connect
Your Independent HPE Business Technology Community

Read it in **Print** and **Online!**

The Connection

A Journal for the Hewlett Packard Enterprise Business Technology Community

A Guide to High Availability and an Approach to an Active/Active Environment in the OSS World

Not All Business Continuity Solutions Are Created Equal

PLUS

Women in NonStop

May – June 2017 { Volume 38, No. 3 }

A Guide to High Availability and an Approach to an Active/Active Environment in the OSS World

Ki Roth >> Business Development >> Lusis Payments

There's no time for downtime, and it's crucial that your Nonstop hardware platform is at the ready 24/7. Lusis Payments has years of experience with open systems and brought our experience to Nonstop and OSS a few years back. One of the key architecture features of an SOA Nonstop environment relates to reliability and high availability, with the agility and the scalability users need. A versatile design and architecture provides the same high availability Nonstop users are accustomed to for continuous online processing plus Active/Active and Active/Passive environments. This allows payment applications to be streamlined across multiple servers without sacrificing speed or accuracy which creates a robust, reliable platform with guaranteed delivery each and every time.

Some elements of high availability are provided by the architecture of the hardware platform, such as fault tolerance, clustering, and remote backup. Lusis has experience with these different architectures and their respective pros and cons. We have integrated several functions to minimize any constraints introduced by these disparate architectures.

Fault Tolerance

In a fault tolerant system, there are minimum requirements. A) The software must not cause an application to stop either during normal processing or when making frequent, everyday changes to the system, such as adding an ATM, a network interface, a financial institution, or even for version changes or bug fixes. B) The software must make its own provisions through its configuration or architecture for those hardware elements that are not fault tolerant, such as older communication cards.

Cluster with a single Application Database and Application Synchronization

In the case of high availability clusters, the hardware is replicated. The application resides on each server and is either permanently active or automatically activated if the primary server fails. The database is seen as one entity by the application. Stopping software on a server is an event that seriously disrupts its operation: reconnection of ATMs, reconnection of network resources, loss of in-flight transactions. As a consequence, point A) above must be assured as a minimum. Point B) can be assured by a mirrored architecture.

The software must also ensure:

- The transparency of the database, which could be located on multiple nodes
- Central management of technical operations

For application synchronization, each occurrence of the application has its own database, and all instances of the database are synchronized by an application notification mechanism. For the same reasons the software must assure point A) above. The software must also ensure sending of notifications, a guaranteed delivery mechanism, forced posting of notifications and technical management of this message flow.

The SOA Contribution

Alongside various hardware architectures, a properly architected SOA environment provides a number of robust and proven technical approaches to provide software with 'no single point of failure':

- The application can be configured to be highly redundant:
  - Service-level redundancy uses multiple instances of a given Service in an application process.
  - Process-level redundancy uses multiple copies of a given Process, each with one or more instances of a given Service.
- The SOA's modular design allows application processing to be decoupled into functional modules, giving the following advantages:

Each module is simpler and therefore more robust.

- The modules themselves may be replicated either within or across processors to accommodate the throughput required.

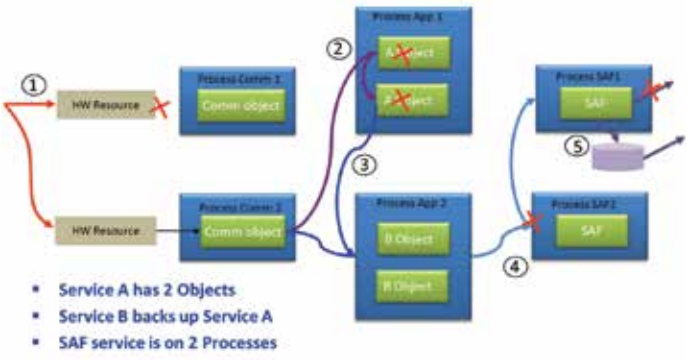
- A High Availability solution supports redundancy at the site level:

- A copy of each transaction can be sent, in the form of a notification message with guaranteed delivery, to the remote site in either an Active/Active or an Active/Passive configuration.

- The technical management of the environment incorporates centralized functions for controlling the entire technical environment:
  - Automatic monitoring of modules
  - Restart commands
  - Dynamic warm-boot commands
  - Stop and start commands by process, by service or by a group of services, to ensure changes can be made without stopping the entire application.

Intra-Application High Availability

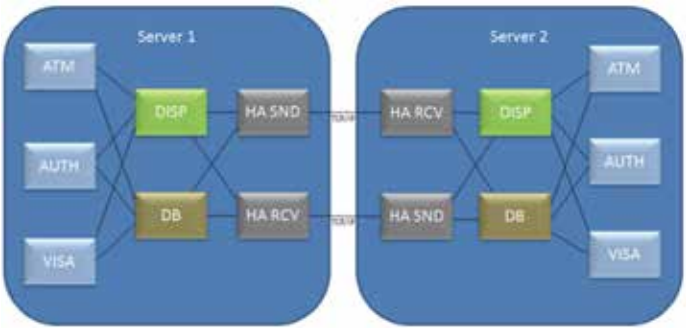
The diagram below presents the inherent software high availability provided in an SOA architecture, as described above, by configuring multiple processes and services and employing an alternative routing capability to ensure maximum opportunity of guaranteeing delivery of a message.



1. System resource becomes unavailable and therefore uses a backup route
2. Routing to the element responsible for a pool of software resources
3. Routing to the backup pool B after an incident in Pool A
4. Sending a notification advice with guaranteed delivery Store and Forward (SAF)
5. SAF the advice then retries if there is a delivery failure

Inter-Application Ultra High Availability

Inter-Application Ultra High Availability can be provided using data replication tools. It can also be accomplished thru the application's own specialized High Availability components integrated within the payments platform to provide an alternative Active-Active solution for two servers, as shown in the diagram below.

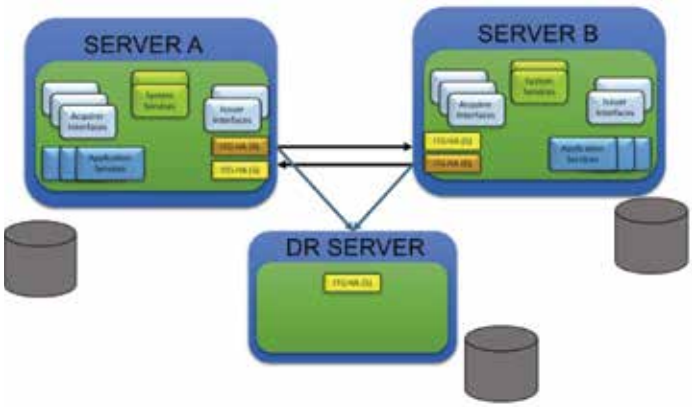


HA SND sends advices to the remote server. HA RCV receives the advices from the remote server. For an Active/Active environment, there is one copy of HA RCV and HA SND on each of the servers. These components are responsible for keeping updated a set of data (tables) on both payment servers. To do this, both payment servers contain two secured communication channels (including SAF capabilities) responsible for exchanging activity advices and status advices.

This additional capability provides the opportunity to streamline current payments platforms as well as reducing the overall TCO for running the authorization system. A SOA based payment platform can be unique by providing both an authorization system and a high availability capability integrated within the same architecture. This will assist both in reducing the ongoing costs and simplifying the management of upgrades as a result of data evolutions.

Inter-Data Center High Availability

In addition to providing the capability for two systems to replicate data between each other in one data center, this mechanism can also be used to replicate data between different systems located in different data centers, as described in the diagram below.



The above diagram shows two local systems replicating data between each other and replicating data to a remote cold/warm standby DR server in another datacenter. The two production systems described could be in different datacenters, providing ultimate flexibility in deploying the system to customers' specific requirements

The previous sections have described the flexibility and configurability that can be designed into an SOA open system application irrespective of the capabilities or configuration of the hardware platform on which it is deployed: on a single node, in a single datacenter or in multiple datacenters.

Implementation of the High Availability components can also provide further flexibility and automation. For example, one option would be to deploy a 3rd cold standby datacenter if required.

Therefore, given the capabilities described above, a properly designed architecture/modules, plus the deployment of Nonstop high availability hardware will provide both the 99.999% availability expected of the system and the required capacity for increasing transaction volumes.

Ultra-High Availability

Ultra-High availability is provided by:

- The capacity to do multi-instances services
- The capacity to implement application services on several systems with the same level of key information
- The capacity to easily provide Active/Active systems.
- The capacity to change configurations without stopping the application
  - Adding processes

- Update services parameters.

A service runs as a thread inside an application process. For each process, the definition of each service contains a parameter defining the number of instances of the service inside the process. Each instance manages its own queue of events (among which are application requests, responses and notifications). To benefit from multi-cores and multi-CPU architectures it is necessary to instantiate more than 1 thread to run per service. Distribution of the message load is the responsibility of a dispatcher process, which maintains counters for all registered service instances.

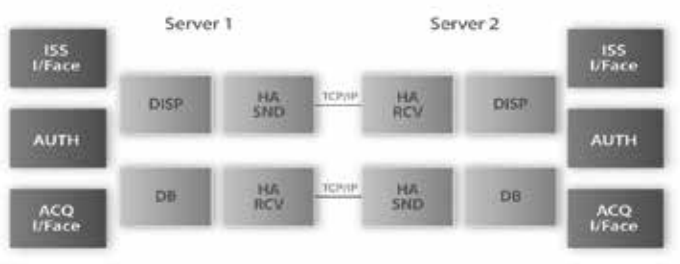
High availability components oversee maintaining an up to date set of data on both payment servers to allow each server to process data flows without any restrictions:

- Authorization data
- Terminal contexts
- Dynamic network and terminal keys

This can be realized without any third-party replication tool. As it is using only messaging features, servers can use different operating systems or even different data models (and databases).

Each server contains at least 2 communication channels in charge of exchanging activity advices (from ATM management, network protocols management and authorization management) and status advices (from ATM management) from one system to the other:

- HASND (High Availability SeNDer process): In charge of sending local system activity advices to remote system.
- HARCV (High Availability ReCeiVer process): In charge of processing remote system activity advices to maintain local database up to date



Both systems are completely symmetrical

The application uses the equivalent of "0120", "0320" and "0820" messages to notify data from one system to another (and vice versa).

Key data elements are:

- Authorization management
  - Hot cards
  - Authorisation activity
  - False PIN
  - True PIN after False
- ATM management
  - Status advices
  - Terminal contexts (counters….)
- Network management
  - Encrypted keys

Time of updating the distant system needs to be within 50 milliseconds. Configurations are synchronized via scripts.

A Real-Life Example

To demonstrate how the above theory is implemented let's take a look at the TANGO architecture from Lusis Payments, although there may be similar solutions available on the market.

For Active/Active

High Availability mechanisms and Interoperability allow simple Active/Active systems integration. RTE package (Remote TANGO Environment) contains the Inter-TANGO process sets.

Authorization activity, terminal contexts (ATMs and POS devices), merchant, and customer data are synchronized through real-time TANGO messages. For instance, any given terminal can connect to any of the Application servers to provide functionalities to customers with 24/7 availability.

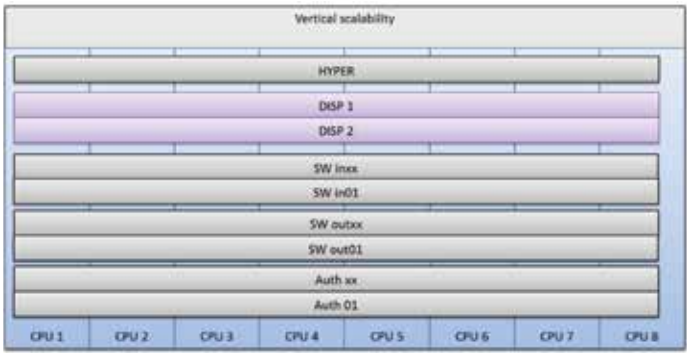
The following schema shows a TANGO fully synchronized DUAL server including an interaction with a remote TANGO batch server that provides safe access to application real-time history at no risk to the real-time transaction flows. This remote server can also host batch extraction activity.

Active redundancy mode

Service is instantiated with a unique name. Several processes host this unique service in parallel. They run on different CPUs and process transactions in load balancing mode, as routing is based on service name. The Hypervisor launches several (minimum of 2) instances of a process at start-up - all are active. The Dispatcher sends transactions over running processes via round-robin with a priority on less loaded processes in the queue. Sizing ensures that if one CPU fails, all the remaining processes can handle the transaction flow.

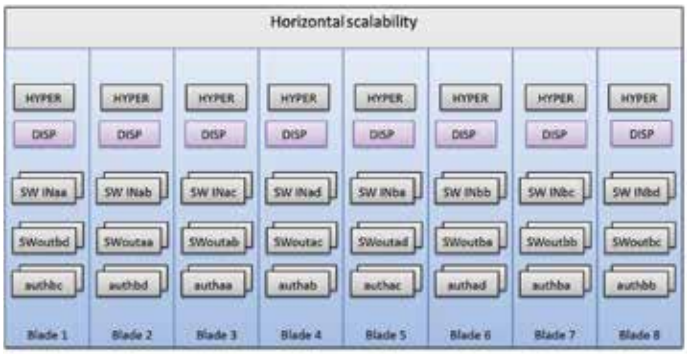
CPU backup mode

The Service is initiated with a unique name. A Unique process is first launched at start time on the first CPU and processes transaction as long as it is present. If a CPU fails, the hypervisor launches the back-up process – configured to run on another CPU. This mode is used when a constraint (external resource, sequence, …) makes use of 'active redundancy' mode impossible. Initial process is launched on one CPU and processes all transactions.



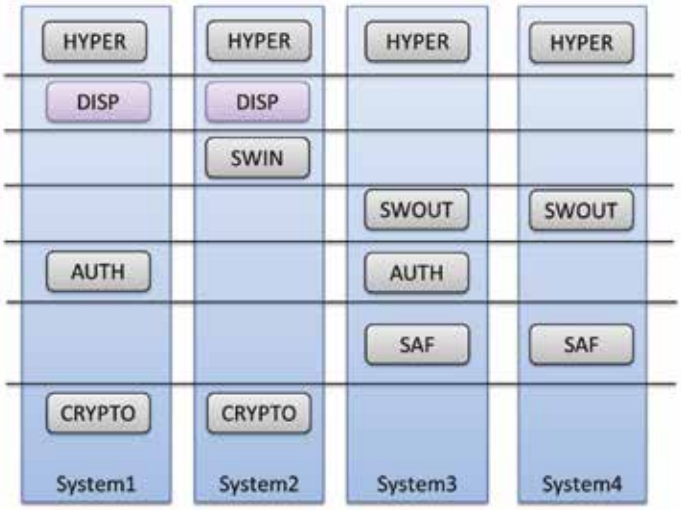
Vertical scalability

Adding CPU and memory in an existing system is a classic way of improving Server capacity without touching the application. All modern Operating Systems can balance process execution on numerous CPUs or CPU cores. Nonetheless, certain OS's may have a very efficient balancing algorithm at the process level but poor balancing at the process threads level.



Horizontal Scalability

New Hardware is now promoting distributed CPU power inside Blade server architectures and high performance interconnection between servers. A solution with high modularity and process organization allows full benefits for such architectures and can seamlessly replicate services on different hardware, either using a centralized Dispatcher registration for services spread over different hardware, or using distributed sub-systems dialog using inter- product functions.

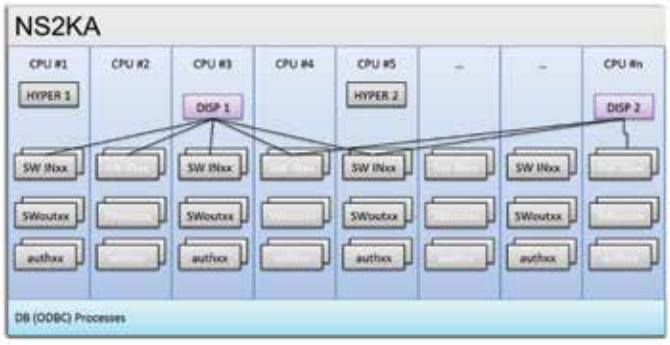


Functional Scalability

All functions being executed in isolated services, the repartition of services on the Hardware and the number of instances that will run the service are highly configurable and can grow by simple configuration changes.

Scalability Example on NonStop

The following schema illustrates the combination of all types of scalability that can be achieved on an HP NonStop with processes spread over logical CPUs executing different sets of services.



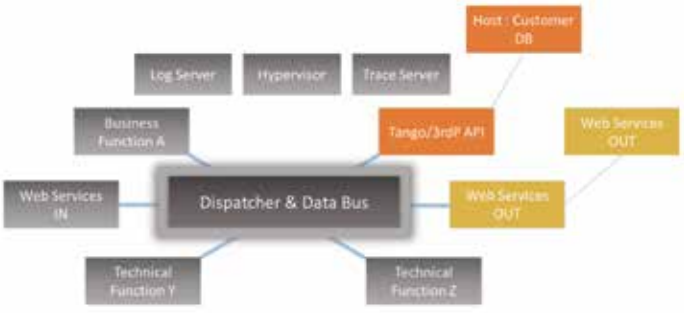
Cooperating with Third Party Applications

Interoperability with third party applications can be managed through integration of APIs or message interfaces when available. In the latter case, this reduces the introduction of a new interface module:

- Providing Perl and Python libraries and C, C++, XML and Java APIs
- Interacting with Java applications by building hybrid Java objects

For example, the high availability solution would interface with the customer database in the Bank's back-end using Web-Service to execute VISA Address Verification Service; or a bank API is used to connect to an external Fraud detection application. The following schema shows how external third party applications can integrate into the services architecture of a high availability solution.

Conclusion



There are many different methods of High Availability and whether to go with an Active/Active environment is a difficult decision to make. At the end of the day each organization must weigh the different factors, some of which are risk assessment, value, resources, time and intrinsic visibility to determine what is best for their situation and ultimately their customer base.

Ki Roth has been with Lusis Payments for over 3 years in a Business Development role. One of his main objectives has been to build awareness for the TANGO solution in the financial sector. Ki has worked in the NonStop space since 1997 when he began working for a large payments software company based in Omaha NE. Over the years, each of his employers have brought solutions that run on the HP NonStop platform. The value that NonStop brings to the market, make it easy for Ki to promote applications that build on the NonStop fundamentals of reliability, stability and high availability. Today TANGO runs on the OSS layer of the platform and uses the SQL/MX database when performing transactional processing.