiated after a failure, but that the system must be available again for the next order within the recovery time set by the KPI.

The current KPI for complete recovery is less than or equal to five minutes. This KPI was set by actual customer requirements.

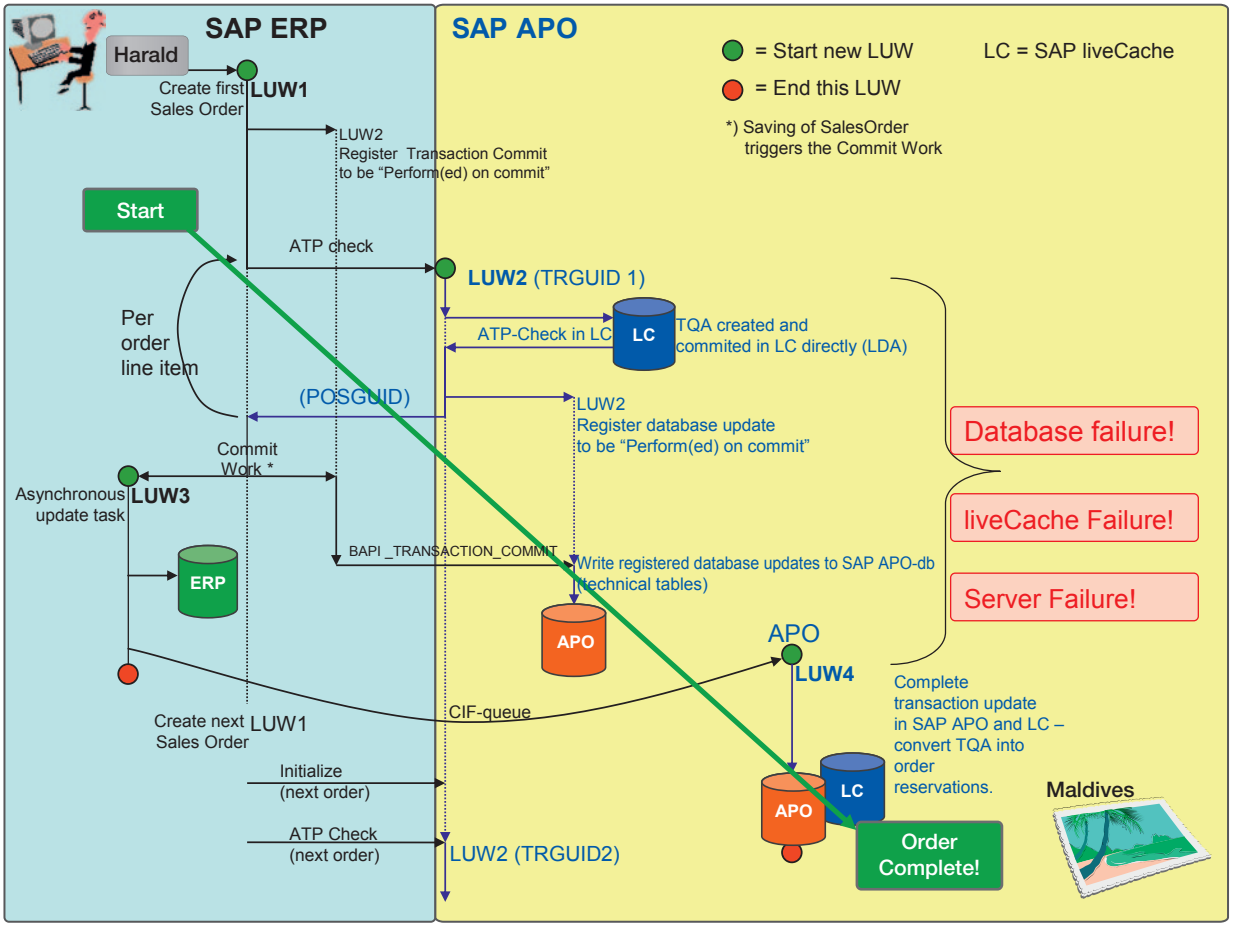


Figure 6.1.2.3: Overview of the order creation process target for recoverability

6.2 Summary of KPI results

The section summarizes the results that were actually achieved in the proof of concept as they relate to the key performance indicator goals. The results are grouped by application target and show the results of each of high-availability cluster components.
The results for the high-availability hardware infrastructure are documented in the hardware section. The hardware infrastructure met its KPIs by demonstrating that failures in redundant hardware components were managed by the infrastructure design and these had no effect at application level – production continued without disruption or need for any recovery activity.

6.2.1 Demand planning under load

Formal PoC KPI: five minutes recovery of service

Failure of HADR database: available in less than or equal to two minutes

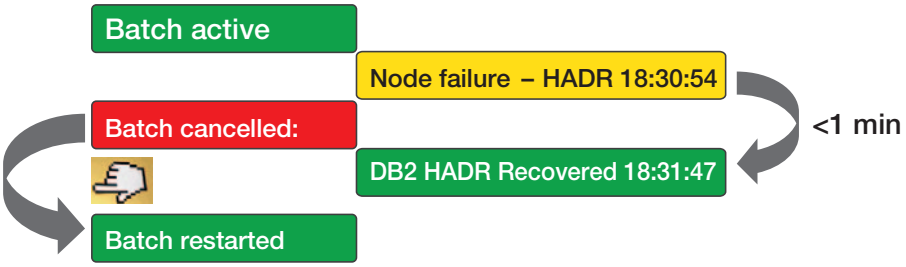


Figure 6.2.1.1: Recovery from DB2 failover after server crash

Failure of SAP liveCache database: available in less than or equal to three minutes

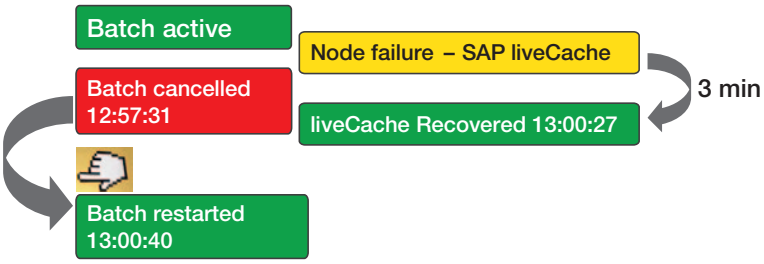


Figure 6.2.1.2: Recovery from SAP liveCache server failure

The System Automation for Multiplatforms (SA MP) supported components (DB2 HADR and SAP CS) showed a faster reaction time then the PowerHA supported component (liveCache) as they are more tightly integrated with the SA MP cluster solution supporting them. In the current PowerHA cluster support, recognition of the loss and failover by the cluster takes somewhat longer then for the SA MP components, but remains well within the KPI. As the objective is reliability, a faster reaction time was not forced. The actual HotStandby takeover activity in SAP liveCache takes from a few seconds up to a minute, depending on the liveCache version. The combined recovery time of liveCache and PowerHA was within the KPI.

Failure of SAP central services: Restored in less than two minutes
Batch jobs in external batch application server do not fail but hang in enqueue wait until enqueue service is restored and then continue to run. The end-to-end runtime of the mass demand planning run (for 896 seconds) was around one minute longer than the normal runtime (for 840 seconds) as result of the failover.

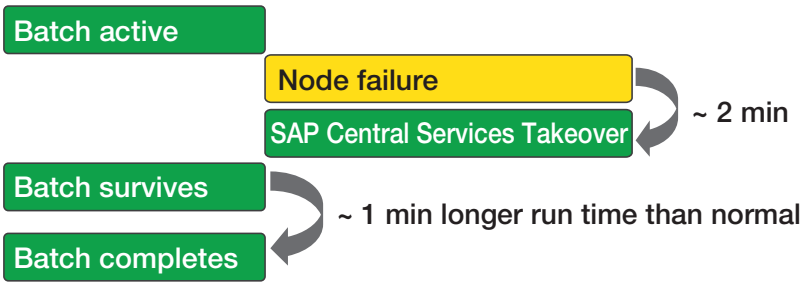


Fig 6.2.1.3: Recovery of SAP message and enqueue server processes after server crash

6.2.2 Available to Promise

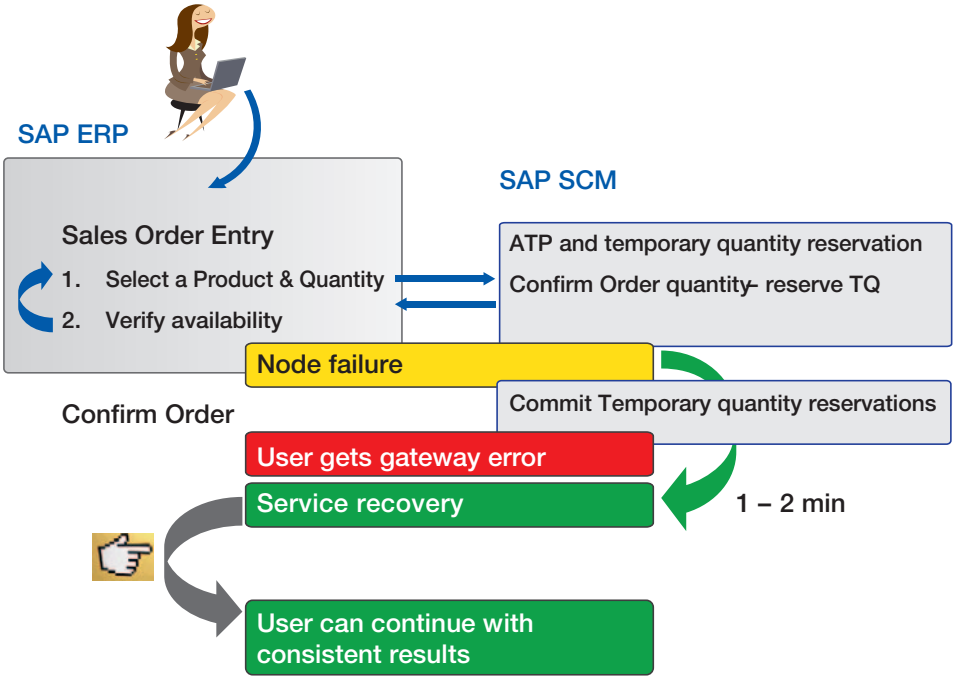


Figure 6.2.2.1: ATP recovery from SCM component failure

KPI: five minutes recover to next order

Current open order is not interrupted but is able to continue correctly to completion despite failover in the SAP APO infrastructure. Data integrity was achieved on both SAP APO and SAP ERP views of the completed order.

Results

Failure of HADR database:	Available in less than or equal to 2.5 minutes
Failure of SAP liveCache database:	Available in less than or equal to 3 minutes
Failure of SAP CS:	Available in less than or equal to 2 minutes

7 Design Components of the Infrastructure

This chapter covers the overall design of the infrastructure used for the proof of concept and explains the purpose of each component design. There are always many different approaches which can be taken to achieve a very similar result. The design presented, therefore, does not pretend to depict the only possible implementation route. In order to make it easier for other implementation teams to modify the design, for reasons dictated by their own requirements, the team has tried to present the options, the choices made, and the reasons for the options selected.

7.1 Infrastructure scope and stack

This section introduces the solution design, and maps the infrastructure components into the end to end design. It explains why the component stack used in the proof of concept was selected. No proof of concept can cover every eventuality and therefore it is important to define what was used, why it was selected, and what the dependencies are.

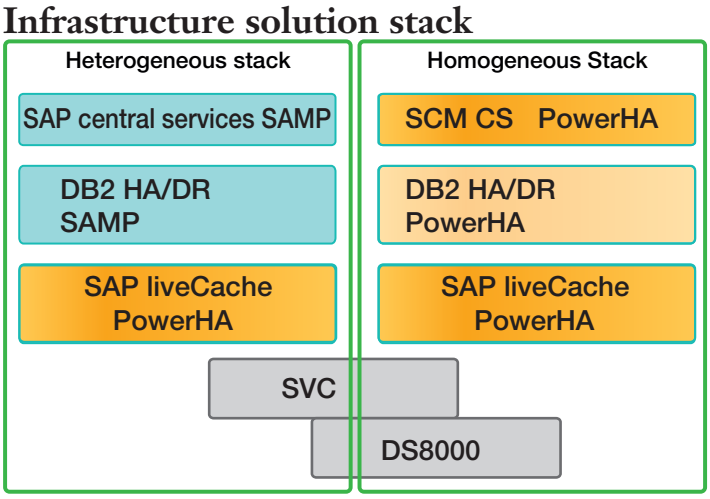


Figure 7.1.1: Overview of component stack

The scope of this proof of concept addresses the cluster component combinations shown in figure 7.1.1. SAP central services (replicated enqueue and message server failover) are supported by both cluster solutions (PowerHA and Tivoli SA MP) and are in production in many sites today.

DB2 HADR and SAP central services are supported by Tivoli System Automation for Multiplatforms in a close integration with SAP. SA MP is available as part of the installation kit for SAP software.

PowerHA or IBM HACMP™ for AIX is a very mature AIX clustering solution that has been a standard in AIX-based SAP implementations for many years. Due to the flexibility of HACMP, and its longevity in the market, there are implementations in production for all SAP components that require high-availability functionality, including SAP central services and DB2 HADR.

The target of this project is to address both cluster functionalities, and thereby support the major customer base, with the exception of SAP liveCache. The SAP liveCache solution is new and is a feature of AIX PowerHA (HACMP). Nevertheless, the SAP liveCache solution can be combined with the other building blocks using System Automation for Multiplatforms to build the complete HA stack in a heterogeneous environment. PowerHA can also be used to implement the entire stack in a homogenous environment.

Both the homogeneous PowerHA stack and the heterogeneous SA MP with PowerHA stack is covered in this proof of concept.

The IBM implementation of SAP liveCache with HotStandby is based on FlashCopy technology and initially supports the IBM storage servers DS8300 and San Volume Controller. The DS8300 is supported using its native interface and functionality directly. For other SAN storage, the SAN Volume Controller (SVC) provides the interface and offers additional flexibility and extended HA functionality. Via the SVC, any available SAN storage can be supported. With this initial combination it is possible to support the majority of the current SAP APO installed base in one way or the other with a high quality solution.

7.1.1 Logical landscape

Figure 7.1.1.1 shows the logical landscape design used for the proof of concept. The high-availability design is focused on the components dedicated to SAP SCM APO. File sharing is typically provided by a highly available Network File Server (NFS functionality) that provides generally for the systems within an SAP landscape rather than a single SAP system. There are a number of different highly available NFS solutions that are commonly in use. For this reason, the proof of concept assumes that an external HA NFS is available and shared by the SAP SCM system. Another common implementation design is to place the NFS file systems under the control of the SAP central services cluster solution.

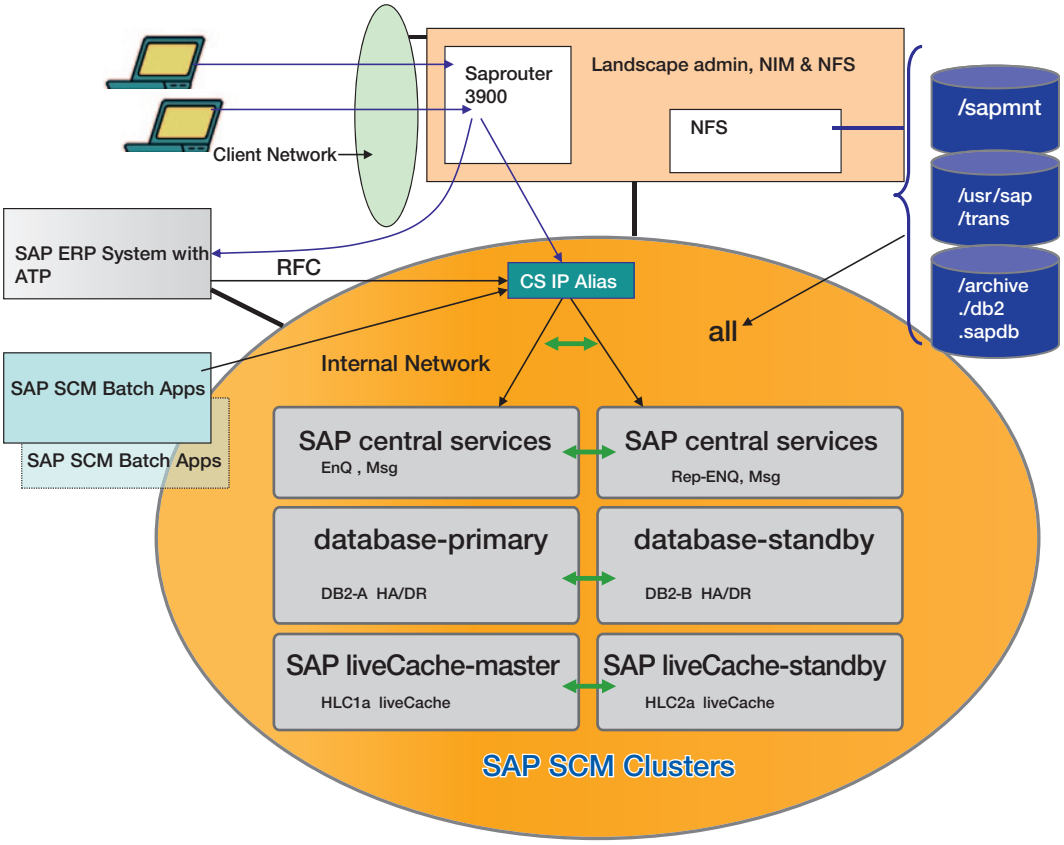


Figure 7.1.1.1: Overview of the logical landscape design with two networks and external NFS used for the proof of concept

Best practices for SAP installations normally see a separation between the client network and the backbone network. The backbone network connects the components of an SAP landscape – application servers to the databases, SAP ERP to SAP APO and so on. The backbone networks are typically high-capacity networks with few hops. Separating the client network from the backbone is also done for security purposes; logged in users cannot directly access a database or a server.

The active SAP message server is reached through the IP Alias for the SAP central services cluster. The access between SAP ERP and SCM is by means of a remote function call (RFC) port, configured to the IP alias.

The SCM batch application server is not part of a cluster as applications servers do not represent single points of failure. Multiple applications servers can be configured for online users and batch. For the proof of concept, a single large batch application server is used as a single server is sufficient to understand the effects of various failures on the batch load.

7.1.2 Server infrastructure

The infrastructure is based on IBM Power® servers using PowerVM to take advantage of the many benefits of this server infrastructure for HA. Both, POWER6 and POWER7 processor-based servers were used in the functional testing to ensure the solution for the current install base as well as for those SAP SCM systems moving to newer hardware. The proof of concept focused on the SAN Volume Controller for storage testing as the SAN Volume Controller provides a wider scope of functionality, and therefore, a broader scope, as well as the broader requirement for testing.

In the proof of concept, the back-end SAN storage was provided by various IBM SAN storage systems including the IBM XIV® Storage System. The SAN Volume Controller was used to provide the FlashCopy functionality and the additional resiliency of storage mirroring across multiple storage servers.

7.2 High-availability infrastructure design on Power servers – the basis

This section describes the design of the hardware infrastructure used to achieve high availability though redundancy of I/O paths. The target is to avoid interruption to the application levels as far as possible, insofar as the hardware design can cover the failure. The infrastructure design is built using PowerVM virtual I/O to enable the actual redundancy in hardware components to be implemented one time and used by all upstream logical partitions (LPARs). This reduces the cost and complexity of the redundant hardware implementation.

7.2.1 Design overview – storage

This design delivers redundant storage access paths such that any component of the access chain can fail without disruption to the application.

Following design rules are used to ensure continuous availability

Physical storage
The actual storage provided through the storage server is configured using redundancy such as RAID, or other protection mechanisms which ensure availability despite the failure of physical disks. The proof of concept used both RAID5 in the enterprise storage servers (IBM System Storage DS8300 systems) and data redundancy provided by the IBM XIV® Storage System. In either case, the failure of a single disk has no impact to the data availability.

Storage server
To prevent data loss in case of the fail of a storage system, the data will be mirrored to two storage systems. This functionality comes with the SAN Volume Controller.

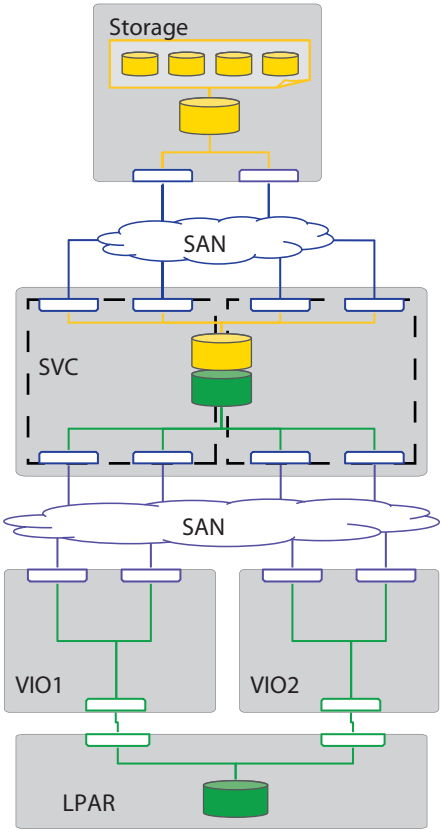
The disk from the storage systems will be grouped into two different managed disk groups (also called MDisk groups). The virtual disks were created as mirrored disks from both MDisk groups, so that the virtual disks are mirrored over two storage systems. In case of a storage system failure, the operation continues with the remaining copies.

The SAN is divided into two fabrics and every storage component is connected to both fabrics. In case one fabric goes down, the traffic continues to flow over the other fabric.

The SAN Volume Controller is a highly-available storage solution because it is a cluster of a minimum of two nodes. In case one node fails, the second node takes over the traffic.

There are two VIO Server partitions on each physical system, which are connected to both of the SAN fabrics. On the client partitions, the AIX integrated multipath I/O device driver takes over the path failover activities.

Parts of the storage layout in this design are:



storage subsystems

SAN fabrics

SAN Volume Controller
(implementing storage virtualization)

VIO Server

LPAR supporting the application

Figure 7.2.1.1: Redundant I/O design for storage

Figure 7.2.1.1 shows the logical stack that connects the logical partitions that house the applications to the storage components that house the data. This stack is implemented using virtual I/O and redundant paths for high availability. The SAN fabric, which appears in two layers in the diagram, will normally be the same SAN fabric, although it need not be.

7.2.2 Native storage design

Design with one storage system
In a native storage design with one storage subsystem, the disk storage is provided by one storage system and the FlashCopy functionality is between LUNs (logical disks) located on a single storage server.

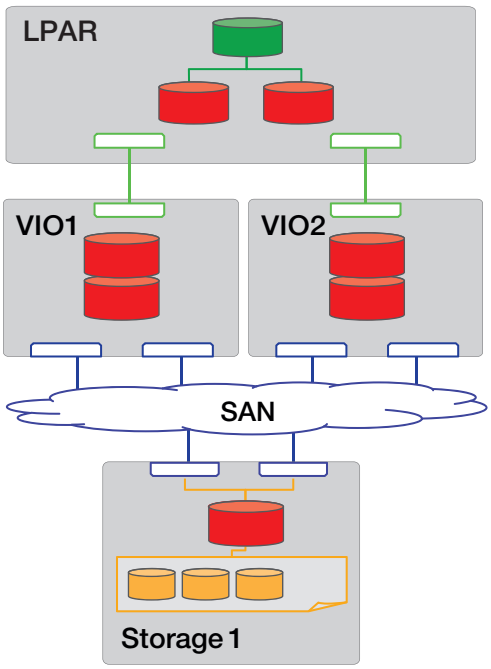


Figure 7.2.2.1: Overview of single storage server access

In a “native storage design” (no SVC storage virtualization layer – disks directly attached to the virtual I/O servers) with two storage servers, the mirroring of the disks from both storage systems could be done at LPAR level with AIX logical volume mirroring (LVM).

This can be done for the databases and other high-availability components, but not recommended for SAP live Cache with Hot Standby. As the data in a Hot Standby is refreshed by means of a storage level FlashCopy, the OS level LVM would not see the refresh happen. Hot Standby, for direct attached disks, is based on FlashCopy within a single storage subsystem. The solution does not span two storage subsystems and is not able to initiate orchestrated FlashCopy over two storage servers as would be necessary. The solution is single storage server, single site only for SAP liveCache with HotStandby, which also means that the storage subsystem becomes a single point of failure. This challenge is solved with the San Volume Controller and storage virtualization.

The disadvantage of a single storage subsystem implementation is that, in case of a storage server failure, both FlashCopy source and target are affected. This would have the greatest impact on the SAP liveCache as the HotStandby uses FlashCopy to generate the data redundancy. DB2 HADR is able to span storage servers at application level.

Design with two storage subsystems

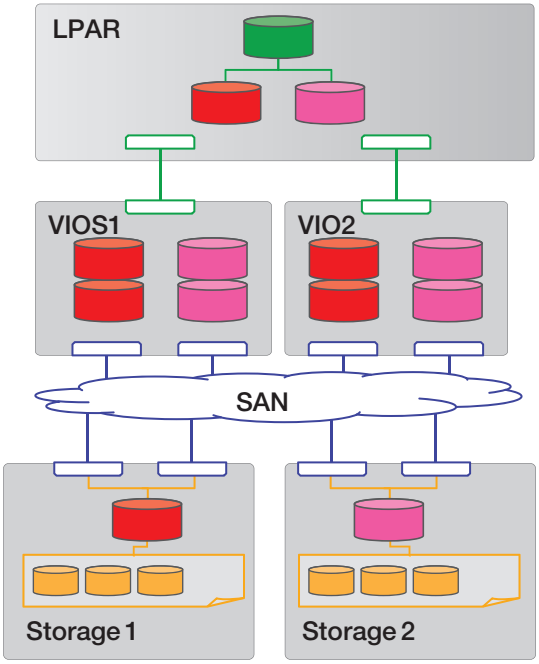


Figure 7.2.2.2: Overview of design spanning storage servers

In a native storage design (without SAN Volume Controller storage virtualization layer), the disks are directly attached to the virtual I/O servers. This will allow data to be mirrored over two storage servers from the LPAR level using AIX Logical Volume Mirroring (LVM).
This can be done for the databases and other HA components, but is not supported for the SAP liveCache with HotStandby. The HotStandby uses FlashCopy, and an implementation that spans multiple storage servers will require FlashCopy functionality that also spans multiple storage servers. The HotStandby solution is not able to initiate an orchestrated FlashCopy over two storage servers as would be necessary. This functionality does not yet exist for direct attached disks, even through the VIOS virtualization layer. Therefore, for directly attached disks, the SAP liveCache solution is a single storage server and single site only with HotStandby. This challenge is solved with the SAN Volume Controller.

7.2.3 Virtualization benefits of SAN Volume Controller

The SAN Volume Controller can attach to multiple storage subsystems and is able to mirror storage over multiple servers. As the SAN Volume Controller is also implemented as cluster for its own redundancy requirements, this solution will allow the critical storage components to be mirrored on multiple storage subsystems in multiple sites.

The disadvantage of a native storage design can be overcome by the additional virtualization layer provided by the SAN Volume Controller.

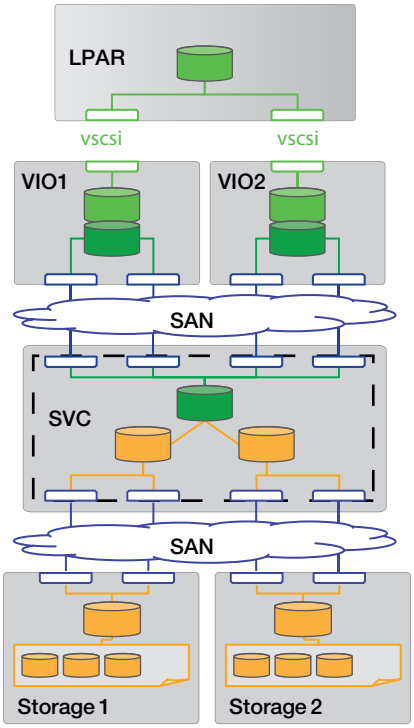


Figure 7.2.3.1 shows storage coming from multiple storage servers. The storage has been organized into storage pools based on storage subsystem boundaries. The virtual disks that are presented to the VIO Servers are created in one storage pool and then mirrored to the second pool such that each copy is on a separate storage subsystem. The failure of a complete storage subsystem should not cause interruption to the application as the data is still available. Root volume groups and other operating system level requirements must also be redundant in order for the LPAR to survive loss of access to a single storage subsystem. In the proof of concept, the operating system volumes were also mirrored across SVC storage pools.

As the SAN Volume Controller is also the focal point for the FlashCopy activity, the FlashCopy will also be reflected in the copies without requiring any effort from the SAP liveCache with HotStandby solution. A FlashCopy from source to target updates the target disk and if this disk is mirrored, then the mirror of the target disk is also updated.

Figure 7.2.3.1: Storage design with VIO and SAN Volume Controller (using virtual SCSI)

The FlashCopy operations are performed with virtual disks so that the handling is easier and storage system independent from the FlashCopy level.

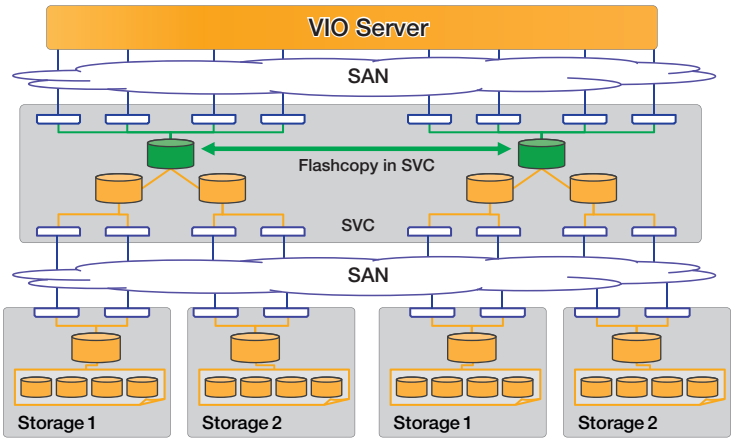


Figure 7.2.3.2: FlashCopy via the SAN Volume Controller

7.2.4 Virtual SCSI versus N Port-ID Virtualization (NPIV)

There are two different methods of distributing the disks from the storage system or the SAN Volume Controller over the VIO Server to the client LPARs.

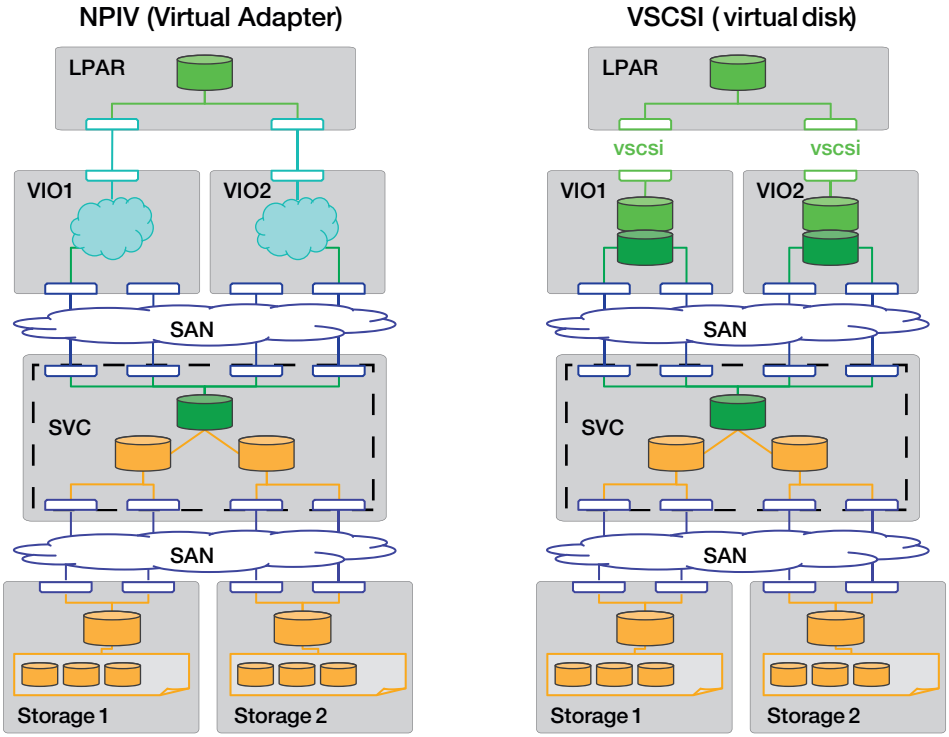


Figure 7.2.4.1: virtual disk or virtual adapter

Until the advent of NPIV capabilities, virtual SCSI (VSCSI) was used. Using VSCSI, the disks from the storage subsystem are attached to the VIO Servers and mapped (using PowerVM functionality) as SCSI disks to the LPARs. In this configuration, the VIO Server provides the binding of the physical disk capacity provided by the storage server to the client LPAR.
AIX Multi-Path I/O (MPIO) drivers on the client LPAR are used to recognize and manage the multiple access paths to each storage LUN. In the case of dual VIO Servers, as depicted in figure 21, there will be two paths to the same VSCSI disk (one per VIOS).

The disadvantage of this design is the relatively complex handling – the mapping that must be configured for each disk. Despite having two paths, access is not load balanced but will select a preferred path.

One benefit of this configuration is that the zoning in the SAN is only between the storage servers and the VIOS. The LPARs do not require specific SAN zoning themselves.

With a new generation of FC controllers, new functionality can be used to implement this assignment of disk to LPAR through VIOS using virtual fibre channel adapters; NPIV rather than VSCSI.

The, N_Port ID Virtualization (NPIV) is a fibre channel functionality allowing multiple N_Port IDs to share a single physical N_Port. This allows multiple fibre channel initiators to occupy a single physical port.

With that the physical SAN can be extended over a virtual SAN and virtual FC adapters can be assigned to the client LPAR. The client LPAR can now directly access the storage subsystem and the disks can be assigned directly to the LPAR.

The advantage of this method is an easier assignment of disks to the client LPAR. All disks assigned to the virtual adapter address can be seen immediately by the LPAR – they do not need to be defined in VIOS. Additionally, this access method implements load balancing over the multiple paths, spreading the I/O over the virtual adapters and thereby over the multiple VIO Servers.

For this solution, the LPARs must participate in the SAN zoning, which increases the zoning complexity to a certain extent.

Using NPIV, the SDDPCM multi-path driver is recommended. The proof of concept on AIX 6.1 used SDDPCM 2.6.03.

7.2.5 Results of storage component failure

To verify the design's ability to meet the application-level requirements, the following tests were done under application load (batch demand planning).

Failure of a VIO Server
The VIO Server LPAR may fail, or more likely, the VIO Server may require scheduled maintenance. This test was used to prove that a VIO Server can be stopped (or fail) without causing any disruption to the application. In this case, the partner VIO Server is expected to assume the load and continue in un-interrupted production.

Result was a short I/O suspend of some seconds for the takeover and no impact on the application.

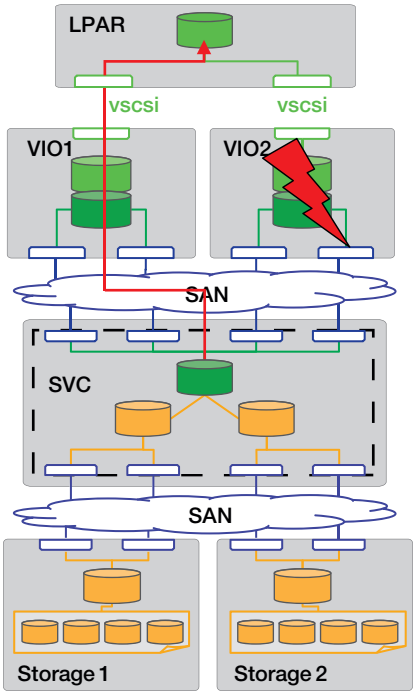
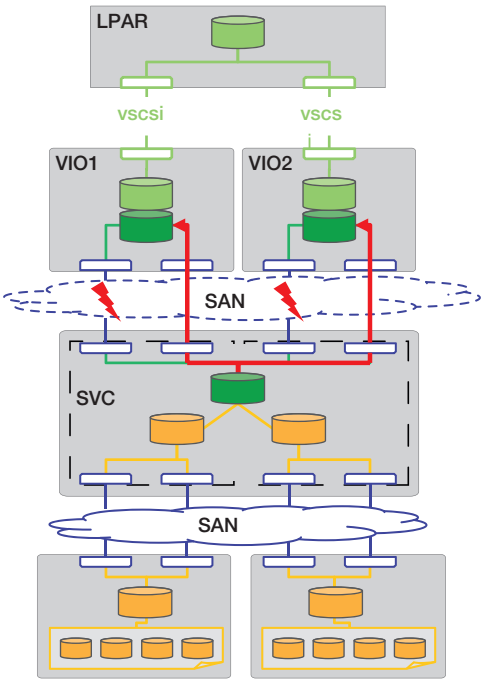


Figure 7.2.5.1: Reaction to the failure of a VIO Server

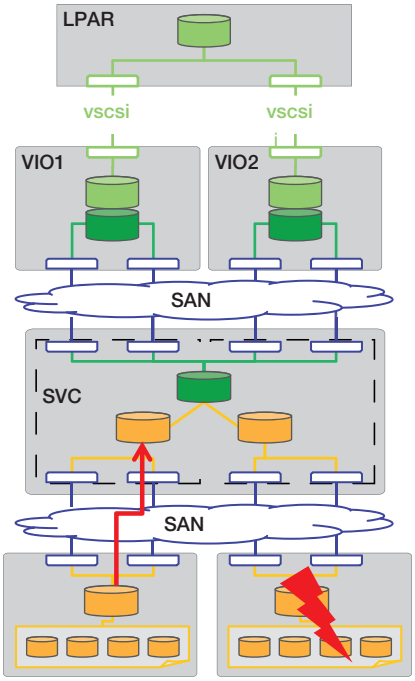


Failure of a SAN fabric or a FC controller

A FC controller, SAN connection, and even a complete SAN switch can fail, or a section of the fabric may be removed for scheduled maintenance purposes. In such cases, the I/O should be rerouted through the redundant path on each of the VIOS and the application should not be affected. The multipath environment should continue production I/O without interruption.

The result for such a failure in the proof of concept was only a short I/O suspend of some seconds, with no impact to the Demand Planning application running in parallel batch.

Figure 7.2.5.2: Reaction to the failure of a SAN switch or fabric component



Failure of a complete storage system

It may be possible that one of the storage systems might fail or might need to be stopped for maintenance purposes.
In this case, the mirroring function of the SAN Volume Controller ensures that the data remains available and production can continue.

Figure 7.2.5.3: Reaction to a storage server failure in a mirrored storage design

7.2.6 Design overview – network

The TCP/IP network design done for the high-availability infrastructure implements the same approach as was done in support of the storage area network. The dual VIO Servers provide redundant paths from the LPARs to the network.

The network is expected to consist of redundant switches, and each network adapter connected to a different switch. With this configuration, a switch, a cable, an adapter and even a VIO Server can fail without disrupting network access for the application.

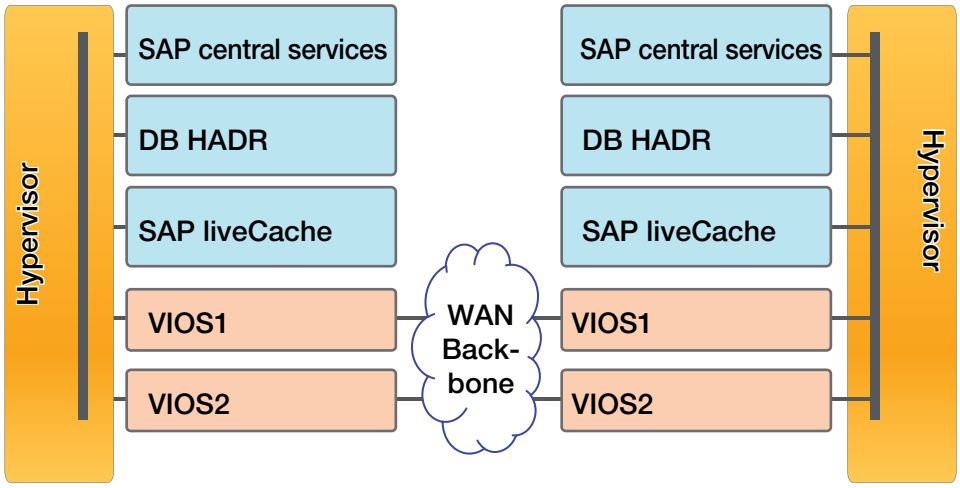


Figure: 7.2.6.1 Overview of the logical network

The network connection for the systems is provided by two networks, one internal (SAP backbone network) and the other external (maintenance network). The maintenance network is protected from external access through a firewall, and is used for administration access and for backup traffic.

Both, the internal and the external networks are routed within the Power systems through the IBM Power Hypervisor as virtual networks. These two virtual networks are connected to the virtual networks on the second Power system over a physical connection to the network switches. In this case, when all primary servers are on one machine, all application traffic is routed through virtual network and the hypervisor.

If a component fails and its service is moved to the standby servers on the second Power system, traffic between the application components will then transverse the physical network as well.

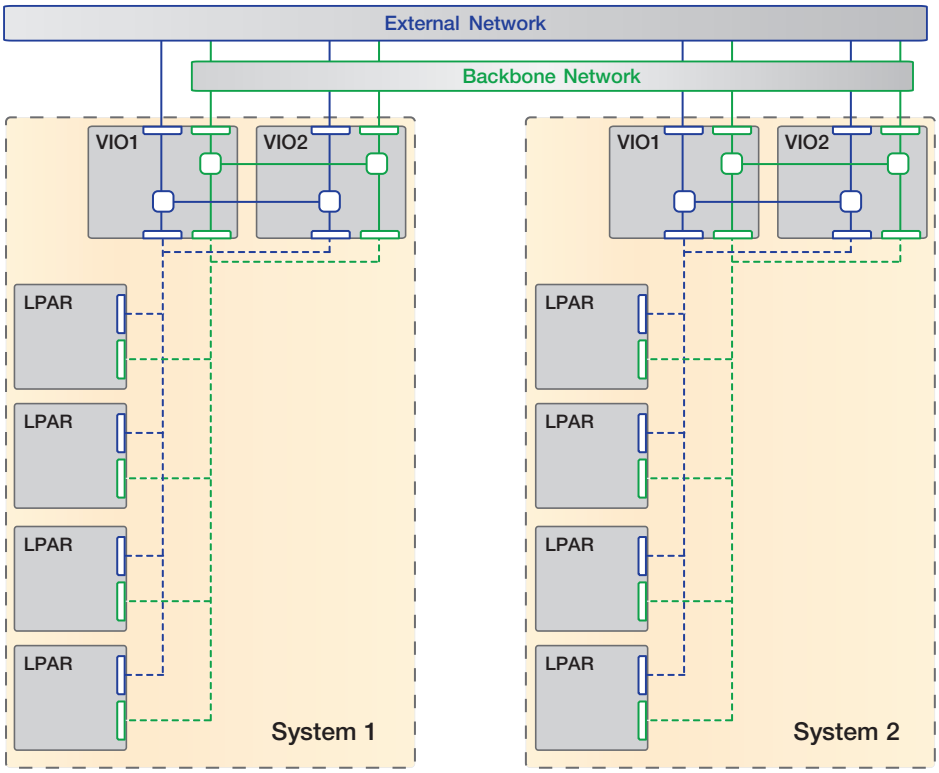


Figure 7.2.6.2: Internal and external network design over Hypervisor

Design for shared Ethernet adapter failover

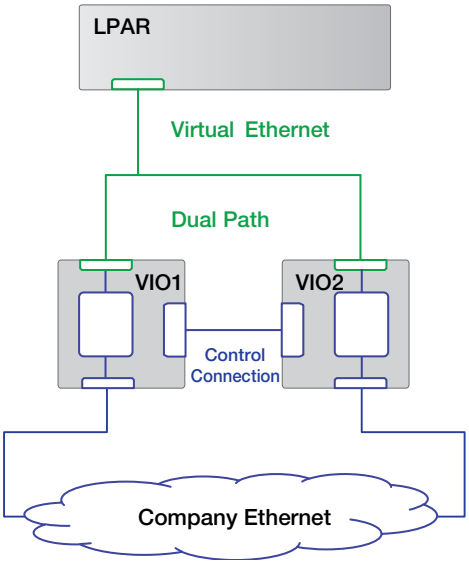


Figure 7.2.6.3: Multiple paths for the virtual Ethernet

The LPAR automatically recognizes a virtual Ethernet adapter in the same way as it a physical Ethernet adapter. In the view of the application levels above, the virtual adapter is a physical adapter. A virtual adapter however has the benefit of providing multiple paths over the VIO Servers and over more than one physical adapter. This virtual Ethernet is connected through both VIO Servers and therefore it has two paths. On each of the VIO Servers there is a bridging device between the virtual and a physical Ethernet. This bridging device, also called a shared ethernet adapter (SEA), is set up as a failover SEA and a control connection is created between the two VIO Servers. The control connection coordinates the failover response.

A priority is assigned to each the SEA, so there is a primary and a secondary SEA. In normal operations, the traffic goes over the primary SEA and in case of a failure; the secondary SEA will take over.

Design with network interface backup in the client partition
An alternative method for an Ethernet connection through two VIO Servers is the network interface backup in the client partition. In this case, two virtual Ethernets are created, one per VIO Server. These two virtual networks both have a connection port to the client LPAR and the LPAR sees two Ethernet interfaces. The client LPAR uses two virtual Ethernet adapters to create a network link that consists of one primary adapter and one backup adapter, the same method as used to bundle physical adapters in the past.

The interface is defined on the network link. If the primary adapter becomes unavailable, the network interface backup switches to the backup adapter. The failover action takes place on the LPAR in this implementation rather than at VIO level. This method makes the network configuration somewhat more complicated, and as also only one network path is used at any given time (no load balancing), it provides no obvious advantage to the SEA failover.

For this reason, the simple SEA solution was selected for the proof of concept.

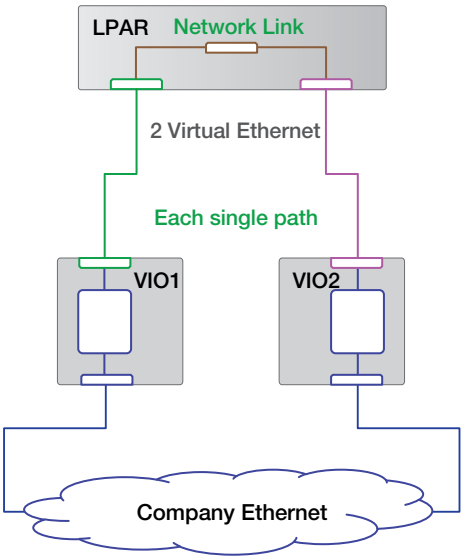


Figure 7.2.6.4: Multiple paths using link aggregation

For the proof of concept it is beneficial that the load is either following one route or the other, but not load sharing over both. This makes the response to a failure easier to track and the shift of load from a failed path to a failover path easier to depict. A consideration for high-availability system, however, would be the effect of the consolidated data volume following a failover. If load balancing were active in the normal state, then a failure of a path would reduce the available capacity by half which may have unpredictable results on the application behavior. The volume would suddenly be all focused over one path. If the system is always working with one path or the other, then a bottleneck in capacity will be visible and need to be addressed, but it will not be hidden. A failover will not change the behavior in any unexpected way, which could be the case when the normal production capacity is suddenly halved.

7.2.7 Results of a network component failure

The following tests executed in the proof of concept to verify the stability of this solution. In each case, the application continued operations without interruption to any of the high-availability cluster pairs, without disruption to the (heartbeats) or between the components of the SAP systems (application server to databases). The 16 batch demand planning load was used for verification.

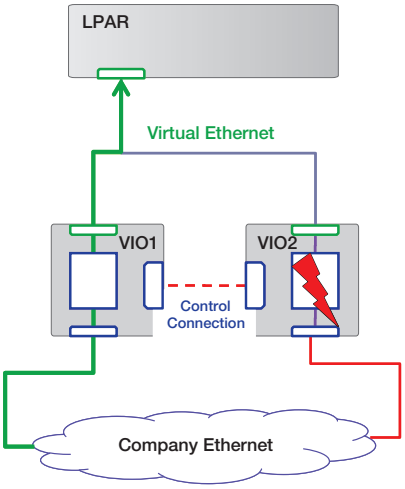


Figure 7.2.7.1: Failure of a VIO Server

Failure of a VIO Server
There are possibilities for VIO Server to fail or a planned downtime to occur.

The SEA of the second VIO Server is informed over the control connection that the partner SEA is not operating and takes over the function.

In these tests, the failover was so fast; we were unable to measure any outage of the network connection. The tools used are limited to one-second intervals.

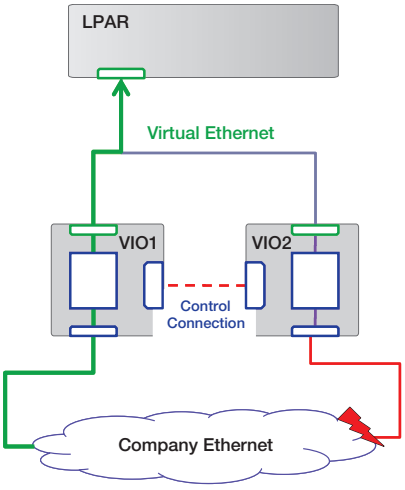


Figure 7.2.7.2: Failure of a network fabric component

Fail of a Physical Network Component
A physical network component, such as a switch or a cable connection of the physical network can fail. In such a case, the SEA of the second VIO Server is informed through the control connection that the SEA on the partner is not functioning and takes over the service.

Test results

The next few graphics show the results from some of the VIO Server failure tests. The red lines show the point of failure and the green lines show the return of the VIO Server. These graphs come from NMON[4] measurements done for a time interval of five seconds. These graphs in figure 7.2.7.3 show the view of ongoing activities as seen from the HADR primary database LPAR.

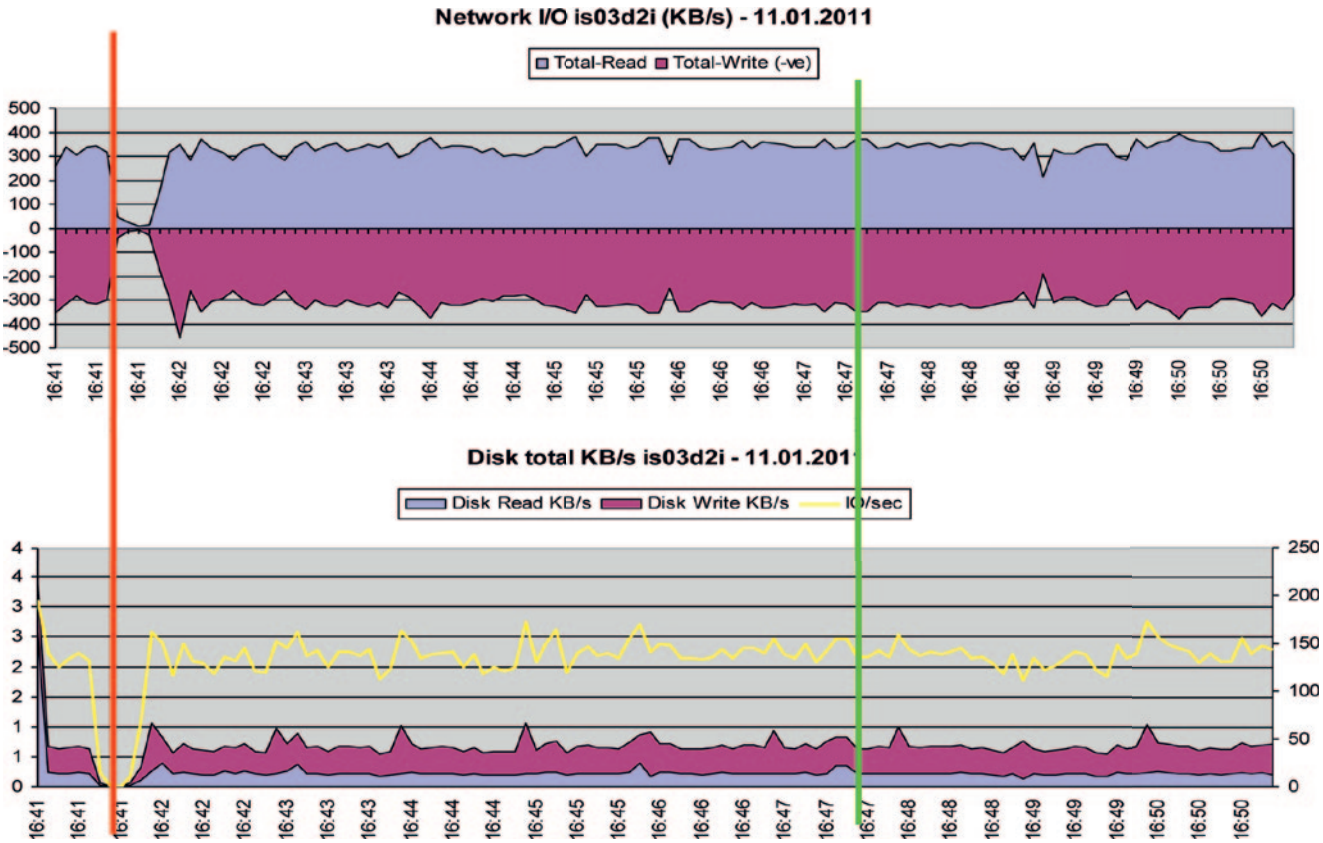


Figure 7.2.7.3: NMON views of I/O component failover

Failure of a VIO Server with the active network connection

The disk activity graph, in figure 7.2.7.3, shows that the disk I/O stalled for a few seconds after the failure, before I/O service was recovered. There was no interruption to application production which was running demand planning batch jobs.

The network traffic (network I/O graph) drops because this traffic is driven by the application and if the application is waiting on disk I/O, it does not generate network traffic. However, the network traffic did not actually stop at any time.

4 NMON is a performance measurement and recording took that is native on AIX. It records a large number of system metrics at set intervals and provides an spreadsheet analysis tool. One highlight of NMON is its ability to graph multiple metrics across the same time axis.

Failure of a SAN fabric in the NPIV environment

The proof of concept tested the high-availability design for both virtual SCSI and for NPIV. Figure 7.2.7.4 shows NMON graphs taken from a SAN fabric failure when using NPIV for disk I/O. The first two graphs are a view from the DB2 HADR primary database LPAR showing a similar behavior as seen in the virtual SCSI environment for a VIO failure. The first graph, disk total, shows that a short suspend occurs in disk I/O during failover. The second graph, network I/O, shows that the impact on the disk I/O is seen as a visible drop in network activity.

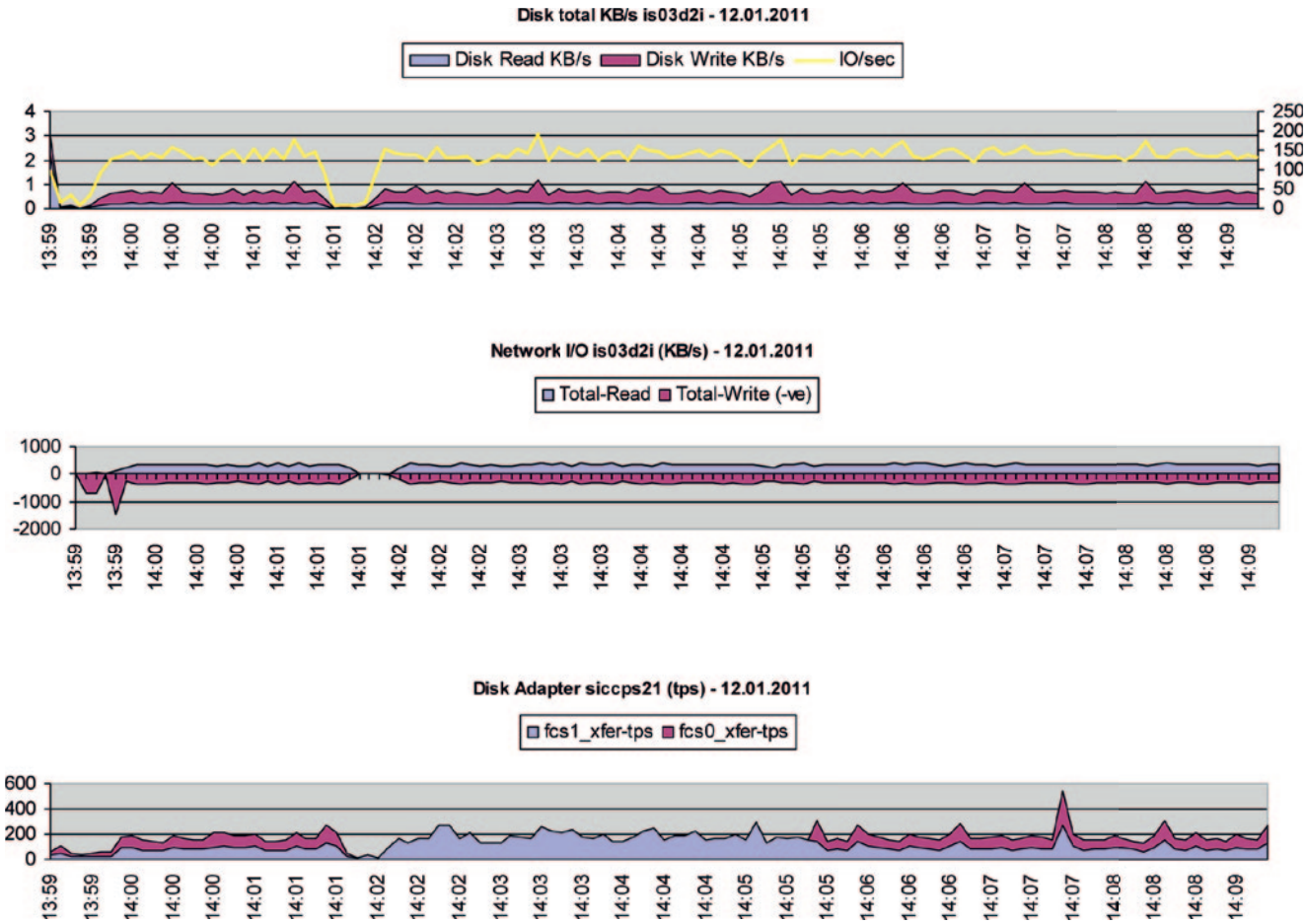


Figure 7.2.7.4: VIO Server failure results with NPIV – as seen from NMON

The final graph in figure 7.2.7.4 shows the view from a VIO Server with NPIV. This represents a SAN fabric failure (actually the device cable to the SAN was disconnected). Prior to the failure, the VIO Server is distributing load over both the NPIV adapters using load balancing. At the point of failure, after a short delay, the total traffic level is seen to have switched to the remaining active NPIV adapter path. The I/O load remains the same, it is simply all routed over the single remaining path from this VIO Server. After restoring the failed component, (replacing the cable), the traffic is automatically resumed over both adapter paths.

7.2.8 Lessons learned during the implementation

Incorrect disk attributes for VSCSI disk

If you have set up disk access through two VIO Servers using VSCSI MPIO to an AIX LPAR, then you need to make some changes to your hdisks because the operating systems sets up the MPIO incorrectly.

With the default settings, if the paths do not automatically reconnect through the second route following a VIO failure, ensure that the hcheck_interval and hcheck_mode are set correctly: Example for default hdisk0 settings:

 # lsattr -El hdisk0

PCM	*PCM/friend/vscsi*	*Path Control Module*	*False*
algorithm	*fail_over*	*Algorithm*	*True*
hcheck_cmd	*test_unit_rdy*	*Health Check Command*	*True*
hcheck_interval	***60***	*Health Check Interval*	*True*
hcheck_mode	***nonactive***	*Health Check Mode*	*True*
max_transfer	*0x40000*	*Maximum TRANSFER Size*	*True*
pvid	*00cd1e7cb226343b0000000000000000*	*Physical volume identifier*	*False*
queue_depth	*3*	*Queue DEPTH*	*True*
reserve_policy	*no_reserve*	*Reserve Policy*	*True*

IBM recommends a value of 60 for check_interval and hcheck_mode should be set to **nonactive**. Normally nonactive is the default but this setting should be verified.

To change these values (if necessary):

 # chdev -l hdisk0 -a hcheck_interval=60 -P
 # chdev -l hdisk0 -a hcheck_mode=nonactive –P

You need to reboot for automatic path recovery to take effect.

If the check_interval and hcheck_mode are not set as described, or no reboot has been done since the change, you are likely to experience the following error even after the failed path is back online:

lspath

Enabled hdisk0 vscsi0
Failed hdisk0 vscsi1

In this case the situation has to be fixed manually using the following commands:

 # chpath -l hdisk0 -p vscsi1 -s disable
 # chpath -l hdisk0 -p vscsi1 -s enable

Rechecking the status now should show:

lspath
Enabled hdisk0 vscsi0
Enabled hdisk0 vscsi1

Duplicate WWN for NPIV

When a NPIV adapter is created, the system assigns two worldwide network (WWN) addresses to this adapter.

You can see the both WWNs on the attributes page of the virtual FC adapter on the Hardware Management Console (HMC).

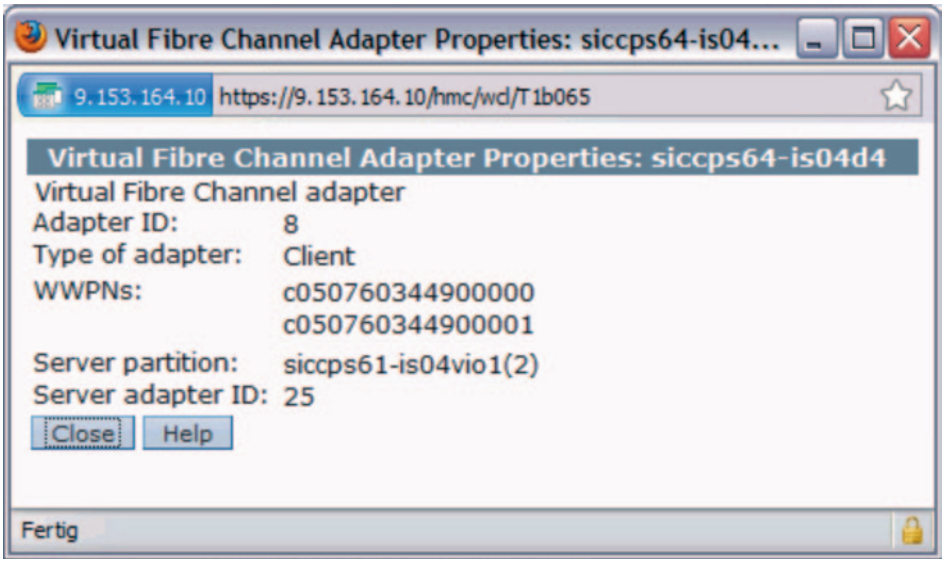


Figure 7.2.8.1: HMC view of NPIV properties

On the SAN infrastructure only one of the WWNs of the adapter is visible and active.
The second WWN is needed for live partition mobility (LPM)[5]. On the destination LPAR, the second WWN will be activated during a LPM operation.

If LPM is in plan, it is necessary that the second invisible WWN is also defined in the SAN zoning between the host and storage subsystem. Otherwise LPM will not work correctly.

5 IBM PowerVM provides the functionality to move an active LPAR between Power servers. This functionality is based on VIO and is controlled by the hardware management console (HMC). The LPAR being moved must take its I/O access with it and therefore virtual adapters must also support mobility.

Disk access failures on VIO Server

There are two attributes that should be set to allow both VIO Servers to access disks that are assigned to both. Otherwise access from the second VIO Server will result in errors.

The following attributes should be set, per hdisk, to enable access from both VIO Servers: reserve_policy=no_reserve and algorithm=round_robin

The attributes can be set as follows:

chdev -dev hdisk0 -attr reserve_policy=no_reserve
chdev -dev hdisk0 -attr algorithm=round_robin

7.2.9 A note on network and domain name server (DNS)

In a highly reactive cluster environment, the domain name server can become a single point of failure. As the IP information is critical to the cluster success, and a long DNS search can cause problems with the cluster behavior, the cluster IP information is maintained locally. The DNS is configured to first use the local and then follow the DNS search path.

7.3 Tivoli clustering – Tivoli System Automation for MultiPlatforms

This section covers the functionality and design overview of the Tivoli clustering software used for both DB2 HADR and the SAP central services in the proof of concept.

7.3.1 Introduction to Tivoli System Automation for Multiplatforms

The minimal target for a clustering solution in this mission-critical SAP SCM software was a recovery time of less than five minutes for the service supported for any component, and application data consistency in the case of database failover. The methodology used in this proof of concept was to go beyond the functionality tests, and rigorously test the solution behavior under typical application load.

In this proof of concept, using the general best practices, the test team easily achieved recovery times that range from one to two minutes under load conditions (massive parallel demand planning) and have proven data consistency after failover for both batch demand planning and the cross-system transactional ATP checks.

IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) was used as the clustering software and IBM DB2 HADR was the database cluster solution providing an ultrafast failover. What made Tivoli SA MP so compelling in our project was the "out-of-the-box"[6] integration with DB2 HADR on the one hand and the predefined polices for SAP central services on the other. SAP and IBM DB2 development teams work closely together to enhance the integration of DB2 HADR/Tivoli SA MP and SAP software to make this an easy to use high-availability solution. With the SAP cluster setup tool (SAP Note 960843) for the database cluster and the Tivoli SA MP policies for the SAP central services cluster, the test team was able to set up and customize the cluster in a very short time.

Main components

The main components of the clustering software were Tivoli SA MP and the reliable scalable cluster technology (RSCT) software products from AIX. Tivoli SA MP is built on top of the infrastructure that RSCT provides on the operating system level to automate the switching of users, applications, and databases. Tivoli SA MP was used to automate the takeover of the database cluster by switching the role of primary DB2 HADR database from Host A to Host B, including the relocation of the service IP address. The service IP address is the access point for the SAP disp+work processes to access the database. It is therefore essential that this IP address follows the active database instance during a failover.

On the application side an additional layer of high availability was introduced: the DB2 replication feature (HADR), for an ultrafast database takeover. HADR does this with an active /semi-active concept of two identical databases that are kept in synchronization through log record shipping, and both the databases are online at the same time. One is the primary and the other the standby database.

6 Tivoli SA MP support for DB2 HADR, with installation methods and supported policies, is integrated into the installation kits for SAP software and delivered as part of the product under an OEM agreement.

From the network side all internal cluster communication, such as IP heartbeat, network tiebreaker, and log record shipping have been routed through the internal network to shield the database from unauthorized access by end users.

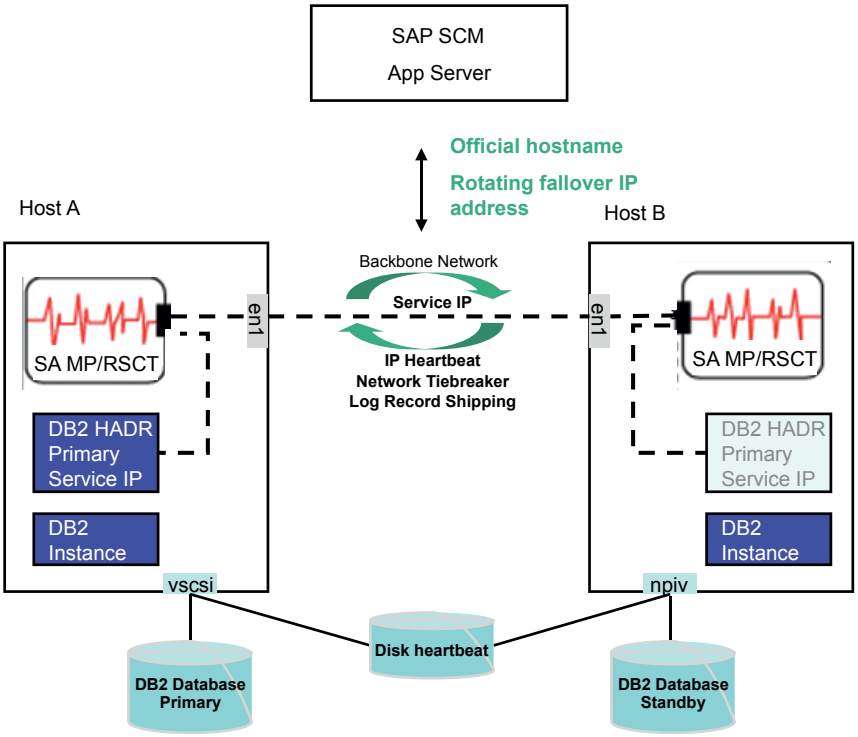


Figure 7.3.1.1: Components of the database cluster

7.4 Tivoli clustering concept for ABAP Database

This section describes the functionality of the highly available DB2 database and how it benefits the end-to-end solution design.

7.4.1 IBM DB2 HADR

During implementation, the test team used IBM DB2 for AIX with the replication feature: HADR provides a high-availability solution for both, partial and complete site failures. SAP customers that purchased IBM DB2 from SAP via OEM[7] can use this feature at no cost as it is an integral part of the database engine, which provides greater protection and higher availability then a traditional failover. Each of the database instances has their own data and logs – there is no data sharing. This is an additional benefit for high availability. Another important fact is that the HADR cluster can span two sites for a disaster recovery scenario.

7 OEM = original equipement manufacturer

Without the HADR feature, the database high-availability design would be based on a mechanism to move the database to the surviving side, implying a cold start of the database which will increase the overall recovery times (traditional failover scenario). Whereas with HADR the standby database is already active, all database buffers are already filled, thus providing an extremely quick failover.

The primary server is the location of the source database and provides the active database service. As transactions are processed on the source database server, database log records are automatically shipped to the secondary server. The test team cloned the standby database from the source database through an offline backup or a FlashCopy through the SAN Volume Controller level. When HADR is started, log records are captured on the primary database and sent to the secondary database. After receiving, they are replayed on the secondary database. Through continuous replay of the log records, the secondary database keeps an in-sync replica of the primary database and acts as a standby database.

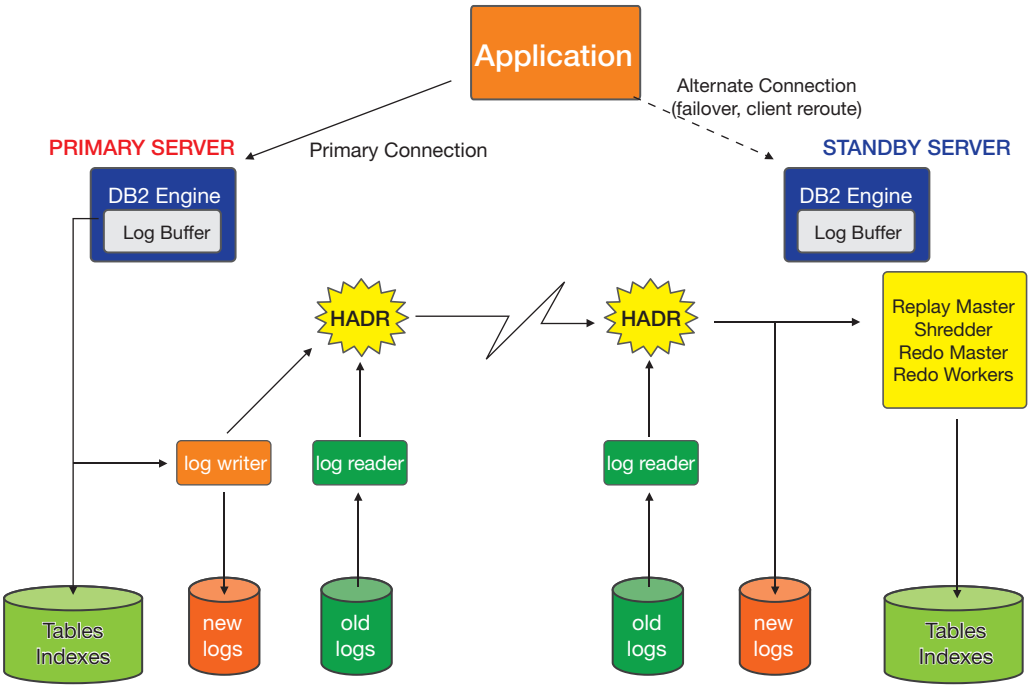


Figure 7.4.1.1: IBM DB2 HADR principles

Level of data protection

HADR offers three levels of protection to prevent potential loss of data:

- Synchronous mode
- Near-synchronous mode
- Asynchronous mode

Synchronous mode offers the best protection of data at the expense of performance. As the commit is not written before, the primary receives acknowledgment that the secondary has applied the log records. The asynchronous mode gives less protection, and in this mode, the log write and send actions are performed in parallel, but the primary does not wait for an acknowledgment from the standby. Therefore network delay is not an issue. This is the reason why this mode is well suited for WAN application. SAP and IBM recommend using the near-synchronous mode

as it combines the advantages of both by providing nearly the same data protection as the synchronous mode and the overhead on the network is minimal. In the near-synchronous mode, the primary writes and sends log pages in parallel and waits for an acknowledgement from the standby before issuing the commit. The standby then sends the acknowledgement after applying the log records to the log buffer.

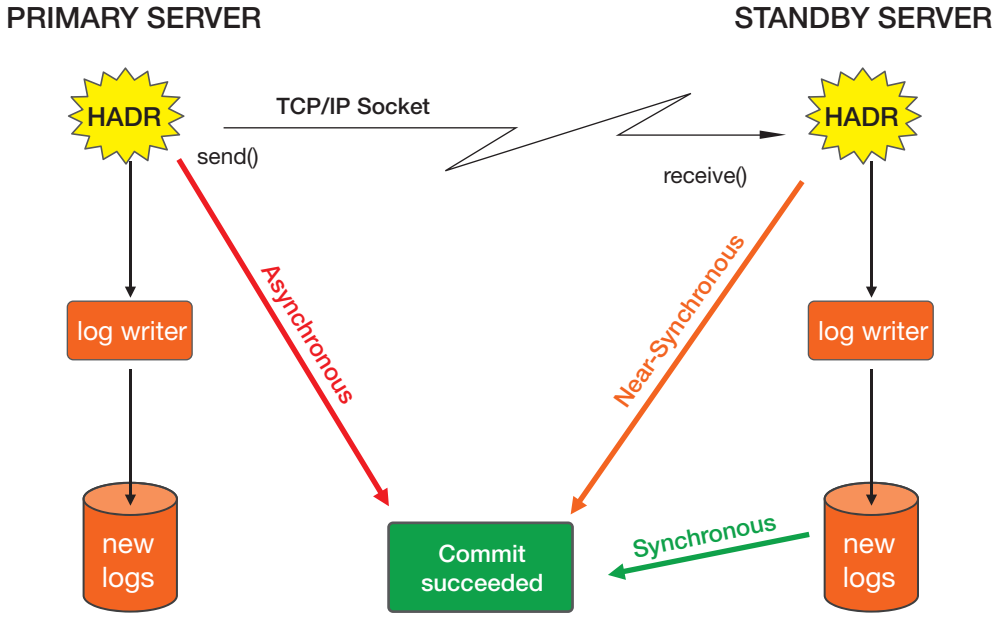


Figure 7.4.1.2: HADR Synchronization modes

IBM DB2 high-availability feature

With the SAP cluster setup tool, sapdb2cluster.sh the DB2 high-availability feature can easily be deployed in an SAP DB2 HADR environment using IBM Tivoli SA MP. Through the strong integration of Tivoli SA MP with DB2, the test team was are able to manage the database cluster either with Tivoli SA MP cluster or with native DB2 HADR commands, such as db2 takeover, db2 start hadr, or stop hadr. This is a big relief in daily operation for the DB2 database administrators who are more familiar with the DB2 syntax. Due to the tight integration of DB2 HADR with the SAP software, SAP customers also the option to start and stop DB2/HADR by using the startsap and stopsap commands instead. This provides further simplification of operating the HADR cluster. The SAP script handles the start and stop sequence for the HADR cluster under the covers.

Hardware and software requirements

The following prerequisites are necessary for implementing IBM DB2 HADR:

a) Same database name for primary and standby database (SID)
b) Same operating system and DB2 levels. (This rule can be violated for the DB2 level/OS system for a short time during a rolling upgrade.)
c) High-speed backbone IP network between database nodes appropriate to the application load
d) Identical table spaces and containers

To accelerate the catch-up process, the log archive device for the primary and standby databases were shared by using NFS mounts for the archive logs, in addition to the SAP shared directories /usr/sap/trans and /sapmnt. This allows a single high-capacity network connection for NFS to serve the SAP needs, the archive needs, and the recovery needs.

In an HADR environment, both the database servers need the same hardware resource for optimal performance. The proof of concept used Power virtualization methods to prioritize the processor resource distribution. This priority schema ensures that the necessary resources are dynamically redistributed according to requirements and the role of the components in the SAP SCM system. A cluster component will have very high priority, and can be sure that the processor resources it needs for recovery will be available within 10 milliseconds, as nothing has higher priority than recovery of the production services.

In this way the test team avoided the congested HADR state. Congestion occurs when the standby is unable to keep up with the transaction load generated by the primary, which causes the standby to fall behind or the performance of the primary to degrade.

The graphs in figure 7.4.1.3 and figure 7.4.1.4 depict the number of commits written by the application on the primary and the log gap (in kilobyte) between the primary and the standby databases resulting from a demand planning batch run. The demand planning application issues commits on a frequent basis which forces the primary to write the log records to disk and send them to the secondary in parallel (near-synchronous mode). The frequent commit rate keeps the log gap small. The log gap shows the difference between the primary log sequence number (LSN) and the standby log LSN. The relatively low log gap and the fast takeover times are an indicator that the standby was capable to apply the log files fast enough to keep this gap very small. The test team has not seen any evidence that the commits are slowing down on the primary which would have been the case if the standby was not capable of keeping pace in applying the log file.

Another aspect which arises in failover situation is the shift in processing capacity requirements. The new primary database needs adequate resources to service the client applications – more resources than it required as a standby. During the tests, the test team did not encounter any signs of congested state or performance degradation when moving from one to the other database server. In this implementation, PowerVM was used to ensure the necessary availability of processor capacity.

During the demand planning batch runs, the standby consumed around 50 percent less processor power then the primary, and has nearly the same memory requirements (12 percent less than the primary). The log record shipping generates additional load over the network, therefore network throughput is another key performance focal area in an HADR implementation. The Gbit Ethernet in the proof of concept scenario was sufficient for throughput/performance.

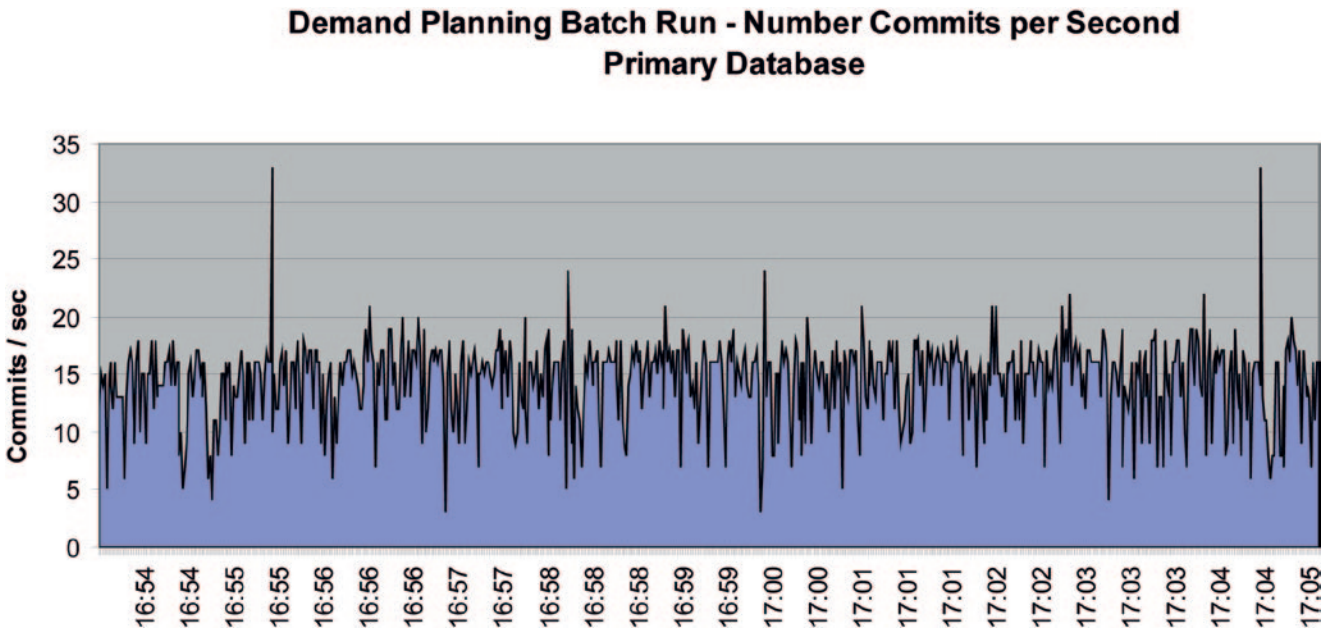


Figure 7.4.1.3: Primary database number of commits during Demand Planning batch run

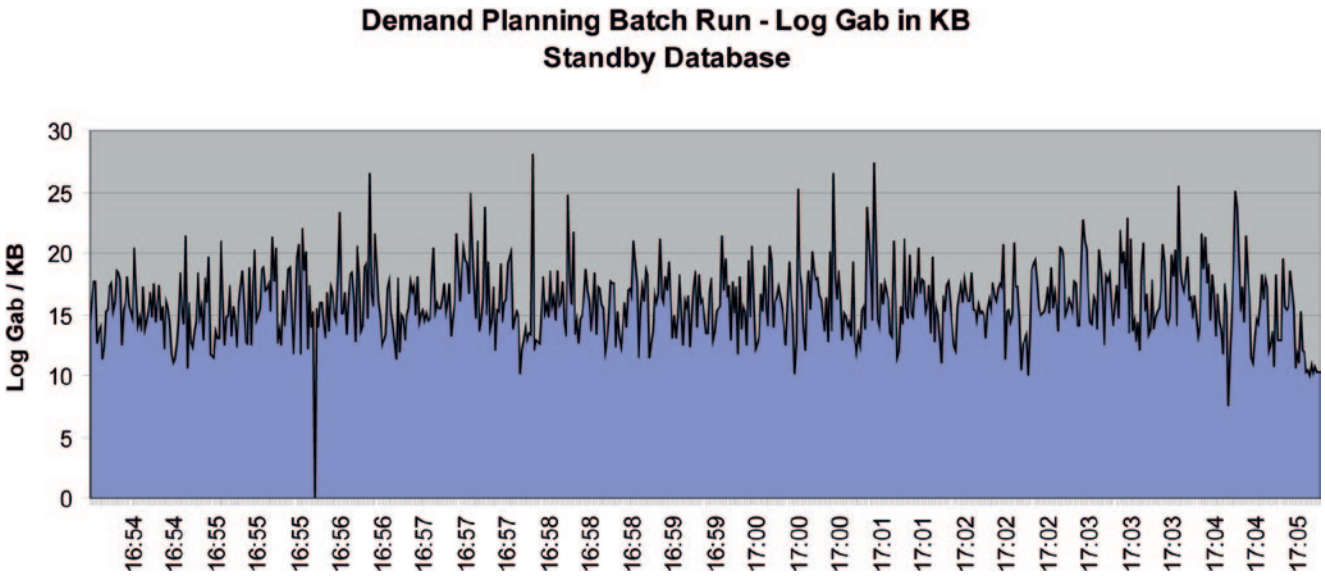


Figure 7.4.1.4: Secondary database log gap in KB during Demand Planning batch run

7.4.2 Automate IBM DB2 HADR cluster with Tivoli SA MP

With IBM DB2 HADR alone it is not possible to automate the failover of the database or the migration of the service IP address. This is where Tivoli SA MP comes into play. Tivoli automates the failover and offers cluster support. In setting up the cluster automation the test team used the SAP cluster setup tool, sapdb2cluster.sh, which defines the SA MP resources as well as setting up the HADR definitions for the database by invoking the DB2 high-availability instance configuration utility (db2haicu).

This section describes in more details how DB2 HADR was implemented with Tivoli SA MP based on a best practices. Figure 7.4.2.1 illustrates the cluster layout.

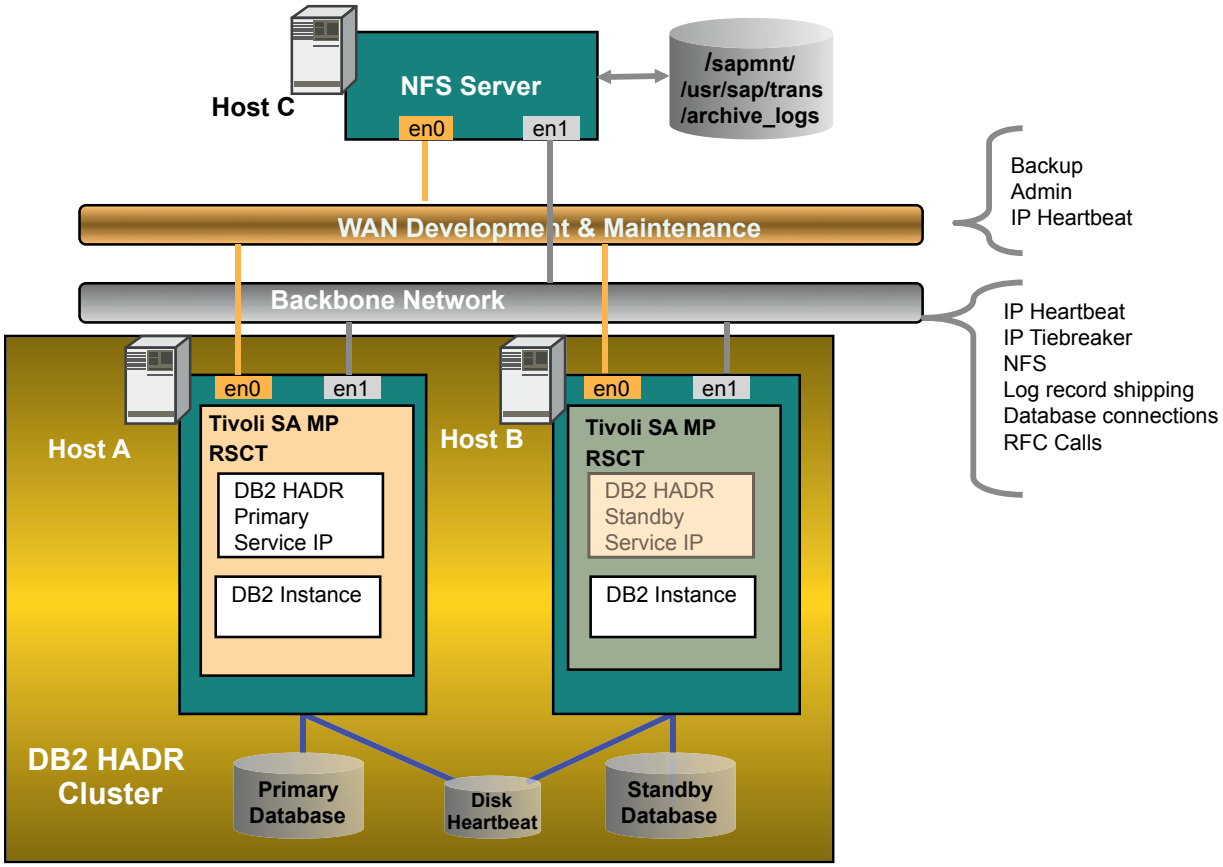


Figure 7.4.2.1: Layout – IBM DB2 HADR – Tivoli SA MP Cluster

Two-node scenario

Setup of a two-node cluster required the use of a tiebreaker to avoid a split brain situation. The following section describes the concept of a tiebreaker and the reason for the decision to use a network tiebreaker together with disk heartbeat functionality.

Network tiebreaker and disk heartbeat

An even number of nodes (two nodes) implementation always requires a tiebreaker to solve a split-brain situation. A split brain can occur when all private links go down simultaneously, but the nodes still remain active. If this happens, the cluster service cannot detect the problem, it might be that the either the nodes or the network is down. In this situation, to prevent each node in the cluster staring services that the other node is still running, a tiebreaker resource will be called which can help decide which node is allowed to run critical resources and gains the operational quorum. If the node with service IP loses the competition for the tiebreaker, it will be immediately rebooted, and the resource will be moved to the winning node. There are four main groups of tie breakers in Tivoli SA MP: disk tiebreaker (most secure), network tiebreaker (easy to implement), operator tiebreaker (manual intervention by an operator) and an additional node (hardware/software/maintenance overhead).

The test team wanted to combine the diagnosis routes of the disk and network tiebreaker, but they cannot be used together. Therefore, the team decided for the network tiebreaker against the disk tie breaker because it is easy to implement, has no hardware dependencies, and evaluates the availability of communication. To overcome the downsides of the network tiebreaker, the recommendation was to add the relatively new functionality of the disk heartbeat (available since Tivoli SA MP 3.1.0.7 / 3.2.0.0). In cases where all IP network connections fail, the disk heartbeat decreases the chances of a cluster split because it is able to distinguish between a network and a node failure. With this solution, the team achieved faster failover times of approximately 30 seconds in case of a network error on the node that holds the service IP address. This is because, the network tiebreaker was not needed to resolve this error as the disk heartbeat was sufficient, and therefore there a reboot of the node was avoided.

The team followed the rules for implementing the network tiebreaker and used the gateway router in the same subnet for the network tiebreaker ensuring that there is only one hop between each node in the cluster and the tie-breaker. The node that acts as the network tie breaker also appears as the NFS server and SAP Router in the proof of concept configuration.

Network considerations

The team implemented the traditional protection of SAP systems from direct client access by shielding the SAP system, including the database resources from the client network. The backbone network covers all traffic between the database and the SAP application servers and SAP central services including intersystem communication, such as RFC and CIF calls. The SAP router node acts as the gateway between the client and the backbone network for network traffic generated by SAP GUI communication. All DB2 HADR communication routes (i.e. for log record shipping) were defined through the backbone network. On the Tivoli SA MP side, the network tiebreaker and the IP Alias were configured over the backbone network. The Tivoli SA MP IP heartbeat uses all available networks.

Global directories, such as the DB2 archive logs, **/sapmnt** and **/usr/sap/trans**, also use NFS through the backbone network. Primary and standby database have access to the same archive log directory through the backbone NFS. The administration system supporting the SAP router also acted as the NFS server in this scenario.

IP Alias

In the literature, the terms virtual IP address and IP alias are used interchangeably, therefore, in this document the team agreed to use the term, IP alias.

We have used the concept of IP alias to enable the SAP services (such as message/enqueue server, dispatcher, and so on) to automatically reconnect to the new primary database in case of a failover without being restarted. SAP fully supports IP alias takeover and also includes automatic reconnect features. The virtual host name is a reference on the DNS server or to the IP Alias in the /etc/hosts files. For more details, refer to the information on DNS in chapter 7.2 of this proof of concept.

In this case, the IP alias was defined as an additional stack on the en1 adapter (backbone network) and the virtual hostname was provided to **sapinst** as a start parameter: SAPINST_USE_HOSTNAME=<virtual hostname>. Also the cluster setup executed with the sapdb2cluster.sh script was done with the virtual hostname (DB2_HA_HOSTNAME) and IP Alias (DB2_HA_IP_ADDRESS).

Mapping DB2 HADR components to Tivoli SA MP resources

Figure 7.4.2.2 shows the mapping of the DB2 HADR components, including the IP alias, to Tivoli SA MP resources. All resource definitions have been done with the sapdb2cluster.sh script.

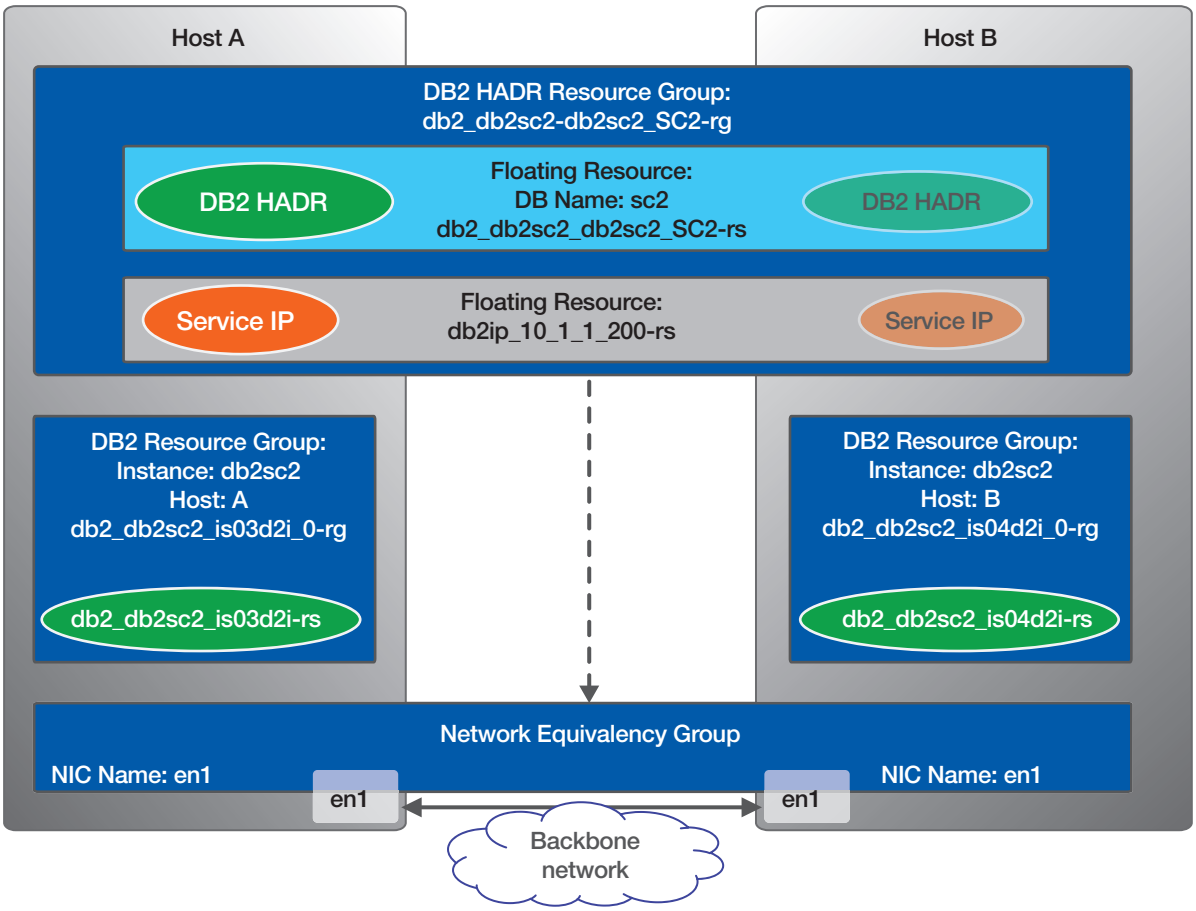


Figure 7.4.2.2: Mapping DB2 HADR components to the Tivoli SA MP resources

The DB2 HADR cluster and IP alias form one DB2 HADR resource group and are defined as floating resources. A floating resource can be moved between the nodes. Service IP represents the IP alias, the access point to the active database. In the case of a takeover, the service IP and the HADR role (primary) will be moved to the other cluster node. The DB2 instances on both the nodes are defined in the DB2 resource groups as fixed resources. Fixed resources are bound to one node.

It is important to understand the difference between the DB2 resources used for the DB2 instances and the DB2 HADR resource used for the DB2 HADR database. The DB2 resources are used to keep the instances online on their local nodes – for example, instance **db2sc2** on Host A and instance **db2sc2** on Host B. This is necessary in a DB2 HADR configuration because both sides of the HADR pair need to be online for normal operation. It is not necessary to failover the DB2 resource group containing the db2 instance (db2sc2) from Host A to Host B or vice versa. Assume that db2sc2 is the primary instance for the database SC2. If the cluster node Host A goes down, the DB2 HADR resource is used to issue a TAKEOVER command on the database SC2 on the standby instance db2inst2 on Host B.

Network equivalencies define on which network interface controller (NIC) i.e. the Service IP will be activated. Therefore, it is necessary to define a network equivalency group so that Tivoli SA MP is notified of NIC failures from the RSCT subsystem. An equivalency is similar to a resource group except that all its members will be of the same class (for example, IBM.NetworkInterface in this case).

A **DependsOn** relationship between the service IP and the network equivalency was defined to ensure that the resource (service IP) is started only when the target (NIC) is online. It includes an implicit collocation (the service IP is started at the same node as the NIC) and a force down behavior (if NIC fails, service IP is stopped). In this case, both the nodes have the NIC in the same network, and therefore, in case of NIC failure, the group is placed offline on the current node and moved to one that still has active NICs in the required equivalencies or networks.

7.5 Tivoli clustering for SAP central services for ABAP (ASCS instance)

This section describes the design around the critical components of the SAP application server architecture and how they are made highly available.

7.5.1 SAP central services – potential single points of failure

In the standard central instance implementation, there are several potential single points of failures which can stop production activities. These include the database server, the global directories (shared file systems) and the SAP central services, consisting of enqueue and message server.

The picture below shows a standard ABAP installation based on the SAP software kernel 7.0 in a non high-availability environment. The SAP central instance (SAP CI) runs the SAP primary application server including the SAP central services (SAP CS): message and enqueue services. The central services are unique system wide services which exist once per SAP system and therefore represent single points of failure together with the database and the NFS Server. The single points of failure are marked with red cycles in the graph below. With the loss of the enqueue services, online and batch transactions will be canceled and rolled back; no transactions can be started. The consequence is downtime for SAP system until the services are restored. It is, therefore, not sufficient to make the database highly available. The other sources of failures must also be addressed, as they will also negatively impact the system availability.

In regard to the global shared directories, the test team assumes that most of the customers have already a HA NFS solution in place as this is a general requirement within an SAP landscape, and therefore, exclude this from the local cluster solution. The next section concentrates on the HA solution for the SAP central services.

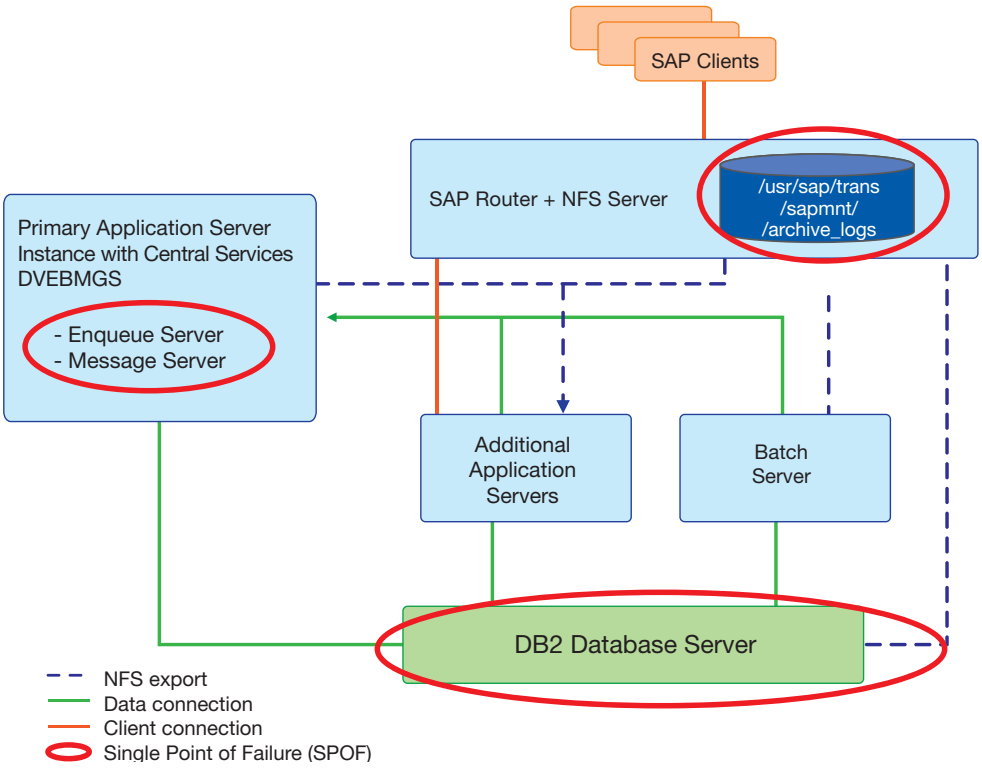


Figure 7.5.1.1: ABAP Central Instance including SAP central services, enqueue and message server (SAP software kernel 7.0)

SAP stand-alone enqueue server and SAP enqueue replication server

The SAP enqueue service maintains application logical locks in an "in-memory" buffer for speed of transaction processing. Loss of the enqueue service will result in loss of the current lock status and therefore a necessary rollback of uncommitted transactions. The system will be nonfunctional (resulting in system downtime) until the services have been restored. To improve on this, SAP has developed an enqueue solution for high-availability scenarios. The solution consists of the stand-alone enqueue server (EN) and the enqueue replication server (ERS). The enqueue replication server is used to maintain an ongoing and up-to-date copy of the state of the logical locks. Clients can connect directly to stand-alone enqueue server. When the stand-alone enqueue server fails, it will be restarted by the cluster software on standby node and the enqueue replication server will be stopped. The stand-alone enqueue server reads the replication table and builds up the enqueue table in memory exactly as it was earlier. The high-availability software routes the clients through the service IP to the new stand-alone enqueue server. In this way, the logical locks are maintained over a failure, and the application can continue.

The picture below shows the enqueue replication process. The test team followed this approach to make the enqueue services highly available.

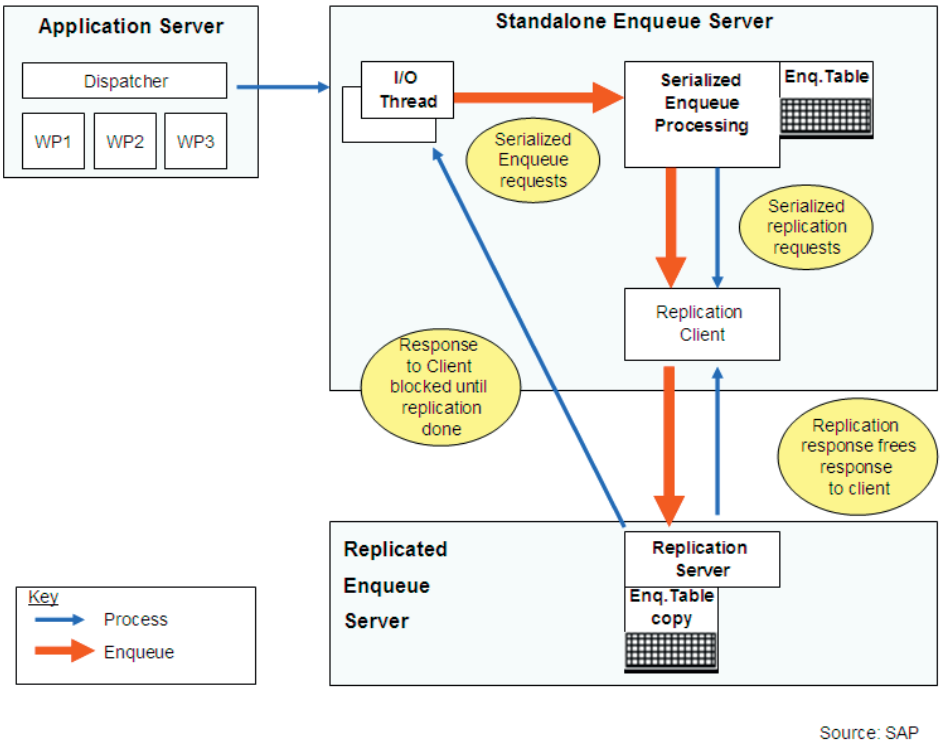


Figure 7.5.1.2: Standalone enqueue server and enqueue replication server (ERS) (Source: SAP Help Portal)

SAP central services for ABAP (ASCS instance)

The figure below shows the main SAP components and the service IP that comprises the ASCS cluster. The network and disk heartbeat components used here, have already been discussed in section: Network tiebreaker and disk heartbeat of chapter 7.4 and therefore can be omitted as the functioning is comparable.

In the figure 7.5.1.3, the components in darker blue are the active resources on this host. The areas in light blue are inactive and will be activated during failover by the cluster software.

To make the ASCS instance highly available, the team followed standard SAP installation procedure and implemented the central services instance for ABAP (ASCS instance) and the central instance with the **SAPinst** tool on Host A. As no SAP components in Java™ were used, no SCS instance was needed.

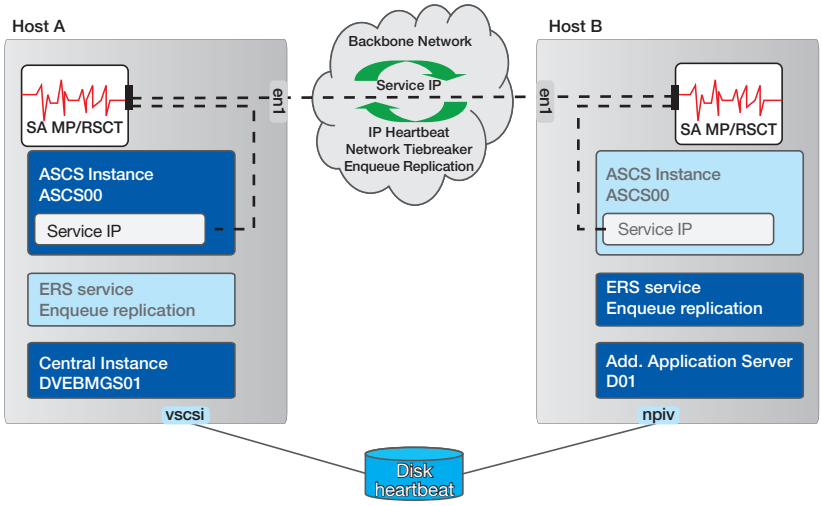


Figure 7.5.1.3: Main components of the SAP ASCS services cluster

7.5.2 SAP central services made highly available with Tivoli SA MP

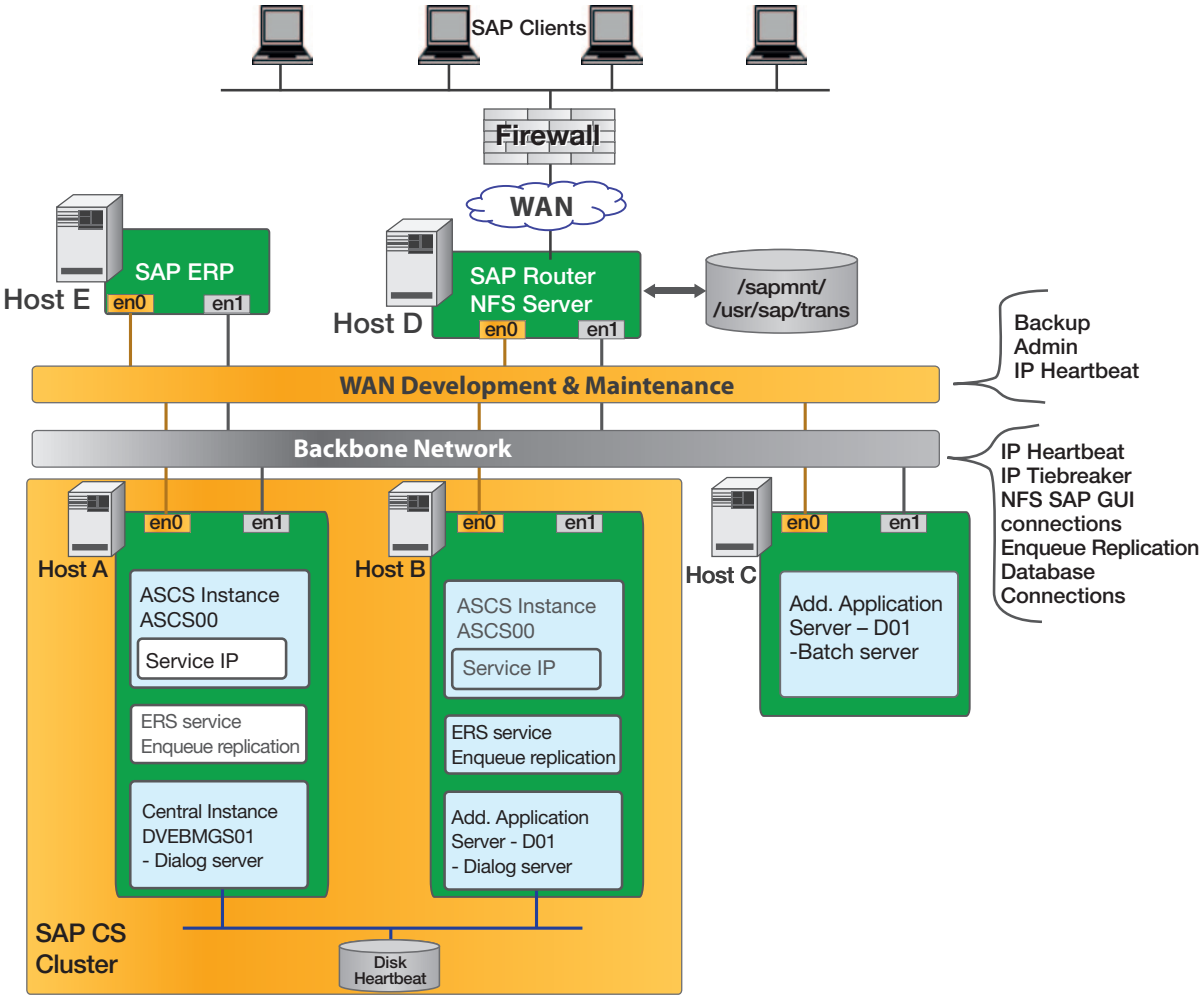


Figure 7.5.2.1: Layout – SAP central services – Tivoli SA MP cluster

Refer to the "IBM Tivoli SA MP and IBM DB2 HADR" section in chapter 7.4 for details on the following topics: Two-node scenario, network tiebreaker and disk heartbeat, network considerations, and IP alias. As the concepts and implementation were identical, the design criteria apply to both.

The ASCS instance hosting the central services on host A and the ERS on host B form the SAP central services cluster. In addition, three application servers were installed, two of them acted as dialog servers and the third one on host C covers the batch load.

Mapping SAP central services components to Tivoli SA MP resources

Figure 7.5.2.2 shows the mapping of the SAP central services components, including the IP alias, to Tivoli SA MP resources. All resource definitions were performed with Tivoli SA MP high-availability policy for SAP.

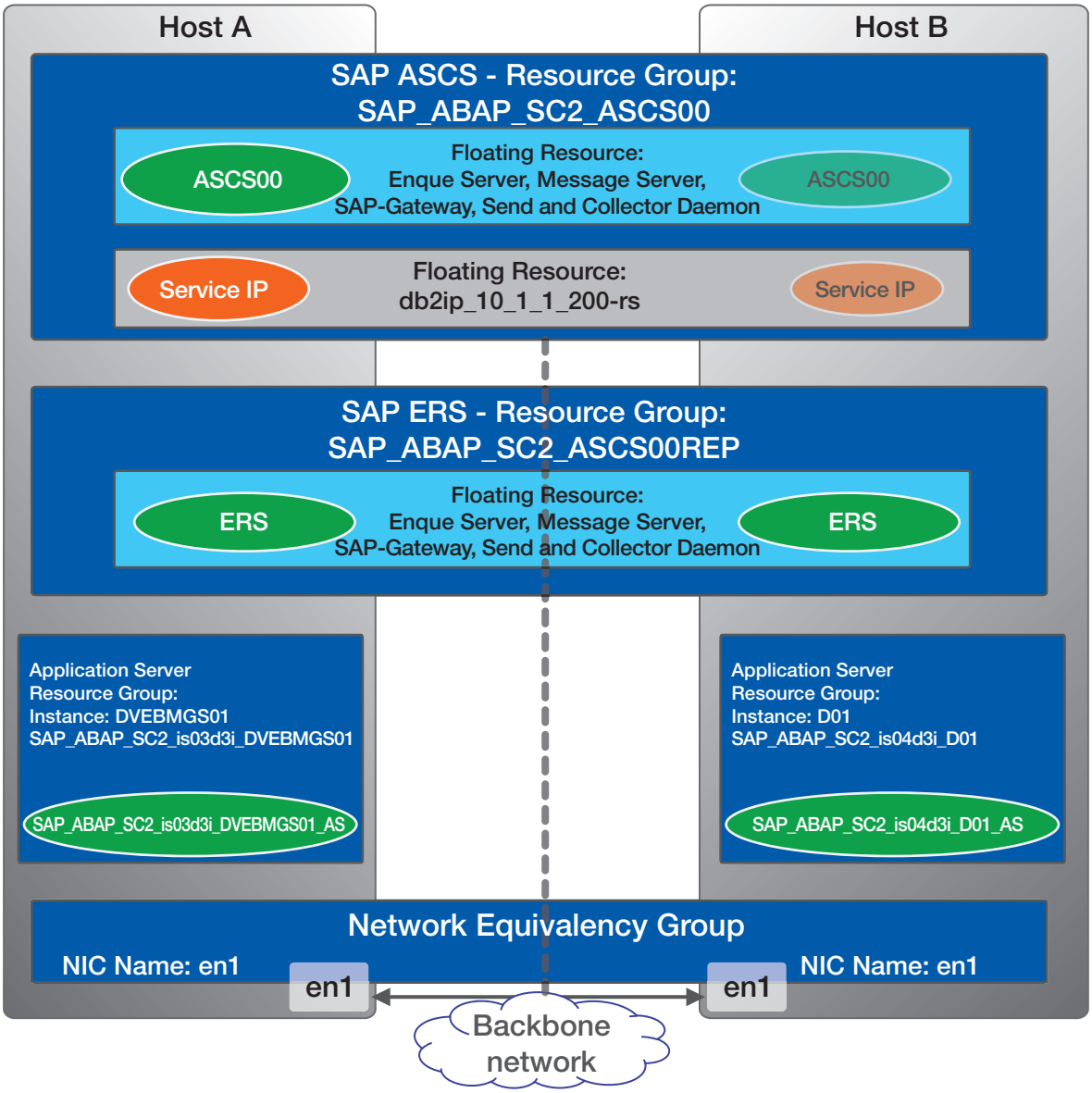


Figure 7.5.2.2: Mapping SAP central services components to Tivoli SA MP resources

The ASCS instance and IP alias form one ASCS00 resource group, and are defined as six floating resources: stand-alone enqueue server (EN), message server (MS), SAP gateway (GW), system log send (SE), system log collector (CO), and the service IP. A floating resource can be moved between the nodes. Service IP represents the IP alias. In the case of a takeover, the service IP will be moved to the other cluster node. In addition, a second resource group with floating resources for the enqueue replication server was defined. The two application servers build two resources groups with fixed resources. Fixed resources are bound to one node.

In this setup there are a number of dependencies to define the start/stop and failover behavior of the resources in a controlled manner. The SAP central services resources depend on the service IP resource and will be started only when the service IP is online. Without the service IP address, no other SAP process might be able to connect to the central services. And again as in the previous DB2 HADR scenario, there is DependsOn relationship between the service IP and the network equivalency, which ensure that the source (service IP) is started only when the target (NIC) is online

If the stand-alone enqueue server or the IP resource fails, no restart is attempted. This is because the memory buffer is already lost and the only intact copy of the enqueue table is on the replicated server, and therefore, a failover of the whole group is triggered instead. If the message server fails, first one restart is attempted. If this restart fails, the cluster manager initiates a failover of the whole group. Only the message server, the stand-alone enqueue server and the service IP resource can trigger a failover. If any of the optional three resources fails (CO, SE, or GW) and cannot be restarted, no failover is triggered.

Tivoli SA MP defines dependencies for the stand-alone enqueue and the enqueue replication server

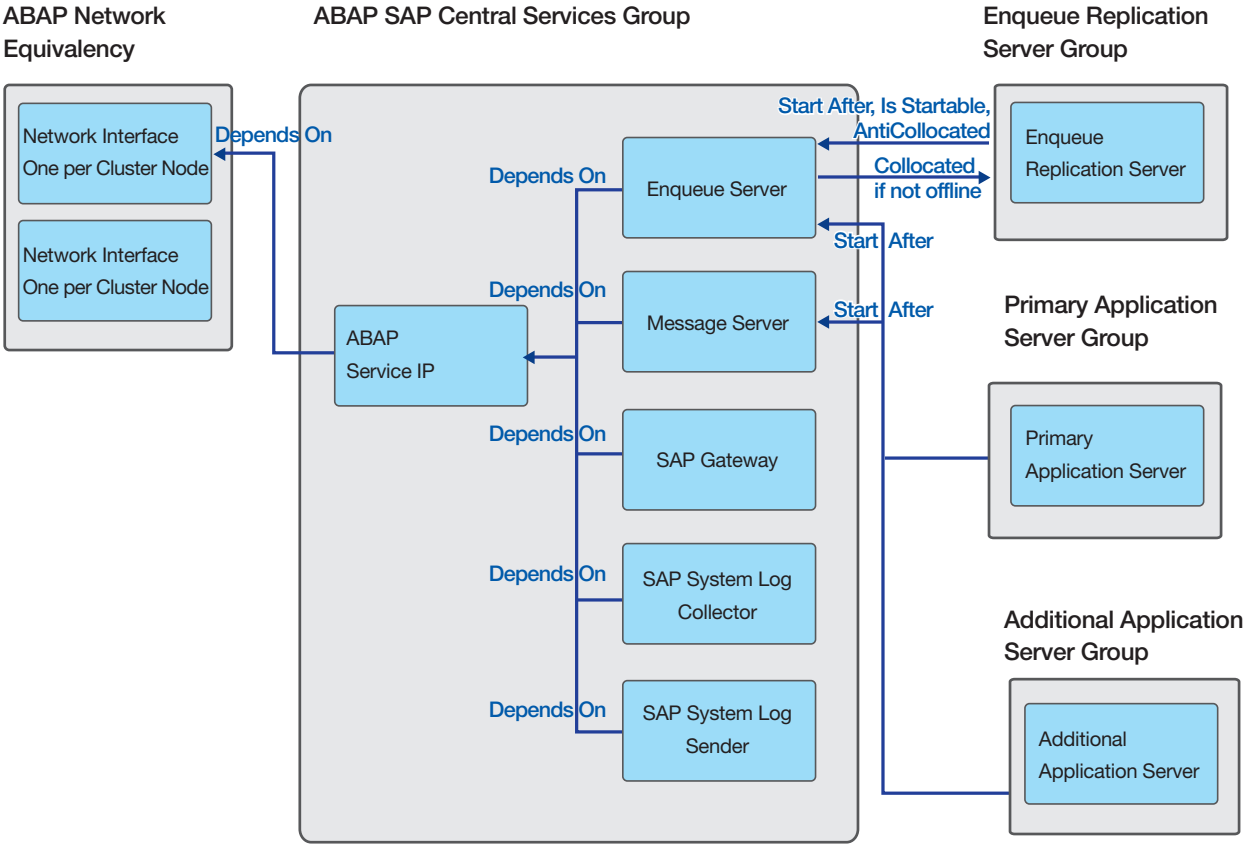


Figure 7.5.2.3: ASCS Tivoli SA MP policy

The stand-alone enqueue server and the enqueue replication server are distributed in the cluster depending on the following scenario.

Startup scenario:

- All resources are offline. The stand-alone enqueue server is started. During startup, only the Collocated/IfNotOffline relationship needs to be considered. As the enqueue replication server is currently offline, this relationship has no impact. The stand-alone enqueue server will be started in the order of the nodes listed in NodeNameList.
- Now the enqueue replication server is started. The relationships of the ERS to the EN lead to the following behavior:
  - ERS → AntiCollocated → EN: The ERS is always started on a different node than the EN.
  - ERS → StartAfter → EN: The ERS is started after the EN has become online.
  - ERS → IsStartable → EN: The ERS is only started on a node where the EN potentially can be started.

Both EN and the ERS are now online on different nodes.

Failure scenario:
In a failure scenario where the stand-alone enqueue server fails, the relationships lead to a different sequence of events, as the stand-alone enqueue server is not online as expected.

- The stand-alone enqueue server is now offline due to the failure. The stand-alone enqueue server is restarted. During startup, only the Collocated or IfNotOffline relationship needs to be considered. As the enqueue replication server is now online, this relationship starts the EN on the node where the ERS is already running. All other resources of the central services group are started on the same node where the ERS runs. After the EN has replicated the data, the ERS terminates.
- Now the enqueue replication server is restarted on a different node. The relationship of the ERS to the EN leads to the following behavior:
  - ERS → AntiCollocated → EN: The EN and the ERS are running on different nodes

7.6 SAP liveCache with HotStandby design and requirements

The following information over the implementation of SAP MaxDB with HotStandby is thanks to the SAP Labs in Berlin where SAP MaxDB and SAP liveCache are developed. A few comments are added to connect the general design for SAP MaxDB with HotStandby to implementation done for SAP liveCache on IBM System Storage. The focus in this document is on SAP liveCache with HotStandby as a component of SAP Advanced Planning & Optimization.

7.6.1 SAP MaxDB/SAP liveCache with HotStandby

A hot standby differs from a conventional failover HA solution in several ways. In a HotStandby solution, both databases are running in parallel in an active / passive partnership. The standby is maintained in a continuous restart state which allows it to maintain synchronization with the production database. This is done by reapplying log records from transactions to the standby database. The STANDBY status of the database, between ADMIN and ONLINE makes it possible for the standby to switch to active mode in a very short time, and maintain data consistency by completing all transactions.

The design of the liveCache HotStandby relies on the functionality of the storage subsystem. The requirements are the ability to generate a full stand-alone read-write split mirror of the SAP liveCache data and concurrent access by both active and standby server to the database log volumes.

The concurrent log volumes are written by the active SAP liveCache, and read by the standby. This is the mechanism for insuring that all data is synchronized in the standby. There is an ongoing communication between the active and standby SAP liveCache to keep the standby informed of the most current log record and log volume position.

In the case of a failure of the active SAP liveCache, the standby commits any outstanding transactions in the log, takes control of the log (switches to write mode) and becomes the active SAP liveCache. The SAP liveCache database instances are peers and provide rotating standby. When the failed primary server is reactivated, it will become the standby.

SAP liveCache with HotStandby relies on the operating system to provide a cluster solution to detect the failure of the active SAP liveCache, switch the standby server to active status, and initiate the transfer of the SAP liveCache IP service address from the failed server to the server assuming active status.

With HotStandby, the time needed for starting the database instance and building the memory structures is saved. This is important in the case of SAP liveCache due to the very large memory structures which must be initialized while starting. Additionally, the time needed for restoring log information is reduced to nearly nothing.

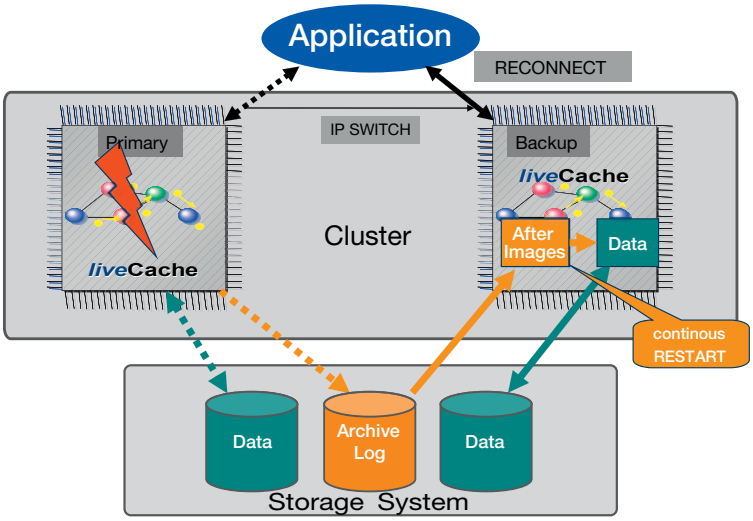


Figure 7.6.1.1: Overview of SAP liveCache HotStandby provided by SAP

Basis of SAP MaxDB with HotStandby

The solution, as supported in this HotStandby implementation, consists of two physically separate database servers with physically shared storage. The cluster instance on the failover system is used to detect a situation necessitating failover and to perform the operations needed to redirect client connections (IP Alias takeover). The SAP MaxDB with HotStandby implementation is based on two or more separate database servers (the IBM solution documented here supports a cluster pair) that access a single storage system. The data-volumes for each database are separate and the log volume is shared. Each database server must have its own unique network address.

Between master and standby instances, a synchronization channel is established which is only needed to transfer synchronization information (such as the last write position in the log volume) but not for data transfer. The bandwidth for this link can be quite small.

The LOG and DATA volumes have special requirements:

- The access type to the LOG volumes is read-write to the master, and read-only to the standby server. Access is concurrent. The IBM solution does not restrict the standby server to read-only but relies on the logic of the takeover to ensure that only one instance is actively writing to the log volume.
- Fast mirror of DATA volumes (so called split mirror or snapshot), that allows the standby server DATA volumes to be established using the current image of the master server DATA volumes and vice versa (if master/standby roles are switched). The IBM solution presented here uses the FlashCopy functionality of the IBM storage servers to generate the split mirror.
- After a mirror of the DATA volumes is established, separation must be possible which will allow both the master and the standby servers to mount their DATA volumes for read/write. The IBM FlashCopy functionality establishes a logical copy within seconds, which can then be used as a completely stand-alone and totally consistent copy. The actual physical coping of data blocks continues in background while the new copy is already read/write capable.

The SAP MaxDB runtime is extended by the API functions to allow mirror establishing, mirror separation, and read-only/read-write switching. These routines have a separate layer that abstracts the used storage system (RETHSS_API). This API is the basis of the shared library integration of SAP liveCache with IBM System Storage.

The SAP application programming interface (API) allows storage solution providers to integrate their storage functionality with the HotStandby control mechanisms of the SAP liveCache. This API exports the SAP liveCache logic that is then mapped to the functionality of the target storage server which fulfills the necessary requirements. The end result of this API integration is a shared library which is then made available to SAP liveCache and enables the HotStandby.

Configuration for HotStandby

The configuration parameters are shared between all HotStandby database servers. They consist of the normal set of parameters for SAP MaxDB databases and some extended parameters for the HotStandby solution. The parameters of SAP liveCache database instances are read only once during startup. This means that the configuration file cannot be dynamically modified. However, HotStandby nodes can be added if the master is running. The default master (by convention HS_NODE_001) and the official node name used by all clients are added.

OFFICIAL_NODE	Official node name used for client access to master node – recommended the host name used for the service IP alias.
HS_STORAGE_DLL	The name of the storage access library which implements the HSS_API.
HotStandbySyncInterval	Defines how often the master sends synchronization information to the standby, instructing the standby to continue with log recovery. The default value is 50 seconds.

All other parameters are common, especially the volume names and sizes, the logical name of the database instance, and the cache size.

Each instance will use the OFFICIAL_NODE for storing the official hostname in the SAP MaxDB system tables. The official hostname is shared over all instances. The matching local HS_NODE_NNN will be searched by using the output of **uname –n** on UNIX® systems. The SAP MaxDB runtime has an additional routine that allows the SAP MaxDB kernel to identify itself. This local node name must be a valid network name as it is used by the master instance to establish the synchronization channel to the standby instance.

The OFFICIAL_NODE must not match any of the HS_NODE_NNN entries (this is the IP Alias used for the service address and therefore cannot be bound to a node).

Each HS_NODE_NNN must be unique and assigned to a separate machine (in this solution, there is a single standby server and so there are three IP addresses: the master, the standby, and the IP Alias).

Example taken from kernel output on the HotStandby cluster:
HotStandbyNodeName001=IS03D11
HotStandbyNodeName002=IS04D11
HotStandbyStorageDLLPath=libHSSibm2145
HotStandbySyncInterval=50
HS_NODE_001=IS03D11
HS_STORAGE_DLL=libHSSibm2145
HS_SYNC_INTERVAL=50
OFFICIAL_NODE=LCHLCIP

The names with uppercase letters are old parameter names used prior version 7.7. The old names can still be used for compatibility reasons.

HS_NODE_00n = HotStandbyNodeName00n
HS_STORAGE_DLL = HotStandbyStorageDLLPath
HS_SYNC_INTERVAL = HotStandbySyncInterval

Below are some of the commands that can be used at DBMCLI level to define the HotStandby. The SAP GUI database manager tool can be used to implement the HotStandby setup directly as well.

Commands for Managing Hot Standby Systems

Commands for Experts

DBM Command	Description
hss_addstandby	Defining a standby instance
hss_copyfile	Copying database files in the cluster
hss_enable	Defining a database instance as the master instance
hss_execute	Executing DBM commands in a standby instance
hss_getnodes	Displaying the hot standby parameters
hss_removestandby	Deleting a standby instance

Figure7.6.1.2: HotStandby Commands

See chapter 9.1 „Related documents and sources of further information" for more details on Hot Standby commands.

7.6.2 Supported SAP liveCache and SAP SCM versions

HotStandby support for SAP MaxDB/SAP liveCache began with version 7.5. Information on SAP liveCache versions belonging to certain SAP SCM versions can be found at https://service.sap.com/pam.

SCM 4.0 → SAP liveCache 7.5
SCM 4.1 → SAP liveCache 7.5
SCM 5.0 → SAP liveCache 7.6
SCM 5.1 → SAP liveCache 7.7
SCM 7.0 → SAP liveCache 7.7

7.6.3 SAP liveCache and SAP APO transaction LC10

The position of SAP liveCache in an SAP SCM system, rather than as a SAP MaxDB database introduces some additional complexity for the failover cluster solution. liveCache is controlled from the SAP APO transaction LC10.

From this transaction it is started, stopped, and initialized. These activities also trigger reports in the SAP APO system which release temporary locks in the SAP liveCache and perform other cleanup/synchronization activities. It is therefore not recommended to start a SAP liveCache instance from the cluster without knowing the status it is expected to be in from the view of application. The cluster is required to maintain some knowledge of the application status – whether it is in status started or stopped.

SAP APO does provide a type of user exit or hook in the routine which starts and stops SAP liveCache. The trigger is the existence of the script, lccluster. If this script is available, information on the action being executed by SAP APO is passed to the cluster through execution of this script. If no script exists, then the cluster is not informed.

The lccluster script is expected in the dependent program path. As of SAP liveCache 7.5 and higher, this is called the installation path. So the SAP APO administration mechanism controlling SAP liveCache searches and uses the following script:

<InstallationPath>/sap/lccluster

If the script is not found at this location, it is assumed that this is not a HotStandby and no linkage is made with the cluster support.

SAP recommends using the installation path /sapdb/<SID>/db. Normally, if you install SAP MaxDB with SAP tools, this will be the default path set during the installation process.

7.6.4 Overview of the solution and support

The implementation of HotStandby supported by IBM System Storage is based on IBM flash copy functionality. Full FlashCopy with FC consistency groups is used to generate the consistent split mirror of multiple volumes. The initiation of a FlashCopy mirror creates a logical FlashCopy within seconds, which is independent and can be used in read/write mode. The actual copy of the data takes place in the background. The freshly initiated copy can be seen as something similar to a paging space – pages that are being accessed are made available immediately and updates are done directly to the new copy. As a result, the HotStandby can be activated in seconds.

The actual full data copy will complete later, and the duration for this can be in minutes or hours depending on the server type, the speed set for the copy, the amount of data, and the layout of the data on the logical disks (whether serial or parallel FlashCopy paths are used). This activity is asynchronous and transparent to the new HotStandby.

If the standby has been offline for some time and the copy of its data is no longer compatible with the current online log, the SAP liveCache will reinitiate the FlashCopy and refresh the standby servers' data.

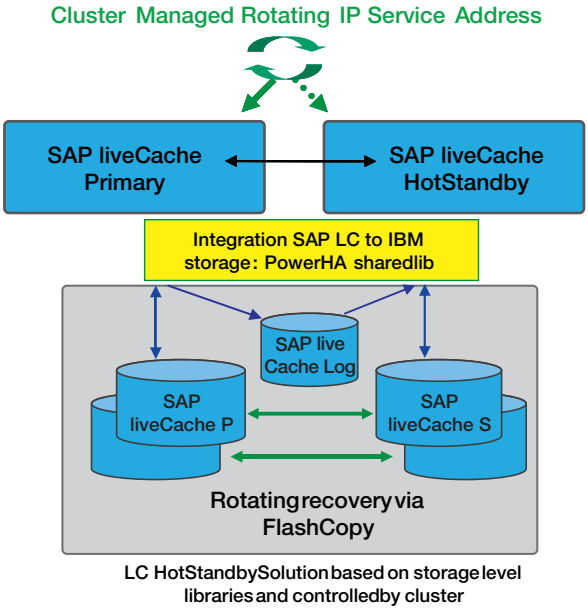


Figure 7.6.4.1: Overview of HotStandby with FlashCopy

In the case of a failure, where the standby has become primary, it will reverse the FlashCopy direction to bring the failed ex-primary back online as standby as soon as it becomes available. So the FC can flow in both the directions. A standby going online will check the state of its data and if the data is stale, will request a new FlashCopy. This FlashCopy will take place between disk pairs that are currently accessible and being accessed from both the servers. The standby will have its volumes open to check for data consistency. The log will need to access the data to check its status, and the primary will be accessing its copy of the data as it is operationally online. For this reason, the data disks are raw disks, or raw logical volumes (without file systems).
The concurrently active log is also a raw device as this is being written by one server and read by the other simultaneously.

The shared library must be installed in the search path as indicated in figure 7.6.5.1 to make it available to SAP liveCache. The other paths are default installation paths that are documented here for reference.

7.6.5 liveCache HotStandby with IBM Storage

The integration of SAP liveCache with the IBM storage servers supports the IBM System Storage DS8000, IBM Storewize V7000, and the SAN Volume Controller. The SAN Volume Controller is a storage virtualization solution, such that below the SAN Volume Controller level, any SAN storage can be used.

Figure 7.6.5.1 shows the components of the solution. The libHSS<type> is the actual shared library which IBM developed for the API provided by SAP. This library will be available as a feature of IBM AIX PowerHA, which also provides the necessary cluster management. The solution package includes storage connectors which are scripts that interface the library to the specific storage server API.

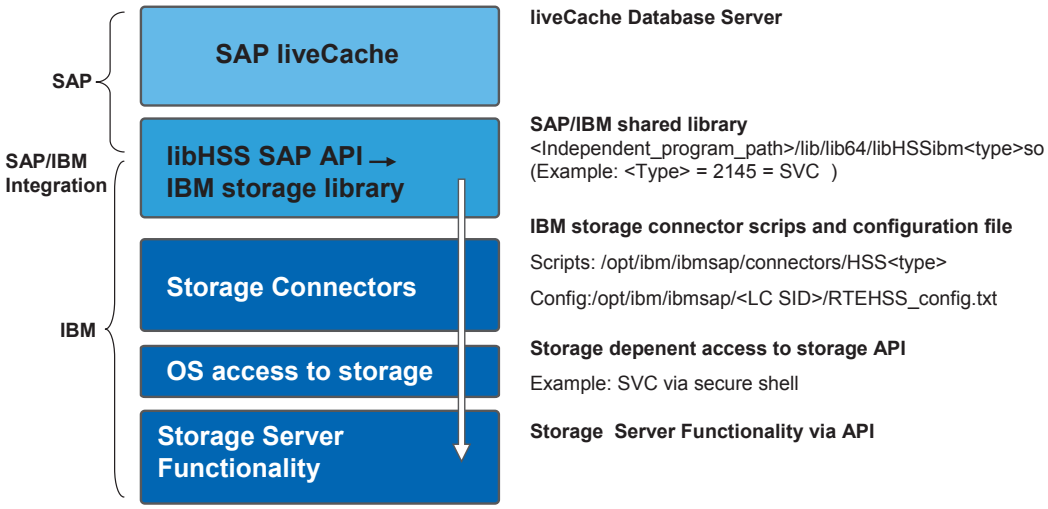


Figure 7.6.5.1: Overview of solution components

The shared library must be installed in the search path as indicated in figure 7.6.5.1 to make it available to SAP liveCache. The other paths are default installation paths that are documented here for reference.

7.6.6 The solution on virtualization

HotStandby can be implemented using directly attached storage, as well as through virtualization in PowerVM. The VIOS provides a virtualization layer for the I/O hardware, allowing the LPARs to share the adapters and I/O paths. This is described in detail in section 7.2. The major benefit for the HA implementation is that redundant I/O paths can be created inexpensively as the components are shared by all the LPARs. Virtual I/O is also a prerequisite for LPM (Live partition mobility) which extends the availability options of the solution. An LPAR can be moved from one machine to another to allow for planned maintenance without disruption to the application.

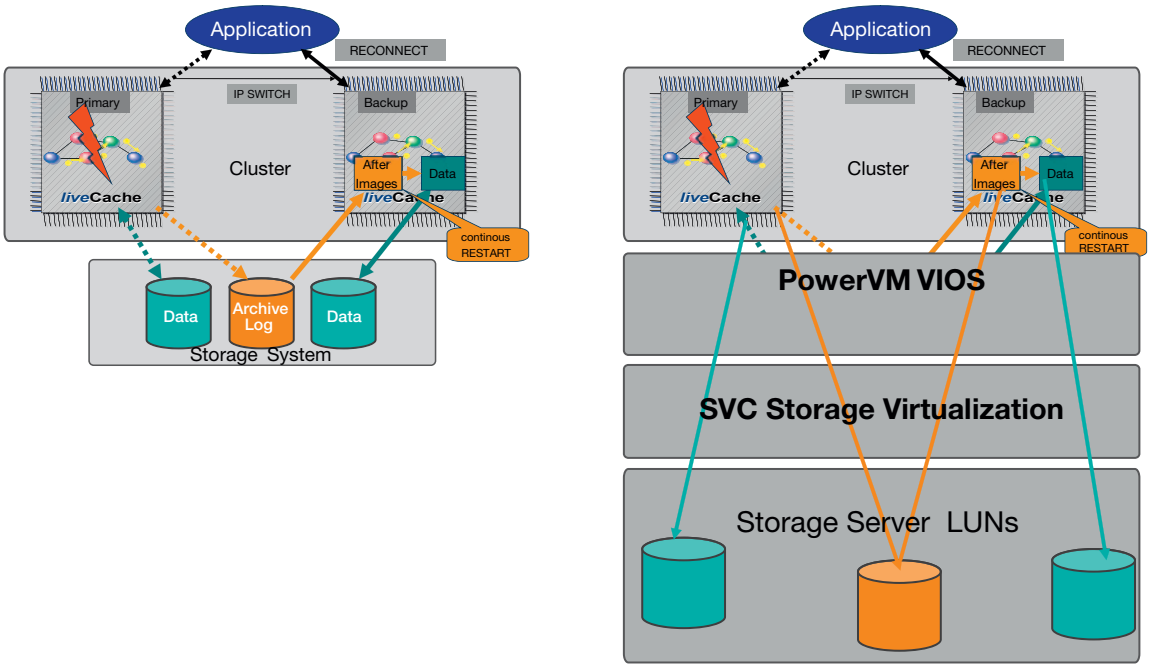


Figure 7.6.6.1: Implementation of virtualization layers

The diagrams in figure 7.6.6.1 show the original design with directly-attached storage and map this to virtualization. In the proof of concept, two levels of virtualization were used in order to ensure that the most-flexible solution was also feasible. In this case, virtual I/O functionality of the server (VIOS) provides the virtualization of the server I/O paths, and the SAN Volume Controller provides the virtualization of the storage.

The integration solution depends on the library being able to issue storage function calls to the storage server through the storage API. The connectors do this, and in order to do this, there must be a network connectivity between the LPARs on which the HotStandby solution is running, and the storage server for FlashCopy services. In the example in figure 7.6.6.1 (right), communication paths must exist between the SAN Volume Controller, which provides the FlashCopy services for downstream devices, and the SAP liveCache LPARs. The access is through secure shell communication and is described in section 7.7.9 ”SSH authorization from HotStandby to SAN Volume Controller”.

The SAN Volume Controller also provides the wherewithal to mirror the mission-critical data across storage servers. With mirrored storage, the application can survive the loss of a complete storage server without interruption.

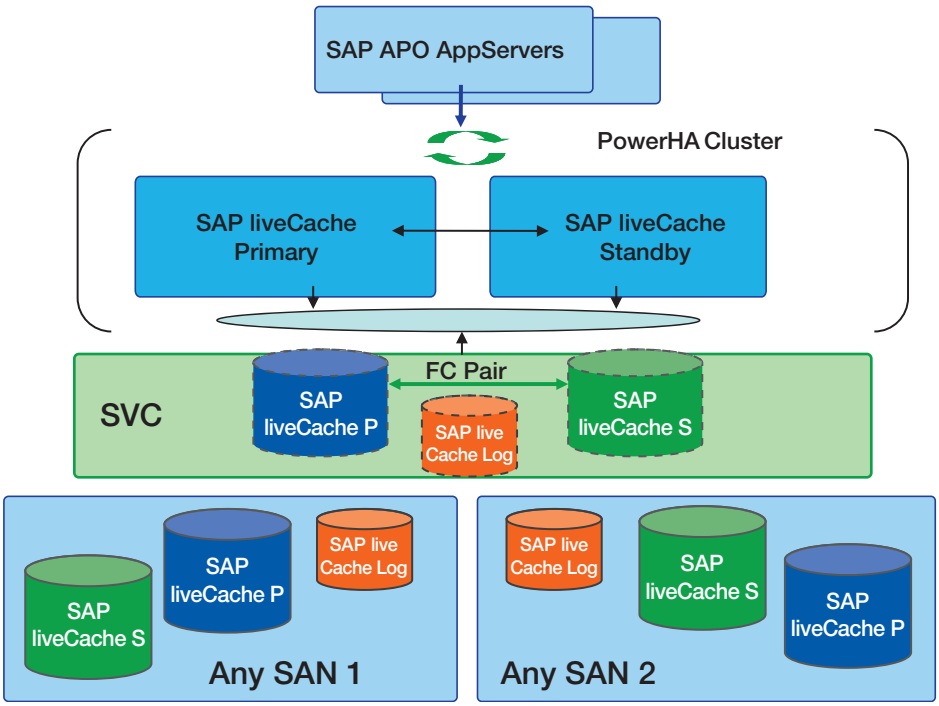


Figure 7.6.6.2: SAP liveCache with HotStandby with SVC mirrored storage

This is a significant benefit for mission critical systems. The flexibility of the virtualization layer in the SVC also makes it easy to provide a non interruptive storage system migration. The volumes can be moved from one storage pool to another (from one physical server to another) without breaking the FlashCopy relationship or disturbing the cluster solution.

7.7 Design of the PowerHA Cluster for SAP liveCache HotStandby

This example shows the design for PowerHA version 7.1. This design will be revisited for any changes coming from the PowerHA version offering SmartAssist support for the HotStandby solution.

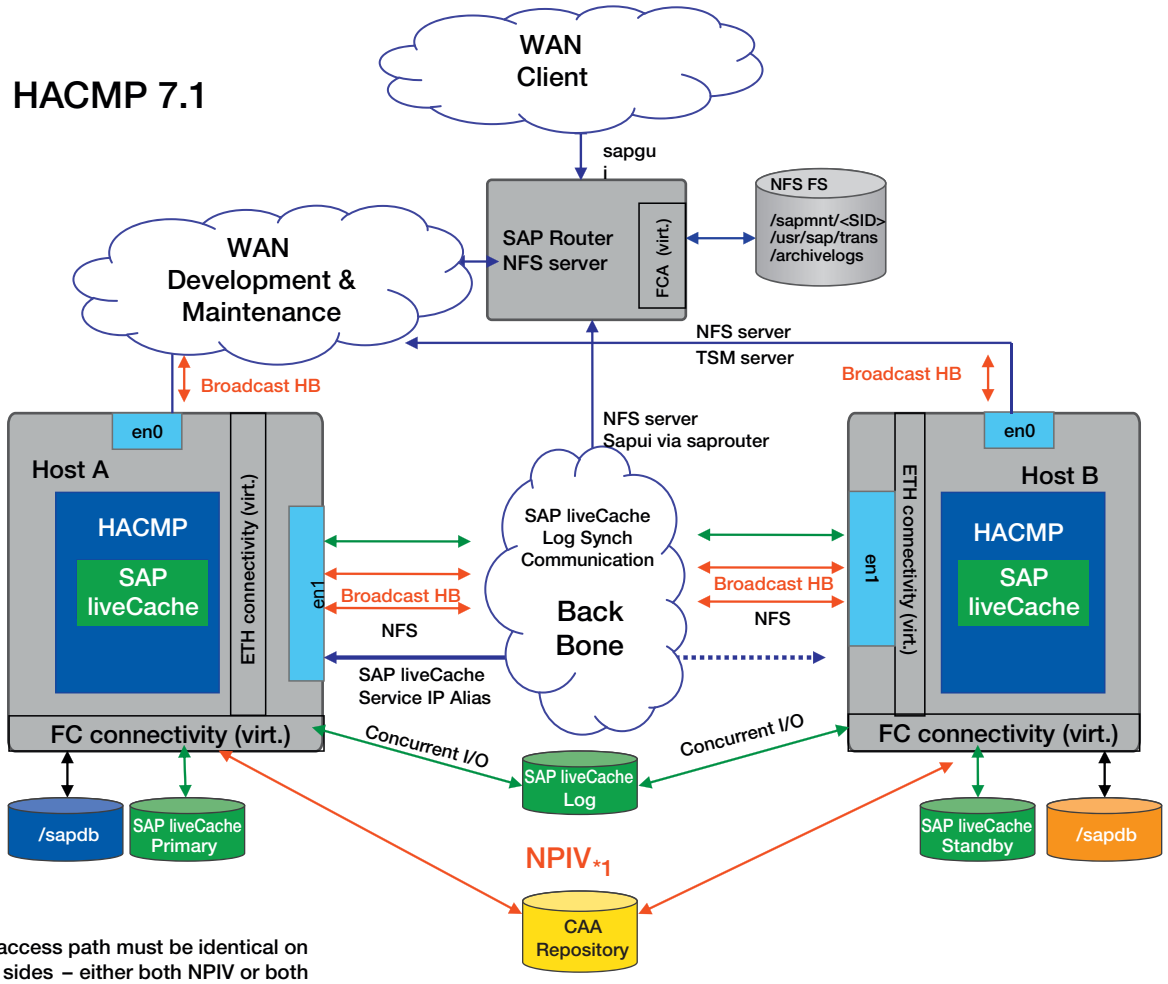


Figure 7.7.1: Overview of proof of concept cluster implementation.

PowerHA 7.1 differences:

- Move from heat-beat disk to combined repository and heart-beat functionality.
- Move from persistent boot addresses to the user of multicast I/O for heart-beat broadcast.

Definition of infrastructure and service in SAP liveCache cluster design

The PowerHA design for SAP liveCache separates the application as a service under control of the corresponding SAP APO and the required infrastructure as seen from PowerHA. The infrastructure consists of the resources combined into the resource groups which need to be made available as pre-requisites for SAP APO to be able to start the SAP liveCache as a service. The SAP liveCache as a service is started and stopped by SAP APO. The reason for this differentiation is the need for SAP APO to administer the SAP liveCache and keep synchronization between SAP APO and SAP liveCache.

- **Infrastructure:**
  Refers to the resources brought online by starting the cluster in preparation for SAP APO to start the service. It includes volume groups, service IP, application monitors and x_server.
- **Service:**
  Refers to the online SAP liveCache service as an active database, whether the SAP liveCache is online or offline according to the actions taken by the SAP APO administrator.

This separation of power results in the following two categories of control:

- Administrative cluster tasks such as starting, stopping, and moving the service provided by SAP liveCache and its required infrastructure.
- Failures of SAP liveCache as a service and/or failure of the server nodes that result in the automatic recovery provided through the implemented cluster.

High-availability considerations for network and storage related components are covered by the infrastructure design.

Cluster design – PowerHA

Figure 7.7.2 shows the PowerHA resource groups used to support the SAP liveCache cluster design. The configuration consists of three resource groups.

- RG_LC – this resource group is online on both nodes and contains the shared SAP liveCache log volume group. It must be brought online by PowerHA as it relies on the cluster functionality for concurrent access.
  Note: The data volumes are brought online on each node automatically at OS level as they require no special treatment.
- RG_LC_MASTER – this resource group consists of the IP-Alias used as the service address for SAP liveCache and the master application monitor. This resource group is only active on one of the nodes at any time.
- RG_LC_SLAVE – this resource group consists of the slave application monitor. It is only active on one node at any time different to the RG_LC_MASTER.

The primary server is represented by the resource group, RG_LC_MASTER. This resource group is active on one node or the other – never on both. It contains the IP alias used by the application to access SAP liveCache. The IP- alias and the master monitor are rotating resources.

Both the standby (slave) and the master have application monitors. The monitors have different actions depending on the role of the application that they are monitoring. The master monitor attempts to keep the SAP liveCache in the ONLINE status, whereas the slave monitor tries to maintain an instance in the standby mode if a start request from SAP APO allowed the service to be started.

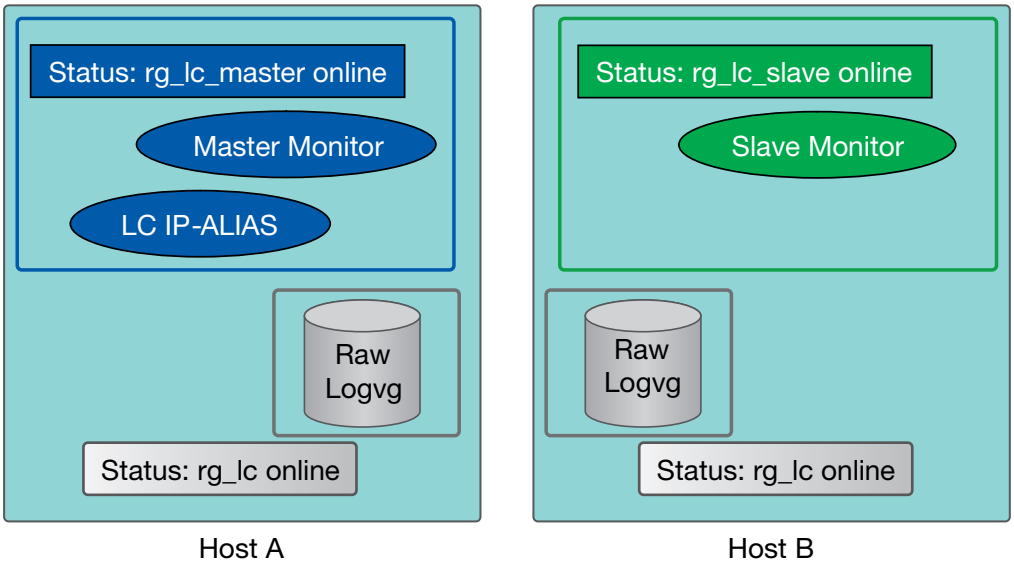


Fig 7.7.2: Overview of the PowerHA resource groups

In general, during a takeover, the slave monitor on the standby side is replaced with the master monitor and the IP alias resource is moved to the node assuming the master mode. Action is then triggered to change the SAP liveCache instance status from STANDBY to ONLINE while both the resource groups are tried to be kept in the cluster related state ONLINE.

View of resource groups in an active cluster

The clRGinfo command displays the status of the resource group infrastructure. This does not mean that the SAP liveCache service is online as this depends on the SAP APO application status. The cluster differentiates between the starting of the cluster, and the starting of the SAP liveCache service. The active cluster and the online resource groups are the prerequisite for starting the SAP liveCache service.

```
# clRGinfo
-----------------------------------------------
Group name          Group state   Node
-----------------------------------------------
rg_lc_hlc           ONLINE        is03d11
                    ONLINE        is04d11

rg_lc_hlc_master    ONLINE        is03d11
                    OFFLINE       is04d11

rg_lc_hlc_slave     ONLINE        is04d11
                    OFFLINE       is03d11
```

NFS shared repository for status information

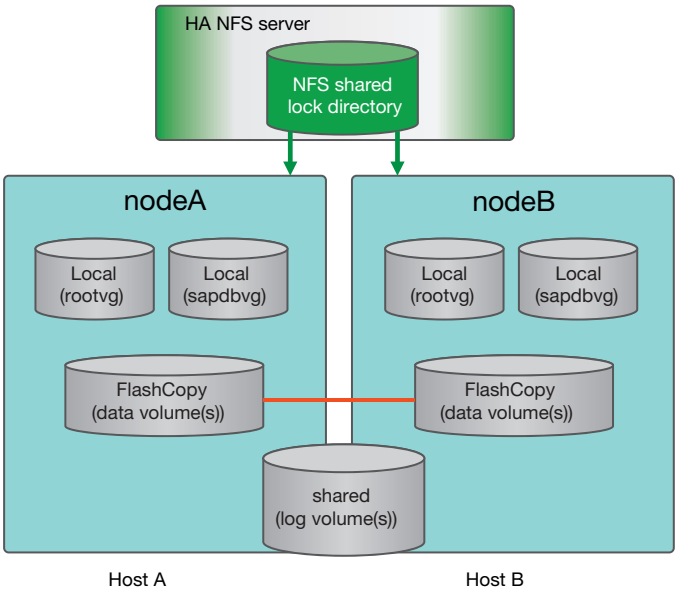


Figure 7.7.3: Overview of the storage components

The implementation design also requires a reliable NFS location for control information which influences the behavior of the cluster. The location of this lock directory is one of the configuration parameters in the cluster configuration file. The shared lock directory is used for both synchronization between the cluster and SAP APO, and the synchronization of the application monitors within the cluster.

SAP liveCache service

The responsibility for the status of the SAP liveCache database service is with the application. SAP liveCache is a component of SAP APO and integrated into the transaction LC10. From this transaction, the status of SAP liveCache is controlled. It is started, stopped, and initialized. These actions can lead to synchronization actions within the SAP APO application itself which are important for data consistency. For this reason, the cluster does not have the logical authority to start and stop SAP liveCache as a service according to the state of the cluster. The cluster must retain knowledge of the status of SAP liveCache as set by the application and manage the cluster activities accordingly.

SAP APO provides a hook in the path from LC10 to SAP liveCache that can be used to provide information on the ongoing action to the cluster. This is implemented through the presence of the script, lccluster.

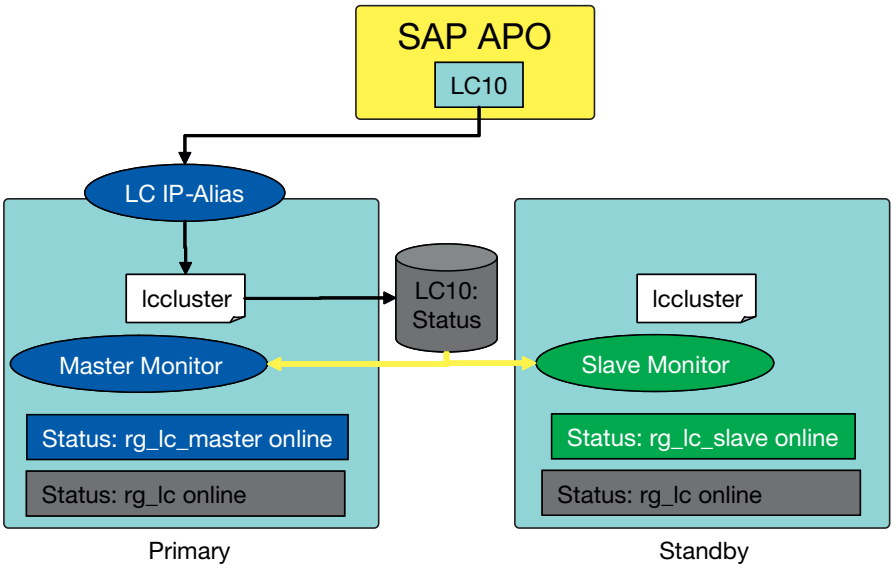


Figure 7.7.4: SAP APO control of SAP liveCache service

The PowerHA cluster uses this script hook to receive and maintain status information on the status expected by the application. When the cluster is started, this information is consulted (yellow line) and depending on the expected status, the monitors are set to active, passive, or deactivate and the SAP liveCache is brought online or not, according to the status of the last SAP APO request. The SAP APO request status is maintained in an NFS file shared by both the nodes and need to be high available. It is referred to as the lock directory later. The read/write access to this data is indicated by the yellow path.

Overview of implementation scripts

This section is provided for an understanding of the logic behind the cluster scripts. These scripts are provided by the solution and no customizing is required, beyond that done for the instance profile which is described in the implementation section.

Figure 7.7.5 shows the logic used by the cluster to ensure obedience to the applications expectation.

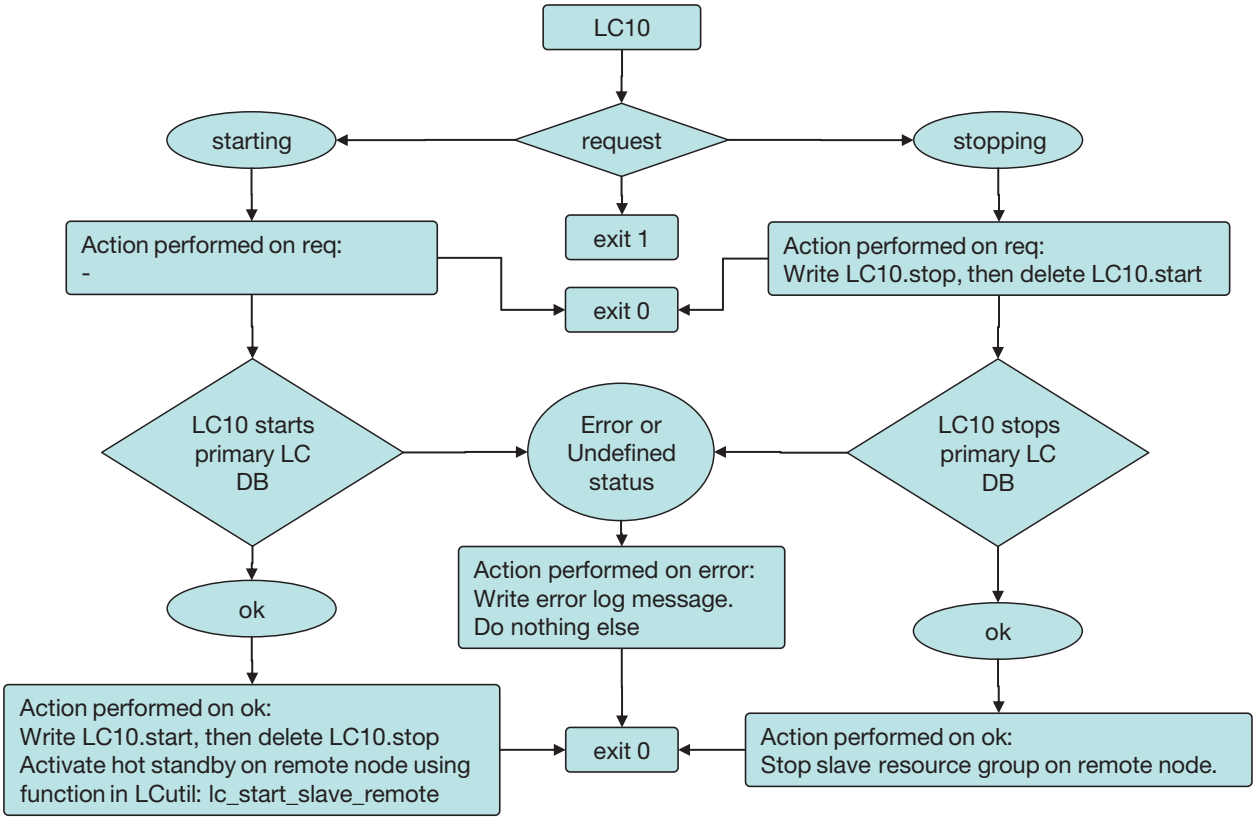


Figure 7.7.5 SAP APO to cluster control flow

The lccluster script is called by the SAP APO application one time at the beginning of an action, indicating what the intention or the request is, and again after the action, indicating the result. The cluster reacts only to the request or intent, and enters either the silent – not active – or active monitoring status. The lock directory can be empty what is treated as a stop or contains either LC10.stop or LC10.start message.

In case of a start request, SAP APO starts the master itself and the cluster arms the master runtime monitor. If the first attempt to bring SAP liveCache online fails the application runtime monitor will take over. The application monitor will drive the activities of the cluster to activate a primary SAP liveCache either on the current node or invoke a failover. Finally, the standby instance is started as well by the lccluster script. This instance will be monitored thereafter by the standby runtime monitor.

7.7.1 Starting the SAP liveCache Cluster under SAP APO control

The diagrams in this section show the logic used by the cluster to start the SAP liveCache service for the master and the standby instance. To fulfill special needs of the SAP liveCache application, the logic needs to maintain state of the SAP liveCache as set by the SAP APO application to determine whether the service should be started or not. For the master's monitors, a special logic was implemented which also maintains both application and monitor states. The monitor status and the application status are maintained in the lock directory. The location of the lock directory is configured in the central configuration file which is introduced in section 7.7.6. The monitor states are described in the next section. This section focuses on the application state.

If the cluster is being restarted, the administrator may need to reset the lock status to achieve the action that is wanted. The cluster will start the SAP liveCache service depending on the status of the LC10 information it finds in the lock directory. The LC10 status can be either LC10.start or LC10.stop. The MONITOR state is expected to be maintained entirely by the monitor logic and cleaned up during cluster start, so no monitor data will normally be present after the startup. The SAP APO LC10 status, on the other hand, is the result of the last application request. If there is no LC10 status information, the application assumes no start request from the application has been received and will not start the SAP liveCache database instance. The SAP liveCache database instance is only started if the LC10 status is LC10.start.

Cluster control hook for the APO:

The script lccluster is used to give APO full control of starting/stopping liveCache. It is activated both starting and stopping liveCache and carries out the following logical sequences according to activity.

Startup Sequence

Step 1 request:

In the picture the box for request the cluster to get prepared – marked with req) – is called, but no action is required. After the script has returned APO will start the MASTER instance. APO itself is not aware of the cluster and also not aware of an existing SLAVE. Therefore the lccluster script needs to give order to the cluster to handle the rest.

Step 2 liveCache start completed successfully:

Now the earlier mentioned lock LC10.start is set to activate the PowerHA application monitors and the standby is started.

Stopping Sequence

Step 1 request:

On request lccluster deactivates the PowerHA application monitors by changing the lock information to LC10.stop. After the script has returned APO will stop the MASTER instance.

Step 2 liveChache stop successfully:

As a cleanup task the lccluster will stop the standby instance and then exit.

In order to enable the lccluster script for a specific instance for liveCache The SID and the directory where the PowerHA start/stop scripts have been placed needs to be edited inside the lccluster script:

```
##Configure
typeset SID="HL2"                                  # edit!
typeset SAPHA_SCRIPT_DIR="/usr/es/sbin/cluster/sap"  # edit!
```

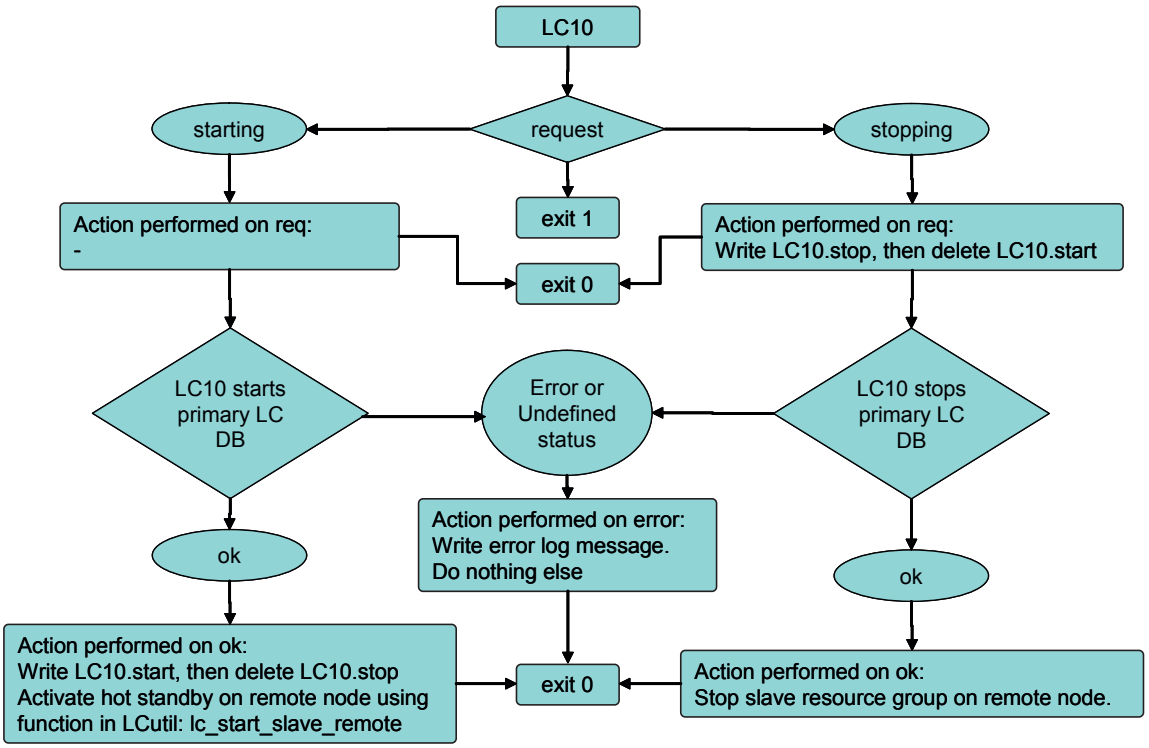


Figure 7.7.1.1 Logic of the lccluster script

To make the script cl_lc_start generic, two parameters are passed to the scripts which indicate the role and what behavior is expected: MASTER or SLAVE followed by the SID.
The two logic flows – master and slave – of the script are executed when the cluster is started as part of the resource group initialization and later they are also used during a recovery or a move.

The first action for both instances is to verify that the x_server (the SAP liveCache listener) is started. Without the x_server, it is not possible to communication with SAP liveCache.
The application status, set by SAP APO, is then consulted to determine whether the cluster has or does not have the authority to start the service. This status is maintained by the cluster and referenced to determine what action it should take as the result of a cluster restart, or a cluster failover. If the status does not exist, the start script assumes that no application direction has been given by SAP APO and the service will not be started. The lock information will be initialized to LC10.stop. The script will prepare the cluster for a SAP liveCache startup request that will be expected to come later from the SAP APO administrator, via transaction LC10.

7.7.2 Starting sequences of master

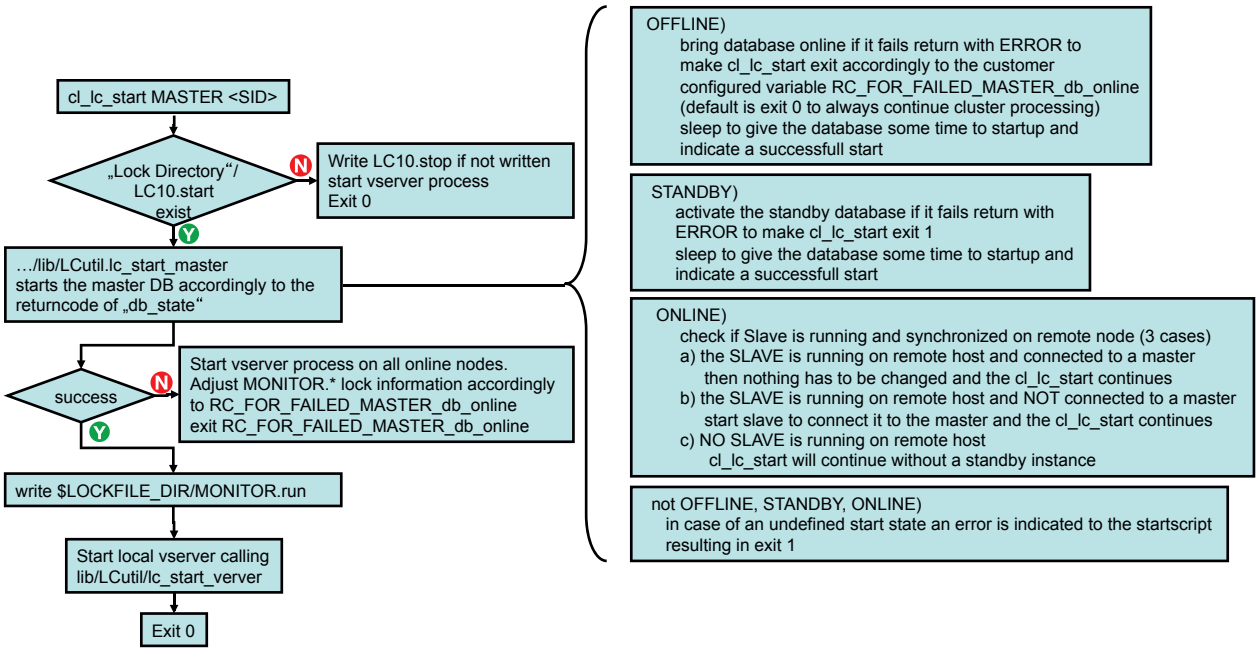


Figure 7.7.2.1 Flow logic for master instance startup

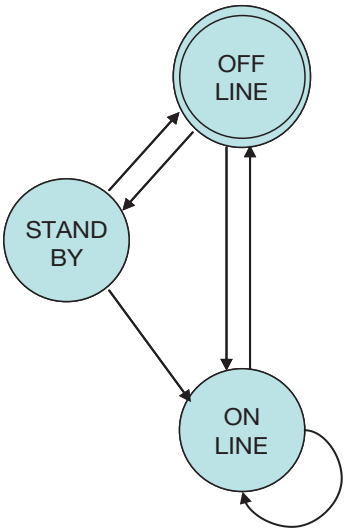
The cluster start script is executed once for either a startup or a failover to activate the SAP liveCache master. If the initial startup of the master SAP liveCache fails on the current node, there are two possible actions – attempt a failover if the second node is active or stop with error. The choice of behavior is configurable in the central configuration file (RC_FOR_FAILED_MASTER_db_online). In some cases the customer might prefer to stop the cluster initialization and investigate the problem on the master server, in other cases the customer may wish the standby to takeover so that the application can come online while the failure on the master is being investigated. The consideration is that this recovery approach can lead to a ping-pong failover if neither of the servers can successfully start the master SAP liveCache. The default is set to continue to try to bring the master online by means of a failover as long as there is a second server available.

If the master is successfully started, the cluster looks at the status of the standby node and attempts to start the standby. A standby SAP liveCache must be started via the master instance to start the ongoing synchronization communication between master and standby.
In case the master was not started successfully the default action is to update the monitor state information and return with success – this in essence completes the startup and does a handover to the runtime monitor. The application runtime monitor is activated every few seconds (configurable in PowerHA) so that after few seconds the application runtime monitor will detect the failure and initiate recovery. This default action is chosen if the configuration variable RC_FOR_FAILED_MASTER_db_online in the configuration file is set to other than 1 (setting of “1” indicates that the cluster should stop if the initial start of the master fails).

Note: The master resource group itself is configured not to attempt a restart and never to fall back. In case we have only one node available, and the db_online fails on this node, the cluster will log an error situation and stop processing. It will not continue to try to bring up the master repeatedly on the same node. For this situation a notification method should be enabled to inform the administrator instantly that the SAP liveCache database cannot be brought online.

For certain environments it could make sense to exit the start script with an error (return code =1). PowerHA reacts on this return code stopping further actions and wait for manual interaction rather than attempting further recovery. To handle this, a notification method should be enabled in PowerHA to inform the administrator instantly.

The background of $LOCKFILE_DIR/MONITOR.run is described later in this chapter. The configuration for the RC_FOR_FAILED_MASTER_db_online variable is discussed in section 7.7.2.



The state transition diagram show the state changes between master and slave database status. If the action is the result of a failover of an active database, the db_state at failover will be **standby** and the database will be brought **online** as master. If this is an initial start, the status of the master database will be **offline** and the database will be brought **online** as master and the slave as **standby**. An **online** master instance is untouched. This logic is encapsulated in the library LCutil. The function lc_start_master performs the required actions to bring the database service online accordingly to the state it detects.

Note: During a startup, these actions are only performed by the cluster when LC10.start indicator has been set by SAP APO.

Figure 7.7.2.2: Database state transition

In the case of a shutdown, if the cluster or a resource group is stopped or the application requests SAP liveCache shutdown, the database is also returned back to **offline** state. If the database application fails, SAP liveCache itself will return the instance to offline status.

In cases other than when a running Master is detected, the Slave is started by it's cluster logic. The components covering this are the application monitor and the resource group relationships defined between Master and Slave. If an ONLINE Master is detected, action is taken to ensure the Slave is correctly connected and communicating with the Master instance. In this case, the standby liveCache instance is brought online via the Master.

As the final action of a successful startup, the script starts the vserver process on the local node enabling network access to the instance.

7.7.3 Starting sequence of the slave

In case of the node being the slave, the script (figure 7.7.3.1) intentionally always returns with success in regards to cluster processing. This is necessary to continue with cluster processing otherwise the complete cluster processing will stop with error and will require manual recovery. This logic takes into consideration the fact that the slave may have failed, but the SAP liveCache can continue without a standby and therefore the cluster processing is not stopped. The slave monitor will attempt the recovery of the slave.

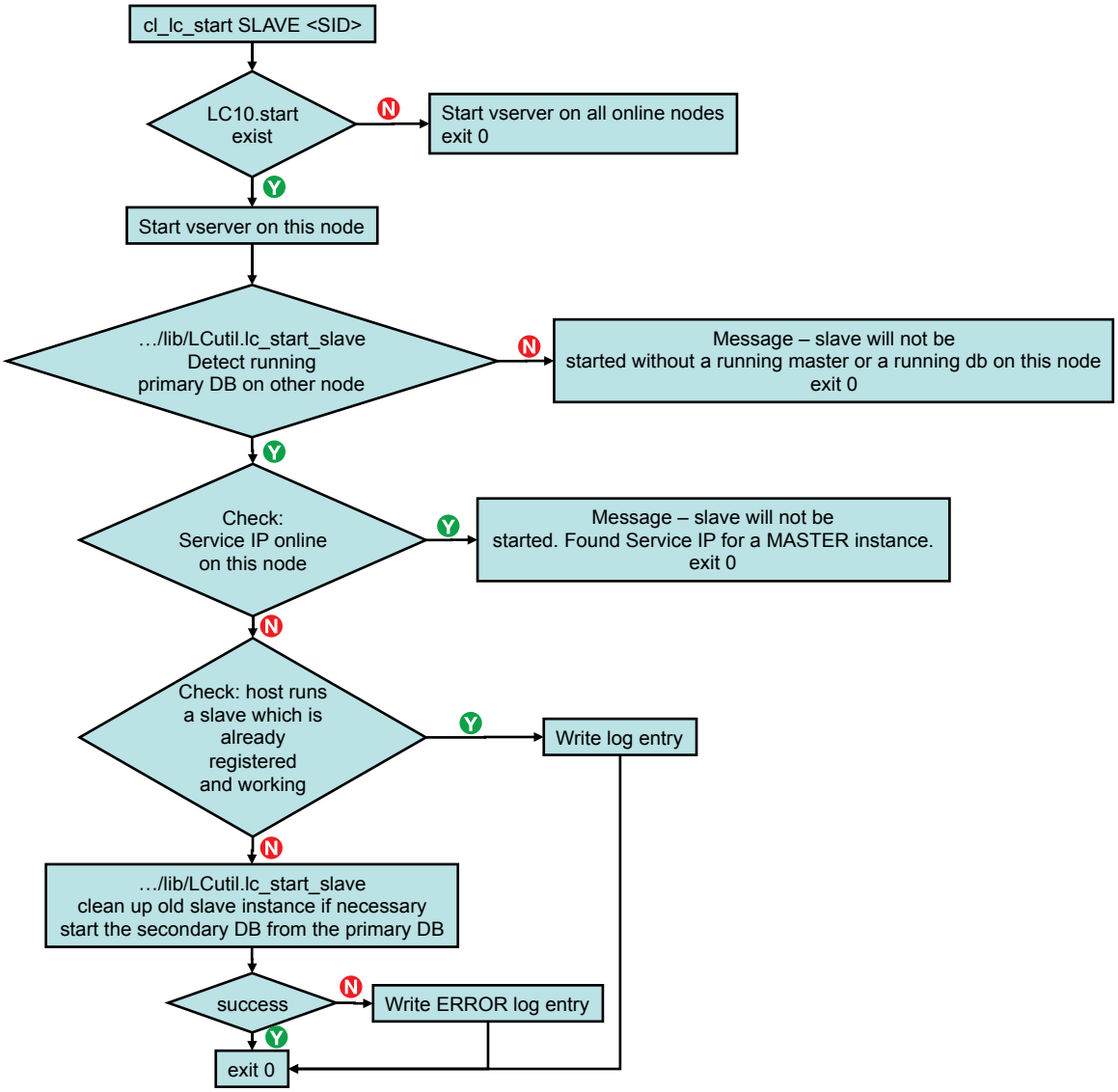


Figure 7.7.3.1: Overview of the monitor state logic

7.7.4 Stopping and cleanup of the SAP liveCache service cluster

CL_LC_STOP

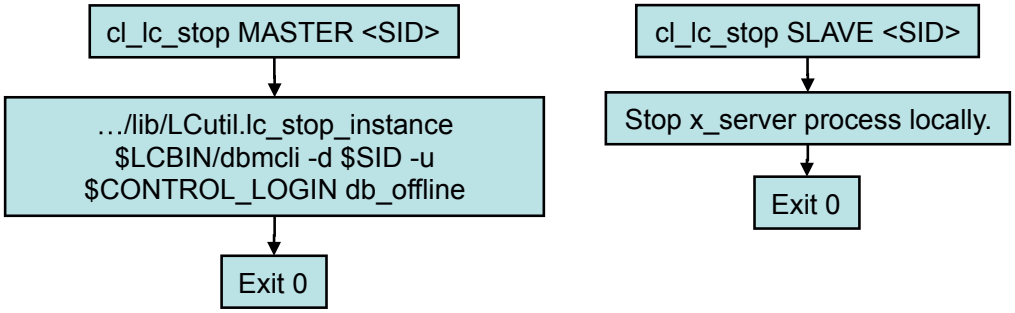


Figure 7.7.4.1: Logic of the stop script

The stop script (figure 7.7.4.1) – which is also used with parameters to handle slave and master instances – is used in failover recovery and cluster shutdown. A failing master is taken offline whereas the standby is left in standby status to complete any open failover transitions. If the standby were to be taken offline, the data cache memory structure would be deactivated and would need to be rebuilt on restart. The objective is to avoid this overhead when transitioning the standby to master status. The standby should simply change status and assume the role of the master SAP liveCache. The cleanup of slave instances is done when starting a slave and in the cl_lc_clean script.

CL_LC_CLEAN

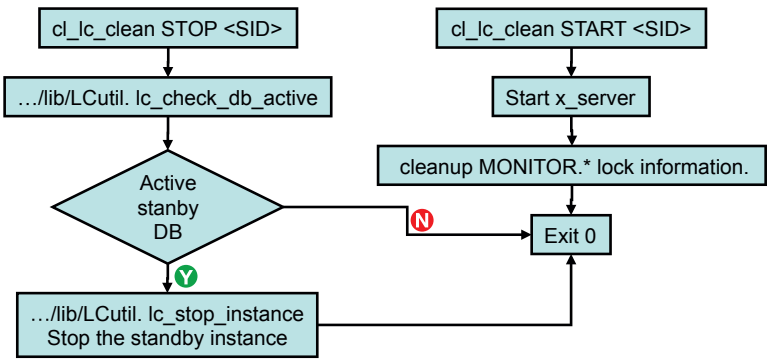


Figure 7.7.4.2: Logic of the clean up script

The clean script (figure 7.7.4.2) is used to reset the cluster status during a shutdown of the cluster. The clean up script is driven by the log resource group. If the log_volume is being taken offline, the slave will have to be stopped. The slave resource group does not take any clean up action (by design), but relies on the cleanup of the prerequisite resource group (the log) to ensure the consistent clean up of the service. This avoids any type of erroneous recovery attempt being triggered.

At cluster startup, this script ensures that x_server process is online and the lock directory is cleaned. This script is used both for a resource group shutdown and start up with different actions and therefore it is parameterized with “STOP <SID>” for shutdown, and “START <SID>” for start up.

7.7.5 Overview of script structure

Figure 7.7.5.1 shows an overview of the script structure and the configuration file used by the cluster. These are usually located in the /usr/es/sbin/cluster/sap directory. The library is a group of general routines which are made available to the individual application scripts. The application scripts are those which are configured in PowerHA as application servers and monitors of the resource groups. The sapha_env links the configuration file sapha_<SID>.cfg and the libraries into the start, stop and monitor scripts along with other environment settings which are globally referenced. The sapha_<SID>.cfg configuration file is used to adapt all the scripts accordingly to the SAP instance type, SID name, and other control setting from a single central reference.

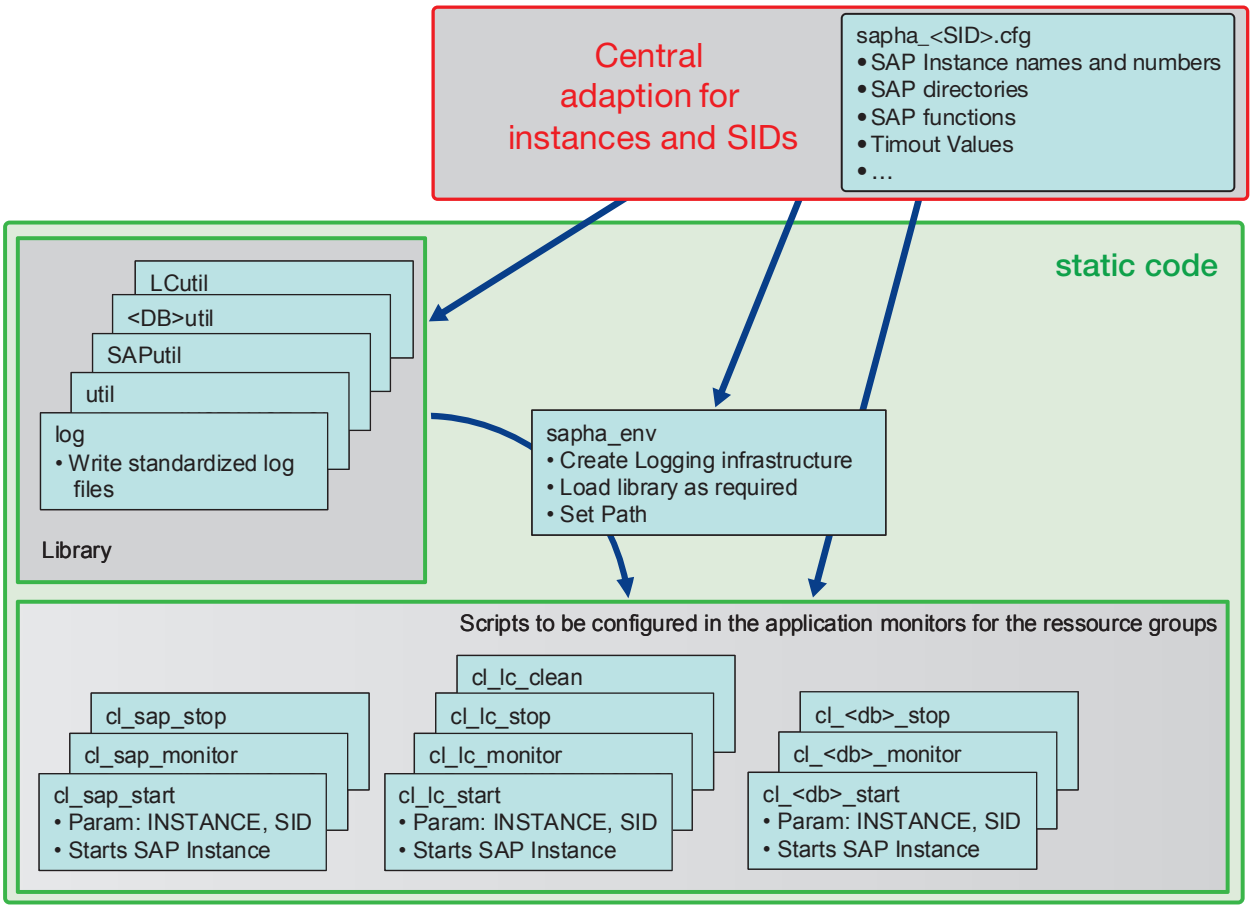


Figure 7.7.5.1: Structure of PowerHA scripts

7.7.6 Monitor logic and the monitor states

This section is provided for reference only. It describes the logic used for the master‘s application monitors. The solution provided here was done for PowerHA 7.1 and may well be implemented differently in the official PowerHA version.

PowerHA provides a structure that drives the application monitor scripts according to specific events which it monitors at low level (such as infrastructure failures) or according to configured settings. The scripts which support the application do not communicate with each other in any direct control flow and these scripts are also invoked as stateless. For a complex application

sequence such as a HotStandby, this statelessness can lead to some unpredictable behavior as result of pure circumstantial timing. To counter this, a design has been implemented which provides state information across iterations of the status monitor so that it is possible to determine the first iteration from consequent iterations. A synchronization status is also implemented between the startup script and the runtime monitor.

In the design for PowerHA 7.1 both a startup and a runtime monitor are implemented. There were three determining reasons which lead to this solution:

1) Before the start script is called, PowerHA 7.1 uses the highest priority monitor to validate that the application is not already started. If no startup monitor is available, the runtime monitor is used for this purpose as well. This would increase the complexity of the runtime monitor as it would need to cope for both roles.

2) PowerHA 7.1 starts the runtime monitor in parallel with the start script which can lead to a race condition if the runtime monitor begins looking at the database status while the start script is still bringing it online.

3) To avoid the race condition, the runtime monitor could be purposely delayed to give the start script sufficient time to complete, which would then mean its reaction time to a failure would also be delayed. As the startup time for the application can vary considerably, dependent on the necessary actions, the monitor delay would need to be generous. As the recovery time of the SAP liveCache is critical for the overall application this time must be minimized and therefore a long delay in the runtime monitor was not acceptable.

Start-up Monitor

If a single monitor is used, it will need to behave differently during the initial check, during start up, and during runtime. The first iteration determines whether the application is running and must return a negative (error RC) in order to trigger the startup. After the initial check, an error return code will trigger a failover. As the monitor is stateless, this provides a challenge.

The design implemented ensures the following:

1) The startup monitor is used for the initial check before the start script is called and always returns with RC of “1” to ensure the start script is called. Return code of 1 indicates to PowerHA that the application is not started and therefore the start script is triggered.

2) The startup monitor gives the application some time to begin the start up and to get past the state of OFFLINE into one of the transient states or even to complete the start up and achieve the ONLINE status. Then it hands over to the runtime monitor by exiting with RC=” 0”. Return code “0” tells PowerHA that the startup was successful and that the runtime monitor can now take over. The startup monitor only runs for a startup or at failover. Once the startup sequence is completed, the runtime monitor begins cyclic monitoring for a failure situation.

3) Starting the runtime monitor early minimizes the delay in starting the runtime monitor as soon as the application is ready. This improves the reaction time to any failure.

The startup monitor uses a monitor status to differentiate between the initial iteration and any subsequent iteration. In the proof of concept, only two iterations were necessary – one to trigger the startup script and one to hand over to the runtime monitor.

This design simplifies the behavior of the monitors. Not arming the runtime monitor before the start script returns avoids false failovers and this is covered by the handover approach. This approach is covered next.

7.7.7 Handover and synchronization between monitor and startup

Due to the design of PowerHA, which starts the runtime monitor in parallel with the startup script, a synchronization mechanism has been established between the startup monitor and the runtime monitor to avoid a race condition and to protect the start sequence. The lock is maintained along with the LC10 lock in the shared repository (specified in the configuration script variable $LOCKFILE_DIR). The following picture will outline the state change of the lock.

The start up script (cl_lc_start for a master instance), and both monitors for the master (cl_lc_startupmonitor, cl_lc_monitor) and the start time cleanup script (cl_lc_clean START) react to and alter the synchronization states depicted in figure 7.7.7.1.
In this figure, the final state depicted is actually a transitory situation and not an actual state. If the startup status of the application itself in successful and the database is online, then either of the states (runtime or run) can directly transition to the online status. This will only be done by the master runtime monitor. This action is described in the script logic below.

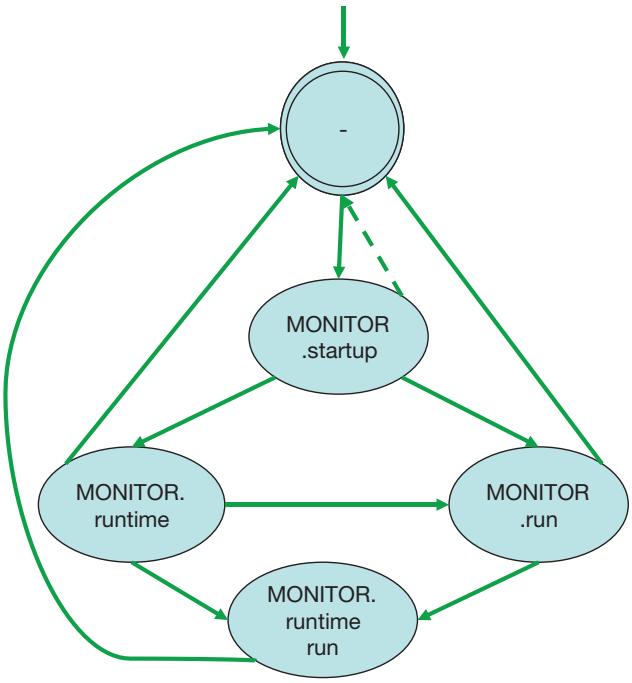


Figure 7.7.7.1: Monitor state transition

cl_lc_clean START:
when the cluster is restarted, it needs to be ensured that any previous lock information is cleaned up. The clean up script is invoked as one of the first activities in the startup sequence and it resets all monitor status. This reset is depicted by the dotted line in figure 7.7.7.1.

cl_lc_startupmonitor:
This monitor is called exactly two times by PowerHA processing. The first time it is called it sets the MONITOR.startup state in order to know it has already been called once.

The second time it changes the status from MONITOR.startup to MONITOR.runtime. Monitor runtime is information intended for the runtime monitor to inform it that the startup sequence may still be in progress. After these two iterations, the startup monitor returns "0" to PowerHA which signals PowerHA that it can start the runtime monitor. The runtime monitor (cl_lc_monitor) will run periodically from now on as long the cluster does not move the resource group. This monitor is the only consumer of the state startup, which differentiates the first from the second iteration.

cl_lc_start MASTER:
This script is triggered by PowerHA after the first iteration of the startup monitor. When the db_online command returned with success (representing an ONLINE master SAP liveCache instance) it sets MONITOR.run just before exiting back to PowerHA. If the db_online request fails, there are two options on how to proceed. The option selected is determined by the configuration of the variable RC_FOR_FAILED_MASTER_db_online. The default case for RC_FOR_FAILED_MASTER_db_online = 0. This would result in the script setting the state to MONITOR.run and exiting with RC=0. In this case it is left to the runtime monitor to discover an error status – a monitor status of RUN but no online database.

The second option, resulting from RC_FOR_FAILED_MASTER_db_online = 1, is to reset the state and remove all locks and exit with a failure. This script does not consume any lock information it only sets the state to RUN or cleans up all status as part of failure.

cl_lc_monitor MASTER:
This is the cyclic runtime monitor. As soon as the runtime monitor encounters the status of MONITOR.run, indicating that the startup sequence is completed, it removes all remaining lock status and normal monitoring is enabled. In case only MONITOR.runtime is found, indicating the start sequence may still be in progress, it checks the status of the database and if the database is online, it can proceed to clean up and enable normal monitoring. If any other DB state is found, it assumes the startup sequence is still in progress and takes no further action.
This monitor is the only consumer of the states run and runtime.

The following picture shows the call sequence of the scripts, the state relevance for the scripts, and the action taken by PowerHA as a result of the return codes in a bit more detail.

7.7.8 Logic of the monitor synchronization

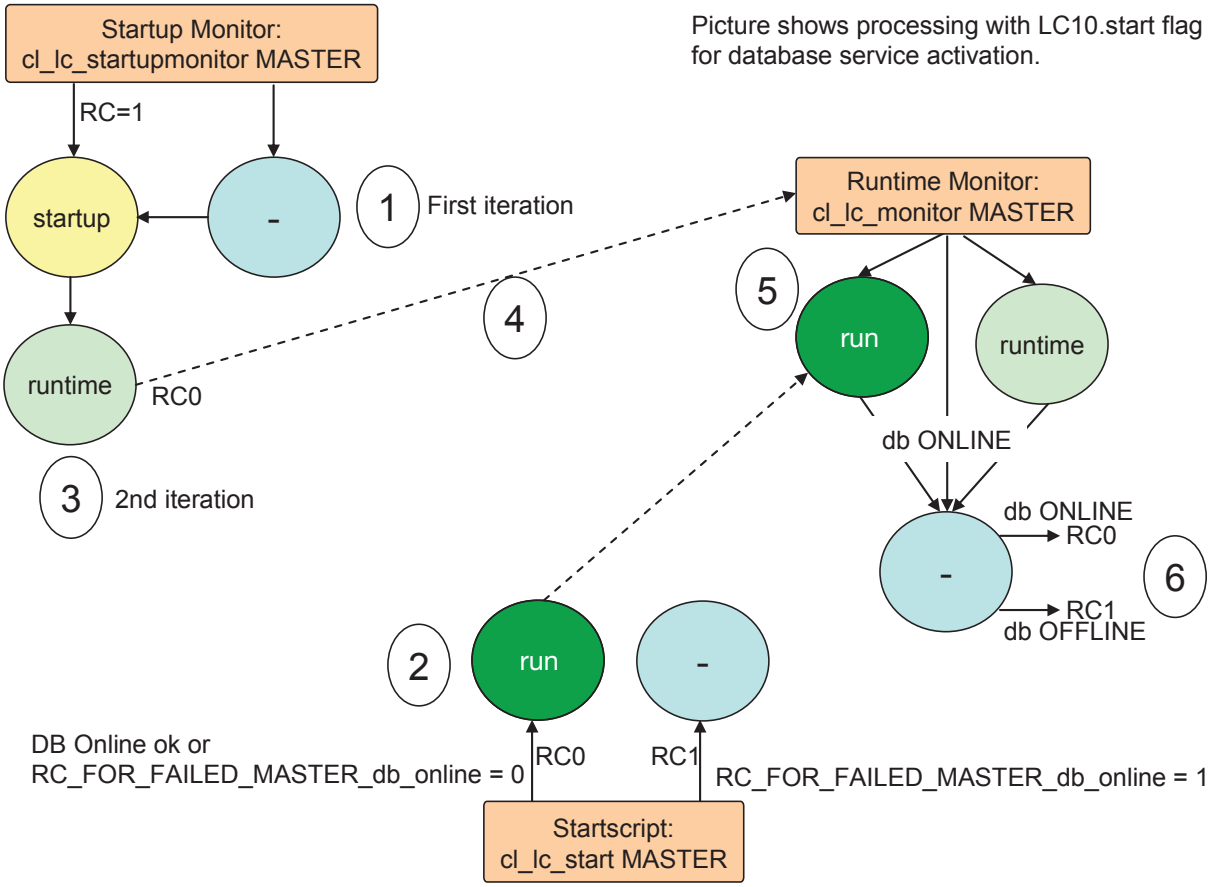


Figure 7.7.8.1: Overview of the monitor state logic

1) When the cluster starts moving the master resource group as a result of a cluster **startup** or a failover, the lock should be empty or at least is should never contain the state startup. PowerHA uses the startup monitor to check if the application is already running or if cl_lc_start MASTER SID should be called. The start script will only be executed by PowerHA if the start up monitor indicates the application is not currently active (by returning a RC=1). As we want the application start script to be triggered, we always return RC=1. To indicate the start script has been called once the lock state is set to MONITOR.startup.

2) In the case that the start script exits the dbmcli db_online command with OK, indicating the SAP liveCache instance has been set online, it sets MONITOR.run which can coexist with the MONITOR.runtime later indicated by the transitory status of MONITOR.run* in the state diagram in figure 7.7.7.1. This informs the runtime monitor later on that the startup phase is completed. The runtime monitor is now enabled to react to a database status outside the expected ONLINE status with a failover.

   This lock MONITOR.run ensures that no race condition occurs between the startup script and the application runtime monitor. In case the start script fails to bring the database online, there are two possible reaction paths. The reaction is determined by

the setting of the parameter variable RC_FOR_FAILED_MASTER_db_online in the configuration file. Either the liveCache service will be stopped with error, or (default) the initial startup will be handled by the runtime monitor.

3) PowerHA waits a configured amount of time before invoking the startup monitor in the next iteration. PowerHA will do this until the startup monitor returns RC=0 (indicating success). Since we want to call it only twice, we always return “0” on the second iteration. as MONITOR.startup status and update the lock information to MONITOR.runtime. PowerHA ensures that this step happens before the runtime monitor is called.

4) From this point on, PowerHA will use the runtime monitor and will not call the startup monitor.

5) The runtime monitor will see the lock and will give the application time to start up. This solves timing issues which could result in false failovers. Of course a failed startup must be handled. This is covered in cl_lc_start MASTER <SID> start script logic if RC_FOR_FAILED_MASTER_db_online = 1 or by the runtime monitor with some delay if RC_FOR_FAILED_MASTER_db_online = 0. The SAP command line interface used to request the database to come online (dbmcli db_online) will only return RC=0 if the database has come up with a running database in ONLINE state.

6) As soon the database is online the lock is removed and normal monitoring starts until the next monitoring event (indicated by a RC=1) triggers a failover of the resource group.

Note: The transition between the lock states run, runtime, run* before they are removed is a timing question between monitor frequency of the startup monitor and finishing the startscript. Several state combinations can lead to a successful transition to normal monitoring. Important is that the startup sequence is coordinated and results in an online database.

Note: the startup monitor must only be used for master instances.

7.7.9 SSH authorization from Hot Standby to SAN Volume Controller

The series of steps described here are not extremely intuitive so they must be followed as documented. The end result is that the AIX user, sdb will be able to access the SAN Volume Controller server with the user admin and carry out the necessary commands to drive the LC to storage integration.

Setting up the keys on AIX
The user on AIX Hot Standby side is sdb. The keys are generated from this user.

From root, change to the sdb user using the following command.
su - sdb

Now generate the SSH keys using the following command.
ssh-keygen -t rsa

There should now be a directory under /home/sdb names .ssh

```
# ls /home/sdb
.profile        .sh_history      key.bk          smit.script
.sdb            .ssh
```

Ensure that the owner is sdb.sdba for both this directory and all files in it
cd .ssh

```
# ls -l
total 24
-rw-------   1 sdb      sdba         1675 Nov  3 19:02 id_rsa
-rw-------   1 sdb      sdba          392 Nov  3 19:02 id_rsa.pub
-rw-r--r--   1 sdb      sdba          442 Nov 15 19:53 known_hosts
```

This directory will need to be copied to all the LC servers participating in SAN Volume Controller communication.
After copying, ensure that the authorizations are granted for sdb.sdba on all Hot Standby servers.

From the laptop, upload the public key in binary format through FTP or by other possible means. The key will have the following name:
is_rsa.pub

This key will now need to be loaded in the SAN Volume Controller.

Setting up the SAN Volume Controller with the public key access

Login to the SAN Volume Controller through the browser and create a user with the following characteristics: No password, local authentication, and SSH.

Attributes	Values
ID	10
Name	ha_scm
User Group ID	1
User Group Name	Administrator
Password	No
Authentication	Local
SSH Key	Yes

Figure 7.7.9.1: SVC user administration

Browse for the SSH public key file and add the public key loaded up from the AIX server.

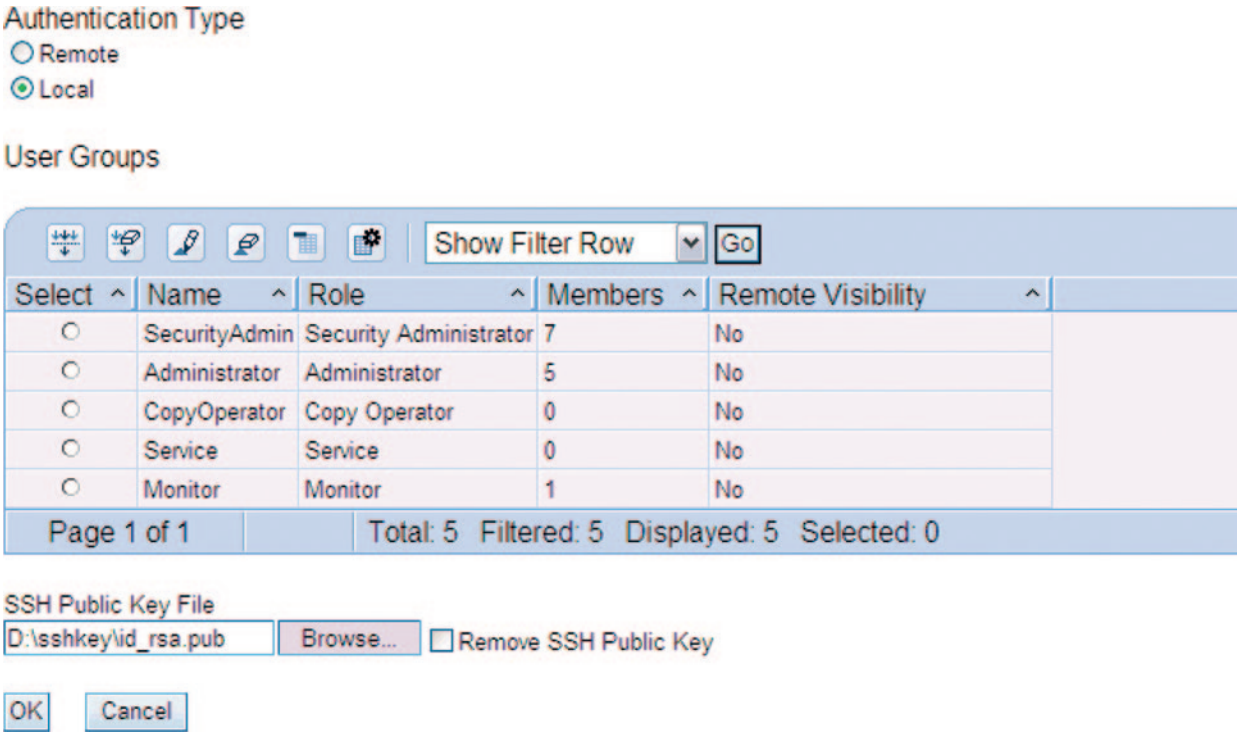


Figure 7.7.9.2: SVC user group administration

Testing the access from AIX to SAN Volume Controller
Regardless of the user you specified, you must login using the ID as admin from the liveCache server.
From user sdb, enter ssh admin@<IP address of SAN Volume Controller server>

$ ssh admin@svc_isicc
IBM_2145:svc_isicc:admin>

8 Summary and Conclusions

The results of the tests on this end-to-end high-availability infrastructure for SAP APO have demonstrated a robust, reliable, and very responsive solution infrastructure for mission critical implementations. The behavior and recoverability of the ATP check design far exceeded our hopes and expectations. That an open order using ATP checking per line item could continue as though nothing had happened even though it spanned the failure and recovery of major SAP APO components surprised and delighted almost everyone. The benefits this offers to both on-line order entry, and complex integrated solutions is nearly continuous availability, with no application level recovery requirements beyond the reactivation of asynchronous CIF queues (which are outside of the transactional critical path).

The confirmation of data consistency following the cancellation of a high volume demand planning run due to component failure proves that the system automatically recovered to such a state that the batch planning jobs and simply be restarted. The batch restart can be initiated within minutes of the failure, which helps to ensure the overnight production planning runs within a tight window of time. The jobs will restart from the beginning, but operations can bypass much of the prerequisite recovery overhead which can represent a major delay.

The selection of components to support the high-availability and cluster solutions have proven a very viable reference architecture stack, which can take full advantage as well of all the strengths of the Power systems virtualization and flexibility. The PowerVM functionality provides a very strong high-availability infrastructure with an eye on TCO.

Based on new functionalities in Power7, additional tests were performed in regards to large SCM systems which are published in a related document. These include the benefits of Active Memory Expansion for large application memory footprints such as the database and liveCache. These tests also demonstrate the option of using Live Partition Mobility to extend the continuous availability of the SCM system. LPM allows a running LPAR to be relocated across machine boundaries and thereby enabling physical server maintenance without application down time. This is beneficial for SCM even if there is a sufficient downtime window, as this method would not require the (sometimes very large) memory resident data cache to be rebuilt and refilled as would be the case with an actual stop and restart of the APO application.

The SAP Supply Chain Management Advance Planning and Optimization application has an optimal infrastructure on Power. PowerVM provides resource sharing between the components (database, liveCache, application servers) according to the load requirements and is able to realign with a change in load distribution. Considering the various scenarios running in the planning cycle of APO, including demand planning, supply and network planning, production planning and detailed scheduling, and service parts planning (Spare Parts Management), there are a number of load distribution profiles which can run in a single system and in a single planning cycle. PowerVM provides excellent load consolidation and automatic (nearly instantaneous) resource realignment according to load requirements.

With this reference architecture, the SAP SCM APO achieves “end to end” high availability on top of all the virtualization benefits provided by PowerVM. It takes advantage of the clustering

benefits of Tivoli SA MP for both SAP central services recovery, and extremely fast database recovery using the DB2 HADR.
And finally, SAP SCM APO is given a boost of functionality with the extended high availability option of PowerHA SAP liveCache Hot Standby, completing the HA coverage for all components. This solution provides the integration of SAP liveCache with IBM storage functionality in support of a Hot Standby liveCache in synchronization with the primary server both on disk and in memory. The solution also takes advantage of state-of-the-art storage virtualization. The flexibility of the SAN Volume Controller for storage virtualization allows running applications to be moved between storage servers “on the fly”. This allows storage systems to be extended or replaced without application down-time. In addition, the mirroring functionality of the SVC can be used to ensure uninterrupted production in SCM even if a complete storage server should fail.

This proof of concept presents a fully tested end-to-end infrastructure which is proven to be an extremely solid basis for mission critical supply chain management.

9 Appendix

9.1 Related documents and sources of further information

An IBM RedBook is available online with details on the functionality of PowerHA 7.1.
http://www.redbooks.ibm.com/abstracts/sg247845.html?Open

SAP liveCache Hot Standby command set can be found at the website below:
http://maxdb.sap.com/doc/7_7/45/0f77bbe82f29efe10000000a114a6b/frameset.htm

Further information on the SAP liveCache command line and DBMGUI tool.
Command Line (dbmcli):
http://help.sap.com/saphelp_nwpi711/helpdata/en/a3/b2462a9ef05c41922b8092257a2e2c/frameset.htm

DBMGUI:
http://help.sap.com/saphelp_nwpi711/helpdata/en/a3/b2462a9ef05c41922b8092257a2e2c/frameset.htm
(use version 7.6 and higher)

9.2 SAP liveCache versions

The proof of concept was executed using two different version of SAP liveCache:

- 7.7.07.17
- 7.7.07.26

Upgrade to SP16 (7.7.07.26) was done due to a known problem in failover recovery in versions prior to support pack 16 that which did occur in the testing.

9.3 AIX, PowerHA, and Java versions

The proof of concept started with PowerHA version 6.1 and then migrated to version 7.1. The version that will provide the official support for SAP liveCache with HotStandby will be closer to 7.1 and the logic implemented for 7.1 will be carried forward into the product.

1) bos packets *(6100-04-03-cluster)*

Smitty nim → install software lppsource_61TL04SP02

bos.adt.syscalls	5.3.7.0	# Base Level Fileset
bos.adt.libm	6.1.2.0	# Base Level Fileset
bos.adt.syscalls	6.1.2.0	# Base Level Fileset
bos.data	5.3.0.0	# Base Level Fileset
bos.net.nfs.server	5.3.7.0	# Base Level Fileset
bos.clvm	→ for concurrent access to logical volumes	
bos.rte.lvm	→ for concurrent access to logical volumes	

2) PowerHA 6.1 + all available APARs for high availability

AIX 6.1 TL05 SP03

cluster.doc.en_US.es.html	6.1.0.0	HAES Web-based HTML
cluster.doc.en_US.es.pdf	6.1.0.0	HAES PDF Documentation – United States
cluster.es.cfs.rte	6.1.0.0	ES Cluster File System Support
cluster.es.client.clcomd	6.1.0.2	ES Cluster Communication
cluster.es.client.lib	6.1.0.1	ES Client Libraries
cluster.es.client.rte	6.1.0.2	ES Client Runtime
cluster.es.client.utils	6.1.0.1	ES Client Utilities
cluster.es.csPoC.cmds	6.1.0.2	ES CSPOC Commands
cluster.es.csPoC.dsh	6.1.0.0	ES CSPOC dsh
cluster.es.csPoC.rte	6.1.0.2	ES CSPOC Runtime Commands
cluster.es.nfs.rte	6.1.0.1	ES NFS Support
cluster.es.plugins.dhcp	6.1.0.0	ES Plugins – dhcp
cluster.es.plugins.dns	6.1.0.0	ES Plugins – Name Server
cluster.es.server.cfgast	6.1.0.0	ES Two-Node Configuration
cluster.es.server.diag	6.1.0.2	ES Server Diags
cluster.es.server.events	6.1.0.2	ES Server Events
cluster.es.server.rte	6.1.0.2	ES Base Server Runtime
cluster.es.server.utils	6.1.0.2	ES Server Utilities
cluster.es.worksheets	6.1.0.0	Online Planning Worksheets
cluster.license	6.1.0.0	HACMP Electronic License
cluster.es.client.clcomd	6.1.0.2	ES Cluster Communication
cluster.es.client.lib	6.1.0.1	ES Client Libraries
cluster.es.client.rte	6.1.0.2	ES Client Runtime

cluster.es.PoC.rte	6.1.0.0	ES CSPOC Runtime Commands
cluster.es.nfs.rte	6.1.0.1	ES NFS Support
cluster.es.server.diag	6.1.0.0	ES Server Diags
cluster.es.server.events	6.1.0.0	ES Server Events
cluster.es.server.rte	6.1.0.2	ES Base Server Runtime
cluster.es.server.utils	6.1.0.2	ES Server Utilities
cluster.man.en_US.es.data	6.1.0.0	ES Man Pages – U.S. English

3) PowerHA 7.1 – Used for SAP liveCache HotStandby

AIX Level Used: 6100-06-03-1048

cluster.doc.en_US.es.html	7.1.0.0	PowerHA SystemMirror Web-based
cluster.doc.en_US.es.pdf	7.1.0.0	PowerHA SystemMirror PDF
cluster.es.assist.common	7.1.0.0	PowerHA SystemMirror Smart
cluster.es.assist.db2	7.1.0.0	PowerHA SystemMirror Smart
cluster.es.assist.dhcp	7.1.0.0	PowerHA SystemMirror Smart
cluster.es.assist.dns	7.1.0.0	PowerHA SystemMirror Smart
cluster.es.assist.domino	7.1.0.0	PowerHA SystemMirror
cluster.es.assist.filenet	7.1.0.0	PowerHA SystemMirror Smart
cluster.es.assist.ihs	7.1.0.0	PowerHA SystemMirror Smart
cluster.es.assist.MaxDB	7.1.0.0	PowerHA SystemMirror Smart
cluster.es.assist.oracle	7.1.0.0	PowerHA SystemMirror Smart
cluster.es.assist.sap	7.1.0.0	PowerHA SystemMirror Smart
cluster.es.assist.tds	7.1.0.0	PowerHA SystemMirror Smart
cluster.es.cfs.rte	7.1.0.0	Cluster File System Support
cluster.es.client.clcomd	7.1.0.0	Cluster Communication
cluster.es.client.lib	7.1.0.0	PowerHA SystemMirror Client
cluster.es.client.rte	7.1.0.0	PowerHA SystemMirror Client
bos.cluster.solid	6.1.6.3	POWER HA Business Resiliency
cluster.es.server.utils	7.1.0.0	Server Utilities
cluster.es.worksheets	7.1.0.0	Online Planning Worksheets
cluster.license	7.1.0.0	PowerHA SystemMirror
bos.cluster.rte	6.1.6.3	Cluster Aware AIX
bos.cluster.solid	6.1.6.0	POWER HA Business Resiliency
cluster.es.client.wsm	7.1.0.0	Web-based Smit
cluster.es.cspoc.rte	7.1.0.0	CSPOC Runtime Commands
cluster.es.migcheck	7.1.0.0	PowerHA SystemMirror Migration
cluster.es.nfs.rte	7.1.0.0	NFS Support
cluster.es.server.diag	7.1.0.0	Server Diags
cluster.es.server.events	7.1.0.0	Server Events
cluster.es.server.rte	7.1.0.0	Base Server Runtime
cluster.es.server.utils	7.1.0.0	Server Utilities
cluster.man.en_US.es.data	7.1.0.0	Man Pages – U.S. English

5) Java

Install the required Java version. We used Java 1.4.2 SR13 for this scenario

9.4 Tivoli SA MP and DB2 HADR versions

IBM DB2:	V9.7 FP 2
SAP:	
SAP patch level for SAP (disp+work/and DBSL):	94
kernel release	701
compiled for	64 BIT
compilation mode	UNICODE

Tivoli System Automation for Multiplatforms: 3.2.1

9.5 Power Systems, AIX, and storage versions

Firmware version: IBM,AL720_066

AIX: Both, 6100-05-03-1036 and AIX 6.1 TL05 SP03 tested

SAN Volume Controller Level 6.1.0.5

10 Copyrights and Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States and/or other countries: Advanced Micro-Partitioning, AIX/L(logo), AIX 5L, DB2 Universal Database, eServer, i5/OS, IBM Virtualization Engine, Micro-Partitioning, iSeries, POWER, POWER4, POWER4+, POWER5, POWER5+, POWER6. POWER7

A full list of U.S. trademarks owned by IBM may be found at:
http://www.ibm.com/legal/copytrade.shtml

UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

SAP, the SAP logo, SAP NetWeaver, and ABAP are trademark(s) or registered trademark(s) of SAP AG in Germany and in several other countries.

More information about SAP trademarks can be found at:
http://www.sap.com/company/legal/copyright/trademark.asp

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

Other company, product or service names may be trademarks or service marks of others.

Information is provided “AS IS” without warranty of any kind.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products.
Questions on the capability of non-IBM products should be addressed to the supplier of those products.

11 Disclaimer and Special Notices

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products.

All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

ISICC PRESS CTB-2011-1.2

IBM SAP International Competence Center, Walldorf