High Availability at Motorola Flat Panel Display Division

By *Xiaolin Zhuo*

TO THE POINT

August 11, 1997, during my first working day at the Motorola Flat Panel Display Division (FPDD), I found an interesting book named “Cluster for High Availability” by Peter Weygant. One month later, I proposed to my management that our business should implement a high availability architecture (HA) for our Enterprise Systems. The timeline goes like this:

- On October 21, 1997, I finished a three-day class on MC/ServiceGuard, a Hewlett-Packard (HP) product designed to provide high availability. Before yearend, the necessary hardware and software had been purchased to enable our HP system to be configured for high availability. As well, a MC/ServiceGuard consulting agreement with HP was completed.
- By March1998, we had the Oracle8 Database and Oracle Applications NCA10.7 running on a two-node HA cluster.
- By April 1998, the Oracle financial application modules were in production.
- As of this writing, Oracle Financials, Manufacturing, Purchasing, Quality and Sales/Marketing are running in production on this HA cluster.

The Oracle applications we implemented in a HA environment are supporting two separate Motorola business units. There is more, our Document Control System Database and Environmental Hazard Safety Database have also joined this cluster.

Since Feb. 1997, the total unplanned down time for this two-node cluster was less than 1.5 minutes, with a total of three outages. Among these outages, one was a memory board failure (caused HPMC), one was a panic dump due to a HSC 100BASE NIC software problem, and the third was due to a HSC 100BASE NIC failure. The LAN card failure was an interesting one. It did not fail over to the stand-by LAN card, instead, it caused a panic dump. *During the outages, all services were successfully failed over to the surviving node and it took less then 30 seconds in each case.*

PART I: BUSINESS PERSPECTIVES

INTRODUCTION

“With maximum availability no longer an option but a necessity, HP’s award-winning high availability (HA) solutions are helping companies around the globe” -- HP Computer NEWS 12/99

The evolution of UNIX systems during past decade has changed the way IT organizations handle business critical computing. Historically, systems support was addressed from a reactive support perspective. The new paradigm encourages the design of systems that improve the chances of providing proactive support. This evolution is the result of continual dramatic improvements in functionality, reliability, performance, and supportability. Another critical enabler to providing proactive support includes the process of IT Re-engineering. That is, re-engineering of the internal IT support roles, processes, expectations and customer commitment. The IT organization at Motorola Flat Panel Display Division (FPDD) is taking full advantage of both the technology evolution and the IT business process changes necessary to provide the best proactive support for the business needs.

INFRASTRUCTURE

The FPDD team started the IT infrastructure design phase in late 1997. Motorola FPDD was a newly formed business unit at the time so business process re-engineering was not the biggest issue to be addressed. However, the decision for the design of reliable hardware and software infrastructure for the highest levels of availability and performance with no single point of failure was a tough call. Simply put, we needed to justify the cost of a complex highly available system for a new startup, mainly an R&D division at that time. The justification questions included the following:

- Does the system really work?
- Is the customer requirement justified for the complexity and cost involved?
- How can such an environment be supported with very limited IT resources?
- Is the system solution scalable as the business grows?
- How will IT professionals be trained to adapt to this new concept within a short period of time?
- What is the return on investment of building such an elaborate system?

Management was fully aware of the pain of IT process re-engineering. To change the IT infrastructure in the future would mean much more money, resources, and many hours of down time. With this in mind, the management from top down made this call. I am grateful for their vision.

The approach for building a robust HA infrastructure then was to begin to experiment with the HP MC/Servicguard product to establish a training ground and experimental configuration to justify the cost and complexity of the expected system. Since there was minimal industry benchmarks to review, the process included close collaboration with HP to implement the tools quickly to meet the business requirements and address the technological ‘open’ questions.

This high availability infrastructure development project was undertaken in parallel with the traditional ERP project planning methodology.

ERP IMPLEMENTATION METHODOLOGY

The primary challenge was to develop a precise strategy to align people and processes with business needs and to ensure they were able to manage ongoing changes. In order to meet the difficult timeline and cost objectives of the Enterprise Resource Planning (ERP) project, the strategy was defined as the following:

- The project would be managed by Motorola
- Temporary 'expert' resources would be contracted to augment the Motorola team with the objective of assisting in the project, but more importantly conducting 'knowledge transfer' to permanent Motorola team members.
- The project would be *phased* in achievable segments or functionality deliverables.
- No customization would be supported, in all cases were business process could be adjusted to meet the software requirements, this would be the case.
- Standard reporting would be the default. Over time, custom reports would be considered.
- Interface development was limited to critical connections between payment systems, EDI, HR and so on.
- All interface software and documentation should be re-used from Oracle sources or other Motorola locations where possible.

Our approach was to use two implementation sub-teams – one for infrastructure hardware and software, another for ERP software. Our methodology proved some important rules in ERP implementations:

- Make sure your team understands that an effective ERP implementation requires an understanding of "business process", not just servers and software. This must be the fundamental competency of the entire team.
- ERP success isn't all in the software, as a fact, it weights heavily on hardware configuration, database design and tuning. Make sure your hardware specialist is well-versed in current and coming technology and operating systems.
- There is nothing more important than the support from top management. Without the vision and encourage from top management, your ERP project will most likely be unsuccessful.
- Every functional business department must take ownership for their corresponding software modules. They must take initiatives in every step during the implementation.
- A strong project leader is a must. He/she absolutely needs to unite the whole team together and be a diligent task master.
- If at all possible, break the project into smaller well defined chunks in order to achieve some partial success without having to implement the entire software suite at once.

Even with the best planning and methodology, any complex ERP implementation project will have many hurdles to overcome. Most of these challenges will be human but some foundational technological challenges live with the system for a very long time.

CHALLENGES

In today's business environment, your mission-critical applications simply must be available all the time—or your business will pay the consequences in lost productivity, customers, and revenue. You can't afford downtime when your competition is just a click away with e-commerce. Downtime can generate unwanted headlines and significantly affect sales revenue, customer loyalty, and stock price. With an increasing number of companies depending on the Internet and Web-based operations for success, the need for HA systems is becoming increasingly critical. This demand for HA is creating a tremendous dependence on the IT infrastructure, requiring solutions that address every aspect of the computing environment in order to ensure that an enterprise runs efficiently—keeping its customers satisfied and its bottom line healthy.

Delivering end-to-end availability throughout the IT environment including the hardware, operating system, database, application and network is a significant challenge. Anyone who is in the process of selecting or implementing ERP systems will face two common challenges: maintaining an acceptable level of platform availability and managing capacity. Traditionally, IT organizations have solved system outage issues with proprietary mainframe or fault-tolerant solutions; however, they can achieve the same or better results at a lesser cost with open systems. The HA concept was born with that purpose in mind.

WHAT IS AND WHY HIGH AVAILABILITY?

"As a computing environment, HA is a combination of environmental, process, software and computing hardware enhancements that are made to minimize the time that the application and system are not available" – Bob Sauers "Understanding High Availability, 1996".

HA environment is not only a software or hardware implementation. Often, the greatest obstacles to HA computing are not hardware or software failure, but lack of process. In many respects, HA is a mind set as well as a technology that must be adopted to support such environment. Administrative, operational, and application changes must also be made to ensure that the application is available as much as possible. Remember, the human dimensions of a HA system will always be the source of additional failure points.

HA is not a buzzword either. It must be utilized to achieve objectives within the organization and eventually to support goals of your business. A service level agreement must be defined so that your users can expect planned and unplanned downtime within a pre-defined tolerance range. A major goal of HA systems is to provide a higher level of availability than standard (non-HA) systems do, at a cost much lower than fault tolerant systems (10:1 ratio). Note that high availability does not imply fault tolerance.

HA computer systems are designed to eliminate or minimize planned and unplanned outages. The outages are caused by Single Point of Failures (SPOF's) in your computing environment. A single point of failure is a critical hardware or software component that, if it fails, causes an outage of the system or application. The following are the most common causes of outages:

- System Processors
- System Memory
- I/O Controller (I/O Bus)
- LAN Media/Adapters
- SCSI Controller/Adapters
- Power Failure
- File System Full
- Disk Full
- System Processes
- Application Processes
- Human Errors
- Natural Disasters

According to University of Texas (1998), when a company suffers a catastrophic data loss there's only a 6 percent chance of survival. Some 43 percent of companies in this situation never reopen, while another 51 percent close within two years. If you are not convinced yet, here are more industrial examples sorted by outage costs (HP source):

Industry	**Business Operation**	**Industry Cost Range/Hr**	**Avg. Cost/Hr**
Financial	Brokerage Operations	$5.6M to $7.3M	$6.45M
Financial	Credit Card/Sales Authorization	$2.2M to $3.1M	$2.6M
Media	Pay-Per-View	$67K to $233K	$150K
Retail	Home Shopping (TV)	$87K to $140K	$113K
Retail	Home Catalog Sales	$60K to $120K	$90K
Transportation	Airline Reservations	$67K to $112K	$89.5K
Media	Tele-ticket Sales	$56K to $82K	$69K
Transportation	Package Shipping	$24K to $32K	$28K
Finance	ATM Fees	$12K to $17K	$14.5K

PLANNED AND UNPLANNED DOWNTIME

The loss of a service as perceived by users is called an 'outage'. The duration of an outage is generally called 'downtime'. It is important that an IT organization and its user community agree upon the definition of planned and unplanned downtime in a HA environment. "Unplanned downtime" includes the following events when they result from defects in hardware, operating system, middleware, or software:

- No system prompt on any nodes in the cluster (entire cluster is down)
- Network switches unavailable
- Data storage is inaccessible
- Communication with the cluster is severed due to technical causes within the cluster

- System fail-over or switchback is taking place, rendering the operating system temporarily inaccessible
- Inaccessibility to the applications that are protected by HA cluster. (package restart or fail-over errors)

What unplanned downtime does not include:

- When a system prompt is available on at least one node in the cluster.
- Planned downtime
- Data and application recovery time during any unplanned downtime, fail-over, or switchback
- Unplanned downtime due to causes external to the cluster, including, but not limited to:

    - Defects or malfunctions to the network beyond the switches, database, or any applications.
    - User, operator, or support staff error or product misuse.
    - Environmental causes affecting power, temperature, etc.
    - Disasters

Statistics show that outages caused by hardware failure account for 40% of the problems, software 30%, human error 20%, and the balance (10%) is from environmental disasters.

HIGH AVAILABILITY TOOLS

To best meet the requirement of availability, scalability, flexibility, and be cost effective, Hewlett-Packard (HP) has developed a robust cluster architecture for HP-UX that combines multiple systems into a high availability cluster. The core product is MC/ServiceGuard. ContinentalCluster and MetroCluster are additional software feature sets that can be added to MC/ServiceGuard.

MC/ServiceGuard

MC/ServiceGuard is a second-generation HA product that supports multiple applications in a cluster of up to 16 systems (nodes). All nodes in the cluster are active peers. Each can serve one or more applications. When an application running on that node fails, the applications are automatically moved to another node in the cluster configured to run that application. The IP address associated with that application is moved to the adoptive node so that clients can connect to the same IP address. Since the adoptive node does not reboot, it need not stop serving its own applications unless there are resource or performance reasons to do so. In a cluster of more than two nodes, multiple nodes can fail with the surviving nodes taking over for the failed ones. Each LAN adapter can support multiple IP addresses, so the fail-over can occur without requiring other applications to be stopped.

ServiceGuard OPS Edition (previously, MC/LockManager)

ServiceGuard OPS is a HA product that supports one application: Oracle Parallel Server (OPS) in a cluster of up to 4 nodes. All nodes are running the same application(s). When a node fails, the other nodes continue to run the application(s). The surviving nodes not only do not reboot, but they do not startup the application(s) since they are already running. Database recovery automatically occurs for the logs that were in use by the failed node. ServiceGuard OPS also includes the features of MC/ServiceGuard. You can have the user connect to the same IP address rather than a different one, the IP address of the failed node can be assigned to the surviving node. ServiceGuard OPS can also use redundant network interfaces in case of LAN adapter failure.

Since the same application(s) run on all of the nodes in the cluster, ServiceGuard OPS allows share read/write access to the same data disks. It coordinates writes among the nodes through a distributed lock manager that prevents more than one node from modifying the same data at the same time. Contrarily, MC/ServiceGuard only allows exclusive access to the disks.

MONITORING AND MANAGING TOOLS

Managing a HA environment is more complex than managing a single system. Not only are there multiple systems to manage in a coherent manner, but there are new commands for controlling the systems and applications. The following tools from HP are commonly used for managing a HA cluster:

ClusterView

ClusterView is an OpenView application that works with OpenVew/Network Node Manager and OpenView/ITO. It monitors HA clusters, cluster nodes, and application packages status. The graphic maps can show where (on which node) an application package is currently running. However, it does not monitor HA disks – a function realized in HP Event Monitoring Service (EMS).

Event Monitoring Service(EMS)

Event Monitoring Service (EMS) is a system monitoring application designed to facilitate real-time monitoring and error detection for HP products in the enterprise environment. It reports information that helps you detect loss of redundant resources, thus exposing single points of failure and eliminating the threat to data and application availability. And now, EMS capabilities cover the entire system: system components, storage, and network interfaces. You can benefit from EMS as it:

- Enables efficient and effective monitoring of your systems within a single comprehensive framework
- Delivers the ability to tailor the monitoring system to fit your specific needs

- Enables a wide variety of notification methods through multiple protocols (SNMP traps, TCP, UDP, OPC Messaging)
- Provides immediate alerts if a component fails, enabling you to proactively schedule its replacement
- Interacts with MC/ServiceGuard and ServiceGuard OPS (MC/LockManager) to provide a more complete high-availability solution for mission-critical needs
- Improves productivity and application availability by enabling automatic detection of HA disk component failures
- Monitors logical and physical volumes: mirrored copies of data, network interface cards, file system status and number of users

<u>Process Resource Manager (PRM)</u>

PRM allows the system administrator to run mission-critical applications in less time by matching processing needs to CPU resources. System administrators can make intelligent decisions regarding computing resources in order to optimize the processing power available. PRM's many benefits include:

- Managing system resources to ensure that important processes receive maximum resources, thereby limiting processes of lesser importance
- Providing the ability to set processing priorities for users, resource groups, or applications
- Allowing real-time analysis of resource consumption when PRM is integrated with HP GlancePlus
- Using HP GlancePlus to view CPU and real-memory utilization, as well as resource allocations
- Providing optimum performance and resource utilization within an HP high availability cluster when PRM is integrated with HP MC/ServiceGuard or ServiceGuard OPS (MC/LockManager).

SUMMARY: HA CLUSTER DESIGN

In order to implement MC/ServiceGuard, Oracle8 Database, and Oracle Applications NCA10.7 in such a short period of time, there must be something we did right. Recall the procedures and processes we had, we can summarize as:

<u>Management support</u>

Management must realize that high availability is not a luxury but a business requirement. HA not only protects your important enterprise systems, but also create new opportunities for your business. It is IT manager’s job to develop an organization that sees HA as the main priority and that has the skill sets to cope with the demand of HA environment.

Define a Goal for Availability

You must know what you are intended to achieve, otherwise, you are wasting money and resources. Not every organization requires HA environment. But if you do, then a service level requirement for each application or service must be fully evaluated. Compare your existing system, if you have one, see if it can meet the objectives.

Define an Acceptable Down Time Range

High availability costs money. The shorter the down time, more the money you will spend. Define an acceptable down time, and select a best HA architecture to accomplish your goals.

Assess Your Applications

Not every application can be run on HA cluster. For example, an application that uses any system-specific information such as, uname, gethostname, MAC address, or SPU ID, can not be configured in MC/ServiceGuard package. Contact HP support if you are not sure about your application compatibility with MC/ServiceGuard.

MC/ServiceGuard Training

It is very important to train persons who are going to be involved with a HA cluster and application configuration. HP has a three-day class for this topic. This should be a good start. If you configure your database in a HA cluster environment, make sure your DBA obtains necessary training as well. Think about a simple but common scenario: database offline backup. You will never be able to back up your database offline if you don't know how to turn off MC/ServiceGuard monitoring flag. The package eventually will fail over to a surviving node (because MC/ServiceGuard considers your database offline backup as a failure, if you monitor your database processes in this case).

Choosing, Purchasing, and Configuring HA cluster components

Before you decide to purchase HP equipment, it is necessary to have a survey done by HP or a reputable consulting firm. They can give you a recommendation based on your applications, number of users, and network load. It is highly recommended to hire a consultant to configure your HA cluster, if you do not have expertise in your IT organization. However, your application specialists and DBA's must be involved in the configuration. They are the ones who know the applications details. Due to the complexity of the configuration, changing it in the future means more money, resources, and down time.

Define, Configure, and Test Application Package(s)

MC/ServiceGuard protects your application or service by configuring them as packages. You must define what application or service is to be configured as a package, as well as how many packages will be implemented. Other considerations are, which node as a fail-over taker, and how many nodes in a cluster, etc.

The next step is to define what processes are to be monitored. MC/ServiceGuard "pronounces" a package failure based on the "health" of the processes it monitors, not to be confused with other conditions such as CPU, memory, I/O, etc. The solution to this is not magic. A script must be created to start and stop your application(s) within each package as well as define under what condition each application needs to be failed over. This probably is the biggest challenge to face. Another area that requires defining is if it is necessary for package restarts to occur from a local node before failing over to another node. In this case it must be determined if it makes sense for the database to fail over to another node. A question to ask yourself: "What if your DBA killed one of the monitored processes by accident"? Our experience indicates that after three attempts to restart a failed package from a local node, the package should fail over to a surviving node.

Finally, testing is a very important consideration. Think about all possible scenarios, don't forget human errors – one of the biggest threats.

Design and Document Procedures to Handle Failures

Without a procedure to be followed in the event of a failure, your HA environment is only insurance without a policy. For one obvious reason, your HA system is not immune to natural disaster unless you have ContinentalCluster, or MetroCluster in place. Even with this kind of protection, you still need to document procedures to handle failures. For example, after a node failure, the package may run on a surviving node. However, the package will not automatically swing back to the failed node when it rejoins the cluster.

Automating processes to detect errors is an important practice. There are only limited things that MC/ServiceGuard can protect. Certain conditions such as file system full, disk full, etc. are not the responsibility of MC/ServiceGuard, but they could eventually cause your application failure. Worst of all, your MC/ServiceGuard may not react to such conditions at all.

Here are some suggestions:

- Disaster Recovery procedures
- Routine backup with offsite storage
- Resource control on surviving node(s) when fail-over occurs
- Procedures for switching a package back to its original node
- Automated responses to error conditions that could eventually cause outages
- Procedures to handle outages which are not handled by automated responses.

PART II: TECHNICAL IMPLEMENTATION

Included in the next page is a graphical representation of the HA cluster we have running our production applications environment. This example will be used for reference throughout various sections of the technical discussion that makes up the balance of this paper.

Technical Configuration:

The HA Cluster is comprised of the following components:

Hardware

- **HP 9000 Series K460 Server w/ 4 CPU's, 1.5GB RAM**
- **HP Jamaica HA Disk Enclosure w/ 6.3GB Disk Space**
- **Dual HSC FastWide SCSI Adapters connected to Jamaica Enclosure**
- **Dual HSC FastWide SCSI Adapters connected to EMC 3430 Storage**
- **EMC 3430 Symmetrix Disk Storage w/ 1GB cache & 125GB usable space**
- **Dual Port HSC 100/BASE as primary network LAN card (lan1)**
- **HP-PB 100/BASE as stand-by network LAN card (lan3)**
- **HP-PB 10/BASE LAN card for dedicated heart beat (lan0)**
- **Dual Port HSC 100/BASE LAN card for redundant heart beat (lan2)**

Software

- **HP-UX 10.20 Release with patches**
- **Oracle8 and Oracle7 (for server partition)**
- **Mirrored Root Volume Group w/ MirrorDisk /UX**
- **MC/ServiceGuard 10.10 Release w/ patches**
- **Package ERP8P contains production database ERP8P (Oracle8) and Oracle Applications NCA10.7**
- **Package ERP8T contains test database ERP8T (Oracle8) and Oracle Applications NCA10.7**
- **FPDD Customized Oracle8 Database & Oracle Applications NCA10.7 Package Script**
- **RAID 0+1, RAID S (EMC flavor of Raid 5) configuration w/ Pvlinks**

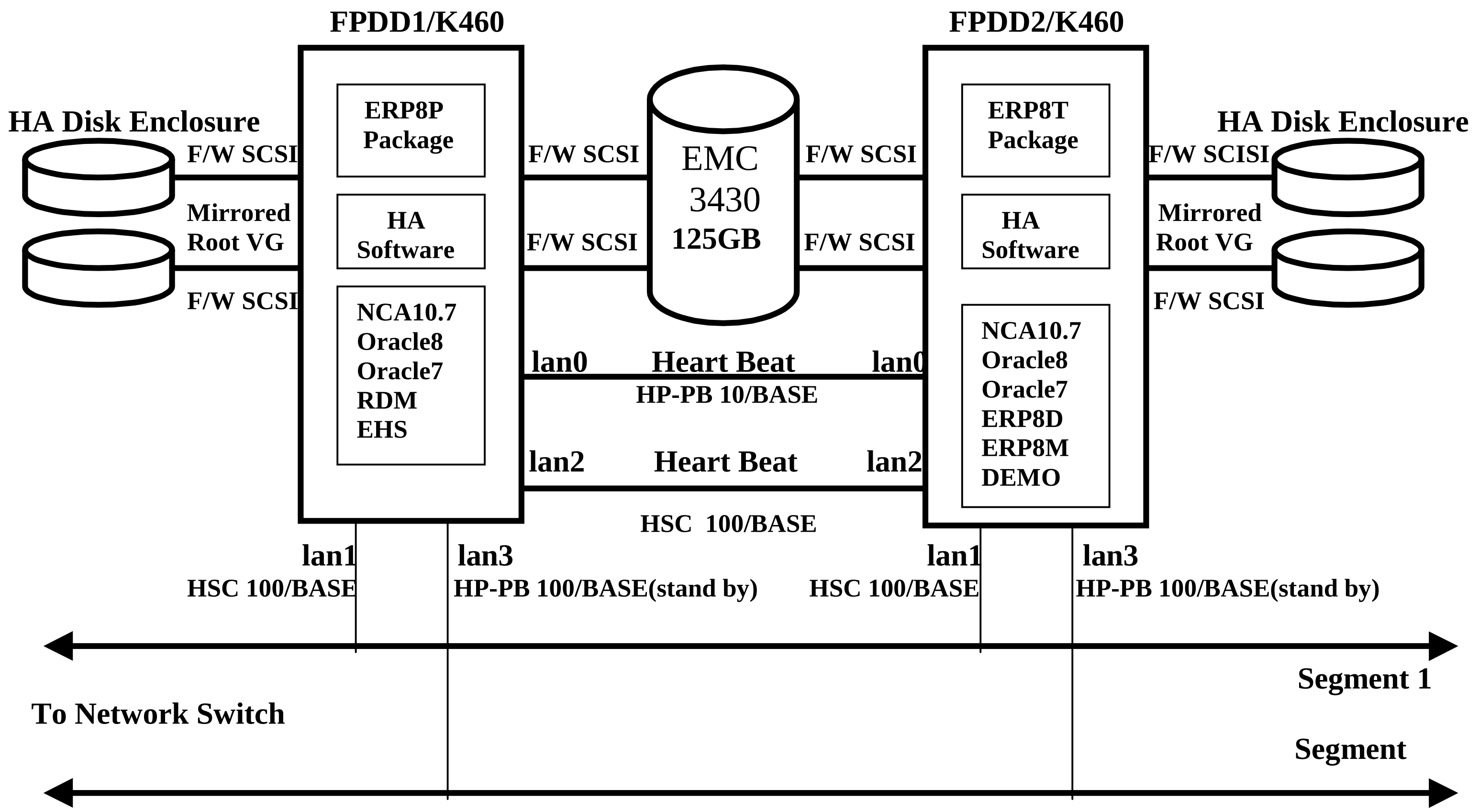
Failure Protection

- **If Node FPDD1 failed, package ERP8P will fail over to Node FPDD2. Users will experience disconnection for less than 30 seconds.**
- **If Node FPDD2 failed, package ERP8T will fail over to Node FPDD1 Users will experience disconnection for less than 30 seconds.**
- **If primary LAN (lan1) card failed, the stand-by lan3 will take over. Note: this action is transparent to users, and there is no disconnection.**
- **If one of the Heart Beats failed, the other Heart Beat handles the communication. There is no impact.**
- **If both Heart Beat connections are broken, one of the nodes will be panic. The other node will form a single node cluster. Both ERP8P and ERP8T packages will run on that node. Note: the panic node will reboot itself and rejoin the cluster. But the package will not swing back automatically**
- **If a package failed, HA cluster will re-start the package from local node. If re-start failed after three tries, the package is then failed over to the other node.**
- **If one of the F/W SCSI Adapter (connect to EMC) failed, the other F/W SCSI will handle the connection via Pvlinks (alternate links)**
- **If Root VG fails (both mirrored disks failed, or disks are removed), what is going to happen?**

SINGLE POINT OF FAILURE

- **Network. FPDD does not have redundant Routers/Hubs.**
- **Oracle Applications NCA10.7 Middle Tier. Middle tiers are not in the HA configuration. They are on NT platform.**

FPDD High Availability Business System Architecture



PREPARE MC/ServiceGuard CONFIGURATION

A successful HA Cluster implementation comes from a sound architecture design and careful preparation. Configuring of MC/ServiceGuard itself is not a difficult task, but the ground work is somewhat critical.

1) MC/ServiceGuard software installation creates entries in /etc/services:

```
hacl-hb      5300/tcp    # High Availability (HA) Cluster heartbeat
hacl-gs      5301/tcp    # HA Cluster General Services
hacl-cfg     5302/tcp    # HA Cluster TCP configuration
hacl-cfg     5302/udp    # HA Cluster UDP configuration
hacl-probe   5303/tcp    # HA Cluster TCP probe
hacl-probe   5303/udp    # HA Cluster UDP probe
hacl-local   5304/tcp    # HA Cluster Commands
hacl-test    5305/tcp    # HA Cluster Test
hacl-dlm     5408/tcp    # HA Cluster distributed lock manager
```

and entries in /etc/inetd.conf:

```
hacl-cfg  dgram   udp  wait    root  /usr/lbin/cmclconfd cmclconfd -p
hacl-cfg  stream  tcp  nowait  root  /usr/lbin/cmclconfd cmclconfd -c
```

2) /etc/lvmrc file modification:

- change from AUTO_VG_ACTIVATE=1 to AUTO_VG_ACTIVATE=0

  When the system reboots, it will not activate any MC/ServiceGuard VG's.

- add non ServiceGurad controlled VG's into custom_vg_activation() section.

  ```
  sleep 1
  /sbin/vgchange -a y -s /dev/vg1_RS_1
  sleep 1
  /sbin/vgchange -a y -s /dev/vg1_R0+1_2
  sleep 1
  /sbin/vgchange -a y -s /dev/vg1_RS_2
  sleep 1
  parallel_vg_sync "/dev/vg00"
  ```

  You don't need to include vg00, it is automatically activated when the system reboots. parallel_vg_sync "/dev/vg00" is for mirrored vg00. It will automatically re-sync vg00 when the system starts. Note: Any future non-

ServiceGurad controlled VG's must be added into this section. Otherwise, the VG's will not be activated when system rebooting.

3) /etc/rc.config.d/cmcluster file modification

   - change from AUTOSTART_CMCLD=0 to AUTOSTART_CMCLD=1

   So the node will attempt to join its cluster automatically when the system reboots. Otherwise, you need to run cmruncl manually to form the cluster.

4) /etc/rc.config.d/cmsnmpagt file modification

   ```
   AUTOSTART_CMSNMPD=1
   ```

   If set to 1, the CM cluster SNMP subagent will be started automatically when the system reboots. Set to 1 if you will be using ClusterView to monitor your HA clusters through OpenView.

5) By default HP-UX allows a maximum of 10 VG, change kernel parameter MAXVGS to a larger number if you need more than 10 VG.

6) VG group minor number and naming issues.

   Assume node FPDD1 failed, MC/ServiceGuard controlled VG's are moved along with package to node FPDD2. If a VG group minor number on FPDD1 is identical to an existing VG group minor number on FPDD2, you have problem. It is important that you have unique group minor numbers for MC/ServiceGuard controlled VG 's across all nodes. It will be easier to add a unique prefix to group minor number for all VG's:

   ```
   FPDD1:  0x010000  -->  0x110000
   FPDD2:  0x010000  -->  0x210000
   ```

   Equally important, you need to create /dev/vg_name/group on all nodes within the cluster. Otherwise, after fail-over, the adoptive node can not activate volume groups.

7) You must make directories for all MC/ServiceGuard controled file systems on every node within the cluster. Otherwise, when a package fails over, the file systems can not be mounted.

Each node must have ALL nodes entry in /.rhosts file. Fail to do this will make it impossible to run any MC/ServiceGuard command. If security is an issue, alternatively, you can create /etc/cmcluster/cmclnodelist file with the entries:

```
fpdd1 root
fpdd2 root
```

8) In order for a package to successfully run on both nodes, you need to carefully prepare which disk/directory/files to fail over with the package.   Following is a list of candidates:

   - any application log/out files
   - any customized directories/files to the application, e.g., MOT_TOP
   - Oracle passwd file, if any ($ORACLE_HOME/dbs/)
   - Oracle redo logs
   - Oracle archive logs
   - Oracle database files
   - concurrent manager log/out directories

10) Duplicate of common files on both nodes

   - package directories (/etc/cmcluster/ERP8*)
   - Oracle Applications home
   - Oracle Database home
   - Oracle Applications concurrent manager related files

   As a general guideline, you should keep application/database home on both nodes. It will be much easier for application/database upgrade in the future. Since you can manually fail over package to the other node while doing upgrade on the local node.

11) Network Consideration

   - Heart beat issues - dedicated or shared, networked or directly connected, etc
   - Dedicated Data traffic or shared with redundant heart beat
   - DNS registration of packages IP addresses (floating IP addresses)

MC/ServiceGuard NETWORK REQUIREMENTS

1) All nodes in the cluster must be on the same subnet (same physical network). If a physical network splits into two segments, they are considered as the same physical network.

2) If a node has multiple **configured** LAN crads, the LAN cards must be on different subnets. Routing protocols get confused when two LAN cards in the same machine are configured for the same subnet.

3) Every node must have at least one heartbeat IP address configured. The heartbeat IP address on each node within the cluster must be on the same subnet.

4) If a separate LAN card is used for a dedicated heartbeat, that IP address must be on a separate subnet from the PRIMARY LAN card (handles data traffic). This is a requirement of routing protocols that become confused when two LAN cards within the same machine are on the same subnet.

5) A LAN card configured as "STATIONARY_IP" is used for data traffic only. This LAN card is monitored by MC/ServiceGuard; however, no heartbeat or cluster communication packets will travel across the card. The term "STATIONARY" indicates the IP address will never leave the node. However, the IP address could failover to a STANDBY card within the node. It is recommended that stationary IP address only be used when there are two or more heartbeat IP addresses configured. It is better to have redundant heartbeats.

6) The STANDBY LAN card must be on the same physical network as the PRIMARY LAN card. This is to allow other systems to continue communication with this local system after PRIMARY LAN card failed to STANDBY LAN card. If this STANDBY LAN card is on a different physical network, other systems can not access this system.

   After the PRIMARY LAN card fails, the STANDBY LAN card will send an ARP broadcast out letting all systems on the network know that they should update their local ARP caches to reflect the new MAC address associated with the IP address previously configured on the PRIMARY LAN card.

7) All node names specified in the cluster configuration file (NODE_NAME) must be host names, they can not be the IP addresses.

CONFIGURE MC/ServiceGuard

Please be advised, I use FPDD1 and FPDD2 two-node HA Cluster as an example. There are differences when you configure a HA Cluster with more than two nodes.

Install MC/ServiceGuard Software and Oracle Database Tool Kit

Installation of MC/ServiceGuard 10.10 is simple. But don't forget to download MC/ServiceGuard patches (PHSS_15531), as well as patches for your NIC software (especially, HSC NIC). I have encountered more troubles on NIC software than anything else. The HP Oracle Database Tool Kit version 10.10 is virtually unusable. Script logic is not working. I ended up re-writing the script. However, some structures of the original script are sound.

Generate Cluster Configuration ASCII File

The best way to create this file is through the command cmquerycl:

- login FPDD1 as root
- cd /etc/cmcluster
- cmquerycl –C cmclconf.ascii –n FPDD1 –n FPDD2

Noticed that you can have any name for this file, here I use cmclconf.ascii. This file should be created in the /etc/cmcluster directory. You don't need this file on every node.

Configure cmclconf.ascii file

The following is the configuration I used in FPDD HA Cluster. You may need to modify parameters based on your architecture.

```
CLUSTER_NAME                 fpdd.cl

FIRST_CLUSTER_LOCK_VG        /dev/vg1_R0+1_1   # Cluster lock VG

NODE_NAME                    fpdd1
    NETWORK_INTERFACE        lan2
      HEARTBEAT_IP           10.1.1.1          # Dedicated heart beat
    NETWORK_INTERFACE        lan3              # Blank, Standby
    NETWORK_INTERFACE        lan0
      HEARTBEAT_IP           15.1.1.1          # Dedicated heart beat
    NETWORK_INTERFACE        lan1
      STATIONARY_IP          222.117.10.1            # Dedicated data
    FIRST_CLUSTER_LOCK_PV    /dev/dsk/c0t0d0   # Cluster lock disk

NODE_NAME                    fpdd2
    NETWORK_INTERFACE        lan2
      HEARTBEAT_IP           10.1.1.2          # Dedicated heart beat
    NETWORK_INTERFACE        lan3              # Blank, Standby
    NETWORK_INTERFACE        lan0
      HEARTBEAT_IP           15.1.1.2          # Dedicated heart beat
    NETWORK_INTERFACE        lan1
      STATIONARY_IP          222.117.10.2      # Dedicated data
    FIRST_CLUSTER_LOCK_PV    /dev/dsk/c0t0d0   # Cluster lock disk
```

```
HEARTBEAT_INTERVAL             2000000          # microseconds
NODE_TIMEOUT                   12000000         # microseconds

AUTO_START_TIMEOUT             600000000        # microseconds
NETWORK_POLLING_INTERVAL       2000000          # microseconds

MAX_CONFIGURED_PACKAGES        4                # up to 30 max

VOLUME_GROUP                   /dev/vg1_R0+1_1   # FPDD1 VG
VOLUME_GROUP                   /dev/vg1_R1_1     # FPDD1 VG
VOLUME_GROUP                   /dev/vg2_R0+S_1   # FPDD2 VG
VOLUME_GROUP                   /dev/vg2_R1_1     # FPDD2 VG
```

Compile and Distribute HA Cluster Binary File

Once cluster configuration ascii file is edited, you need to check syntax for errors:

- login FPDD1 as root
- $ cmcheckconf –C /etc/cmcluster/cmclconf.ascii

You can now generate binary file:

- cmapplyconf –C /etc/cmcluster/cmclconf.ascii

This command will generate /etc/cmcluster/cmclconfig file and automatically distribute the file to other nodes within the cluster.

Anytime the content of cluster configuration file changes, you re-compile to reflect the changes. After you created packages, you also need to re-compile this binary.

Start Cluster

The only command to manually bring up the cluster is cmruncl. You must make sure that there is no active node running within this cluster. Otherwise you may risk corrupted data if you try to start an already started cluster

CONFIGURE MC/ServiceGuard PACKAGES

Now you have a running HA cluster. But it only protects your LAN card (fail over data traffic from lan1 to lan3, had lan1 failed), activate certain VG’s on other node if one of the node failed (CPU, memory, or I/O bus). But what about the applications that you want to protect?

Applications that run in a MC/ServiceGuard HA cluster must be configured with all their related resources (VG's, IP addresses, service processes) into an application called **package.** This is probably the most challenging work you need to deal with, on top of the HA cluster preparation work.

A package must have two files: **package configuration file** and **package control file.** A package configuration file defines the dependency, attributes, and characteristics for the application. While a package control file is responsible for bringing up/down the application(s) and services in MC/ServiceGuard HA cluster. Within a package control file, you can call a customized shell script to accomplish things like startup, shutdown, and monitoring service processes, etc. HP product "Oracle Database Tool Kit", "NFS Tool Kit" are actually customized shell scripts.

Packages must have a unique name within the cluster. A package may contain one or more resources. Each cluster can have maximum 30 packages, and each package can have up to 30 services.

Create a Package Configuration File

You can generate a package configuration file template from cmmakepkg command:

- login as root on FPDD1
- mkdir /etc/cmcluster/ERP8P   # create directory for your package
- cd /etc/cmcluster/ERP8P
- cmmakepkg –p erp8p.conf   # you can give any name

This file is created at /etc/cmcluster/ERP8P/erp8p.conf. You don't have to copy this file to FPDD2 after the file is customized. It is only needed when generate HA cluster binary file. Package erp8t.pkg on FPDD2 will not be discussed here, it is similar to the configuration of erp8p.pkg.

Customize Package Configuration File

The following is from erp8p.pkg package configuration file. You may need to modify parameters based on your application requirement.

```
PACKAGE_NAME                    erp8p.pkg       # must be unique within a cluster

NODE_NAME                       fpdd1   # The order is important, the original node first
NODE_NAME                      fpdd2    # The first adoptive node second, etc…..

RUN_SCRIPT                     /etc/cmcluster/ERP8P/erp8p.cntl
RUN_SCRIPT_TIMEOUT             300      # in seconds
HALT_SCRIPT                    /etc/cmcluster/ERP8P/erp8p.cntl
HALT_SCRIPT_TIMEOUT            600      # in seconds

SERVICE_NAME                   erp8p.ser   # must be the same as in erp8p.cntl
SERVICE_FAIL_FAST_ENABLED      NO       # Service failure will not cause TOC panic
SERVICE_HALT_TIMEOUT           200      # in seconds

SUBNET                         222.117.10.0       # must correspond to pkg control script

PKG_SWITCHING_ENABLED          YES      # Allow the package fail-over to other node

NET_SWITCHING_ENABLED          YES      # Allow IP address switch to Standby LAN card

NODE_FAIL_FAST_ENABLED         NO       # Node failure will not cause TOC panic
```

Only one package control script per package is allowed. Please be advised that the timeout limit must be long enough for run and halt to finish. Otherwise, when a timeout happens, the package will be precluded from execution on any other node (global switching is disabled).

Create a Package Control Script

The package control file is called by the HA cluster daemon cmcld. Each package must have a package control script, and it must reside on all nodes that would run this package. You can get a template from cmmakepkg command:

- login FPDD1 as root
- cd /etc/cmcluster/ERP8P
- cmmakepkg –s erp8p.cntl

This file is created at /etc/cmcluster/ERP8P/erp8p.cntl. Don’t forget to copy this script to FPDD2, after it is modified.

Customize Your Package Control Script

A package control script is the heart of a package. Due to its size, I only list part of contents from our example. Although this script is big, fortunately, you don’t need to modify every line of code. The examples shown here are the ones that you need to customize. It is a good practice to use double quotes for your parameter values. Double quotes are required if there is a space between.

```
VG[0]="vg1_R0+1_1"                        # FPDD1 VG for this package
VG[1]="vg1_R1_1"                          # FPDD1 VG for this package

LV[0]="/dev/vg1_R1_1/lv1_redo1";          FS[0]="/u1_redo1"      # lv and fs
LV[1]="/dev/vg1_R0+1_1/lv1_dbf1";         FS[1]="/u1_dbf1"       # lv and fs
LV[2]="/dev/vg1_R1_1/lv1_arch1";          FS[2]="/u1_arch1"      # lv and fs

IP[0]="222.117.10.10"                     # package floating IP address
SUBNET[0]="222.117.10.0"                  # must correspond to pkg configuration file
SERVICE_NAME[0]="erp8p.ser"               # name must be identical as in erp8p.conf
SERVICE_CMD[0]="/etc/cmcluster/ERP8P/erp8p.sh monitor"           # full path required
SERVICE_RESTART[0]="-r 3"                 # restart package 3 times before failover

# START OF CUSTOMER DEFINED FUNCTIONS

# This function is a place holder for customer define functions. You should define all actions you want to
# happen here, before the service is started.  You can create as many functions as you need.

function customer_defined_run_cmds
{
# ADD customer defined run commands.

        set -m
        /etc/cmcluster/ERP8P/erp8p.sh start
        set +m
        test_return 51              # call test_return if “erp8p.sh start” fails
}

# This function is a placeholder for customer defined functions. You should define all actions you want to
# happen here, before the service is halted.

function customer_defined_halt_cmds
{
# ADD customer defined halt commands.

        set -m
        /etc/cmcluster/ERP8P/erp8p.sh halt
        set +m
        test_return 52              # call test_return if “erp8p.sh halt” fails
}
```

SERVICE_NAME=erp8p.ser is the name of the service process for which the package will be dependent. You can have from 0 to 30 services for each package. Service names must correspond to the service names used in package configuration file. Service names must be unique in the cluster. A service can only belong to one package. Each service has a service command, as defined in SERVICE_CMD. In the above example, service erp8p.ser is associated with command “erp8p.sh monitor”. This command is spawned by MC/ServiceGuard command cmrunserv. Its PID will be monitored by MC/ServiceGuard every 10 seconds. If the PID exits or terminates, MC/ServiceGuard (Service sensor, part of cmcld daemon) responds with a service failure and takes appropriate actions. In our configuration, the package erp8p.pkg will fail over to the other node.

erp8p.sh is a customized script, its purpose will be discussed in a later section. Erp8p.cntl calls “erp8p.sh start” before starting service process erp8p.ser. If “erp8p.sh start” fails, test_return function is called with parameter 51. The test_return then do the following:

```
print "\tERROR:  Function customer_defined_run_cmds"
print "\tERROR:  Failed to RUN customer commands"
halt_services
customer_defined_halt_cmds
disown_dtc
remove_ip_address
umount_fs
deactivate_volume_group
exit 1
```

Before cmcld halt erp8p.pkg package, it calls “erp8p.sh halt” to stop applications. If “erp8p.sh halt” fails, test_return function is called with parameter 52. The test_return then do the following:

```
print "\tERROR:  Function customer_defined_halt_cmds"
print "\tERROR:  Failed to HALT customer commands"
exit_value=1
```

Since application halt failed, cmcld can not halt the package. This is considered as halt package failure. Basically, the package is disable for global switching.

Reasons to halt a package:

- subnet failure
- service failure
- cmhaltpkg is issued
- cmhaltnode is issued
- cmhaltcl –f is issued

After a package is halt, its resources are cleaned up.

Generate HA Cluster Binary File

It is time to generate a new binary for your cluster:

- login FPDD1 as root
- cd /etc/cmcluster
- cmcheckconf -v -C cmclconf.ascii –P /etc/cmcluster/ERP8P/erp8p.conf
- cmapplyconf -v -C cmclconf.ascii –P /etc/cmcluster/ERP8P/erp8p.conf

Again, this binary file is automatically distributed to all nodes within the cluster.

MC/ServiceGuard Online Reconfiguration

Starts MC/ServiceGuard version 10.06, you are able to configure cluster online:

- change cluster configuration while cluster is running
- online package change
  - add a package
  - delete a package
  - modify some package attributes while package is running:
    node failover order, remove a node for the package to run, add a node for the package to run, package timeouts, package switch and failfast parameters.
  - modify entire package while cluster and other packages continue to run, only the package to be modified will be down:
    add or remove services, add (subnet must be already configured in the cluster) or remove subnet, add or remove resources.

- online node change
  - add a node
  - remove a node
- issue cmapplyconf while cluster is running (MC/ServiceGuard 10.10)

Oracle Database and Oracle Applications Toolkit

Why do I need a toolkit for Oracle Database and Oracle Applications? Why don’t I configure them as service processes in a package control script? Before answering these questions, lets explore more on the service processes.

Service processes are defined in both the package configuration file and the package control script. Within the package control script, the parameter SERVICE_CMD contains the full path name of the program/command to run as the service process. The service process is spawned by cmrunserv command. The resulting process ID is then monitored by cluster daemon cmcld to determine the health of the process. For any reason, if this PID is terminated, MC/ServiceGuard will consider this as a service failure.

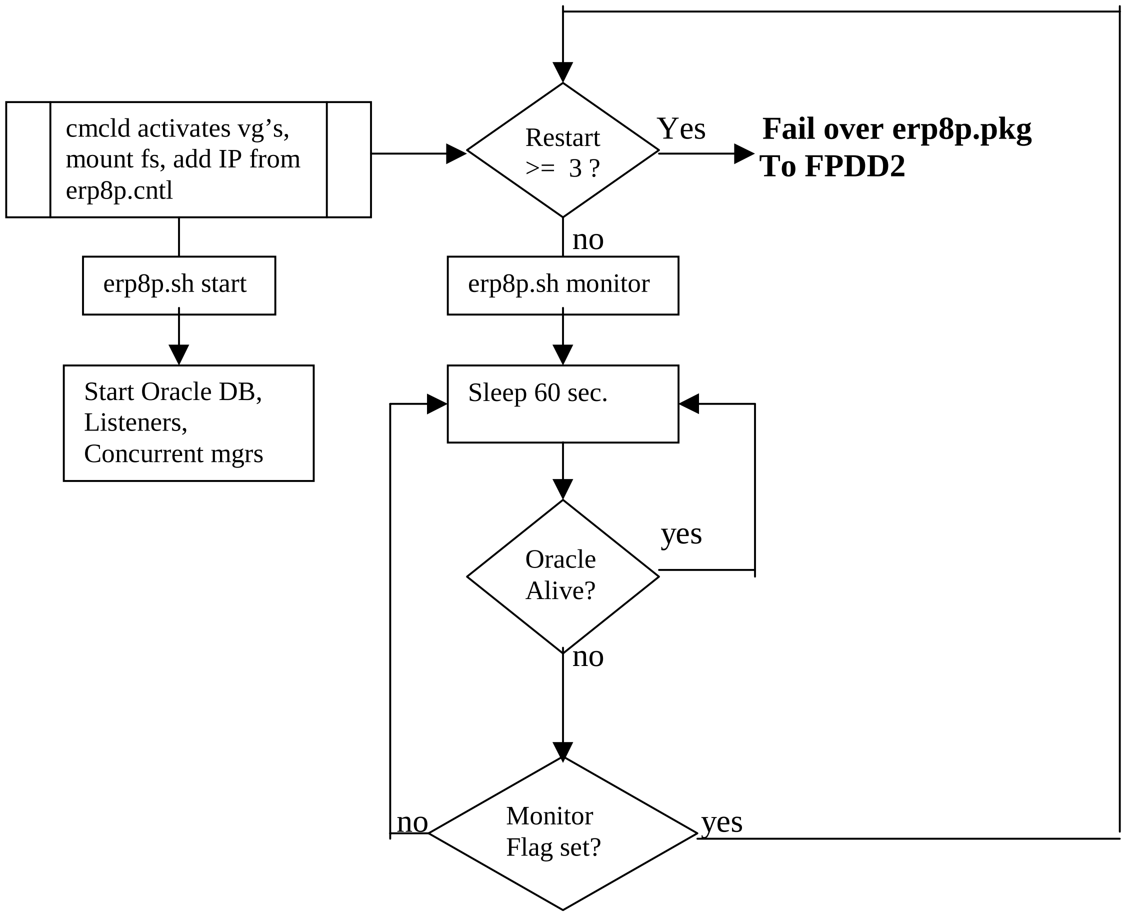
Now lets take a look of Oracle database processes. You start a database either through a script called dbstart, or through svrmgrl. In either case, it creates several Oracle database processes. Does MC/ServiceGuard cluster daemon cmcld care about these processes? cmcld only monitors dbstart(or svrmgrl) PID, not these database processes! Guess what happens next, after the database starts, dbstart (or svrmgrl) process is terminated. Immediately, cmcld will "announce" a service failure. Can you define dbstart (or svrmgrl) as a service in a package?

How can we resolve this problem then? Basically we need a process that can stay alive if Oracle database processes are alive. What about creating a separate process, and let this process decide the health of Oracle database processes? If any of the Oracle database processes (you can define them, by the way) dies, this process terminates. What a great idea! Recall in erp8p.cntl script, there is a service "serp8.ser" with a service command defined as SERVICE_CMD="/etc/cmcluster/ERP8P/erp8p.sh monitor". erp8p.sh script takes care of starting, halting and monitoring the Oracle Database and Oracle Applications Concurrent Managers.

A major advantage of using a monitoring script is the flexibility that can be used to monitor your applications. You can add check lists like file system full, errors in database, etc. It is not necessary to stop your package for a non-major error. You can fix these minor errors within the monitoring script without even telling MC/ServiceGuard!

The rule is, MC/ServiceGuard watches your monitoring script, and the monitoring script watches all aspects of your applications.

Customized Script erp8p.sh Design Logic


cmcld activates vg's, mount fs, add IP from erp8p.cntl
erp8p.sh start
Start Oracle DB, Listeners, Concurrent mgrs
Restart >= 3 ?
Yes
Fail over erp8p.pkg To FPDD2
no
erp8p.sh monitor
Sleep 60 sec.
Oracle Alive?
yes
no
Monitor Flag set?
no
yes

How does erp8p.sh work?

1) cmcld/cmassistd calls erp8p.cntl script, activate VG's that belongs to erp8p.pkg, mount file systems, add floating IP adresses, etc.
2) from erp8p.cntl, it calls "erp8p.sh start". MC/ServiceGuard gives you an option to run some programs before cmcld starts a "service". You can use this opportunity to start your real services, i.g., Oracle database, etc.
3) "erp8p.sh start" calls run_cmds function to start Oracle Database, Oracle TNS listeners, and Oracle Applications concurrent managers. Then it returns to erp8p.cntl

4) erp8p.cntl now starts erp8p.ser service with command "erp8p.sh monitor". The process ID from "erp8p.sh monitor" is monitored by cmcld every 10 seconds.
5) "erp8p.sh monitor" calls monitor_processes function – an infinite loop that exits only when certain conditions are met. The conditions are: failure of any Oracle database processes while monitor flag is set. The monitor flag is important. If you want to do any database maintenance that involves a shutdown, you must unset the monitor flag. Otherwise, cmcld will consider a service failure and fail over the erp8p.pkg package.
6) If any of Oracle database processes is down, erp8p.sh then checks for the monitor flag. If the flag is set, "erp8p.sh monitor" process is terminated and returns to erp8p.cntl.
7) Now cmcld knows erp8p.ser service is dead, for whatever reasons. Next it checks SERVICE_RESTART parameter in erp8p.cntl (in our case, it is set to 3). If the restart counter is less than 3, cmcld calls "erp8p.sh start" again. Correctly setting SERVICE_RESTART is a crucial decision that you need to make. Do you really want your package to fail over to the other node after the first failure of the service? In most cases, the answer is no. You probably want to re-start failed service on local node until it fails for a few times.
8) If the restart counter reaches 3, erp8p.cntl calls "erp8p.sh halt" to stop the Oracle database and Oracle Applications concurrent managers. MC/ServiceGuard gives you another chance to run any program before it halts your package. In our case, "erp8p.sh halt" calls the abort_cmds function.
9) cmcld then releases all resources that belong to erp8p.pkg package, e.g., unmount file systems, deactivate VG's, and release floating IP address, etc.
10) MC/ServiceGuard fails over package erp8p.pkg to node FPDD2.

It is not my intent to analyze the entire erp8p.sh script. This script should be highly customizable to fit your needs. The above steps served well for our case, but it may not be an ideal solution in your environment. This is the area that you need to pay more attention. As I said before, this is the most challenging work in configuring MC/ServiceGuard. Following are the functions that were mentioned above:

Set Variables:

```
PACKAGE_NAME=erp8p.pkg              # The name of the package as defined in erp8p.cntl.

export SID_NAME=ERP8P               # Define your database name

# Location of monitor flag file
SOURCE=/etc/cmcluster/ERP8P/CM_MONITOR
RESULT=/etc/cmcluster/ERP8P/CM_MONITOR.result

case $(hostname) in
```

```
fpdd1)
  # Define your local ORACLE_HOME
  export ORACLE_HOME=/u1_ora1/app/oracle/product/8.0.3

# Define Oracle Application Concurrent Manager start/stop scripts
  export START_CONC_MGR=/u1_app1/applmgr/admin/start_$SID_NAME.sh
  export STOP_CONC_MGR=/u1_app1/applmgr/admin/stop_$SID_NAME.sh
  ;;

fpdd2)
  # Define your local ORACLE_HOME
  export ORACLE_HOME=/u2_ora1/app/oracle/product/8.0.3

  # Define Oracle Application Concurrent Manager start/stop scripts
  export START_CONC_MGR=/u2_app1/applmgr/admin/start_$SID_NAME.sh
  export STOP_CONC_MGR=/u2_app1/applmgr/admin/stop_$SID_NAME.sh
  ;;
esac

# Mail list where you want messages sent to
MAIL_LIST='x.zhuo@motorola.com'

# Define what listener.  For sqlnet 2, use LISTENER=lsnrctl, or sqlnet 1, use LISTENER=tcpctl
LISTENER=lsnrctl

# Define your database startup pfile
PFILE=$ORACLE_HOME/dbs/init$SID_NAME.ora

# Set an array to contain the names of all database processes which you want to monitor.
set -A MONITOR_PROCESSES ora_smon_${SID_NAME} ora_pmon_${SID_NAME} \
ora_lgwr_${SID_NAME} ora_dbwr_${SID_NAME}

# time in seconds, this script should wait between checks of the Oracle database processes.
MONITOR_INTERVAL=60

# When halt package, the database shutdown is in progress.  If for any reason that the monitored
#processei(s) are still running, SIGKILL is sent to the monitored Oracle database processe(s) after the
#TIME_OUT(in sec) is reached The value of TIME_OUT must be less than the time out variable set in the
# package configuration file. The time out variable does not impact any failover times, it is only a fail safe.
TIME_OUT=300

# Check svrmgrl or sqldba
SVRMGRL=svrmgrl
su - oracle -c "$SVRMGRL <<EOF
exit
EOF" 1>/dev/null 2>&1

if [ $? != 0 ]; then
SVRMGRL="sqldba lmode=y"
fi
```

run_cmds:

```
function run_cmds
{

  if [[ ! -f ${PFILE} ]]
  then
    print "ORACLE: The file ${PFILE} does not exist."
    print "\tERROR:  Failed to start Oracle database"
    exit 1
  fi

# start database
su - oracle -c "export ORACLE_SID=$SID_NAME; $SVRMGRL <<EOF
connect internal
startup pfile=${PFILE}
exit
EOF" 1>$TMP_FILE 2>&1

  grep ORA- $TMP_FILE >/dev/null
  if [[ $? = 0 ]]
  then
    su - oracle -c "export ORACLE_SID=$SID_NAME; $SVRMGRL <<EOF
connect internal
startup force pfile=${PFILE}
exit
EOF" 1>$TMP_FILE 2>&1

     grep ORA- $TMP_FILE >/dev/null
     if [[ $? = 0 ]]
     then
     print "\nOracle startup failed with error:\n"
     cat $TMP_FILE
     else
     print "\nOracle startup done!"
     fi

  else
    print "\nOracle startup done!"
  fi

# Start Oracle Listener

  su - oracle -c "$LISTENER start" 1>/dev/null 2>&1
  su - oracle -c "$LISTENER status" 1>/dev/null 2>&1

  if [[ $? != 0 ]]
  then
     print "\nOracle Listener failed to start"
  else
     print "\nOracle Listener started!"
  fi

# Start Oracle Application Concurrent Manager

#set -m
print "\nStarting Oracle Concurrent Manager....\n"
nohup su - applmgr -c "$START_CONC_MGR" 1>/dev/null 2>&1 &
#set +m
}
```

abort_cmds:

```
function abort_cmds
{
# The command below is this scripts calling itself with the kill option to kill any process that will not die
#normally after waiting for the TIME_OUT period

#  set -m
  ${0} kill &
#  set +m

# Stop Oracle Application Concurrent Manger

#set -m
print "\nStopping Oracle Concurrent Manager...."
su - applmgr -c "$STOP_CONC_MGR" 1>/dev/null 2>&1
#set +m

# Shutdown abort Oracle database instance
su - oracle -c "export ORACLE_SID=$SID_NAME; $SVRMGRL <<EOF
connect internal
shutdown abort
exit
EOF" 1>/dev/null 2>&1

  if [[ $? != 0 ]]
  then
    print "\nOracle shutdown abort failed!"
  else
    print "\nOracle shutdown abort done!"
  fi


# Make sure all processes have gone away before saying shutdown is complete. This stops the other node
#from starting up the package before it has been stopped and the file system has been unmounted.

# The variable "c" is used to step through the array.
```

```
  typeset -i c=0

  while true
  do
    for i in ${MONITOR_PROCESSES[@]}
    do
      ps -ef | grep ${i}$ > /dev/null
      if [[ $? != 0 ]]
      then
        print "\n *** ${i} process has stopped! ***\n"
        unset MONITOR_PROCESSES[$c]
        (( c = c + 1 ))
      fi
    done

    if [[ ${MONITOR_PROCESSES[@]} = "" ]]
    then
      exit
    fi

    c=0
    sleep 5
  done
}
```

monito_processes:

```
function monitor_processes
{
  sleep ${MONITOR_INTERVAL}

# The variable "n" is used to step through the array.

  typeset -i n=0

  while true
  do
    while true
    do
      for i in ${MONITOR_PROCESSES[@]}
      do
        Source_Monitor
        ps -ef | grep $i$ >/dev/null
        if [[ $? != 0 ]]
        then
          sleep 2
          ps -ef | grep $i$ >/dev/null
          if [[ $? != 0 ]]
          then
          break 2       # Two breaks needed to skip the rest of the for loop.
        fi
      fi
    done
    sleep ${MONITOR_INTERVAL}
  done

  Source_Monitor

  case $CM_MONITOR_flag in
    1)
          # Send Mail message from here.
          Monitor_Exit
      ;;
    0)
          # echo "ENTERING CASE 0"
          # Send Mail message from here. but send only once.

          # Do not exit
          # Return and repeat the application monitor code
      ;;
    *)
       print "\n error in $SCRIPT_NAME script `date +%x\ %X"
      ;;
   esac
 done
}
```

MANAGING MC/ServiceGuard HA CLUSTER

Managing MC/ServiceGuard is an equally important practice. There is nothing magic about MC/ServiceGuard. It listens to you and you make it work for you. MC/ServiceGuard logs errors in two files: /var/adm/syslog/syslog.log and /etc/cmcluster/pkg_dir/pkg.cntl.log. You need to monitor these logs constantly. You can write a cron job to catch cm* messages in syslog.log, and send email with pkg.cntl.log file every time it is updated.

Understand MC/ServiceGuard Commands

1) Monitor cluster and package status

If your cluster is started successfully, you should be able to get the following status by issuing cmviewcl –v command on any node of the cluster:

```
CLUSTER     STATUS
fpdd.cl          up

            NODE     STATUS      STATE
            fpdd1       up              running

                        Network_Parameters:
                        INTERFACE   STATUS      PATH        NAME
                        PRIMARY      up              8/12/2/0     lan2
                        PRIMARY      up              10/12/6       lan0
                        PRIMARY      up              8/12/1/0     lan1
                        STANDBY     up              10/4/16       lan3

                        PACKAGE     STATUS      STATE       PKG_SWITCH   NODE
                        erp8p.pkg       up              running       enabled              fpdd1

                                    Script_Parameters:
                                    ITEM       STATUS  MAX_RESTARTS  RESTARTS   NAME
                                    Service    up             3                          0                erp8p.ser
                                    Subnet     up                                                           222.117.10.0

                                    Node_Switching_Parameters:
                                    NODE_TYPE    STATUS       SWITCHING    NAME
                                    Primary            up               enabled            fpdd1       (current)
                                    Alternate          up               enabled            fpdd2

            NODE        STATUS      STATE
             pdd2          up               running

                        Network_Parameters:
                        INTERFACE   STATUS      PATH        NAME
                        PRIMARY       up              8/12/2/0     lan2
                        STANDBY      up              10/4/16      lan3
                        PRIMARY       up              8/12/1/0     lan1
                        PRIMARY       up              10/12/6       lan0

                        PACKAGE     STATUS      STATE       PKG_SWITCH   NODE
                        erp8t.pkg        up              running       enabled              fpdd2

                                    Script_Parameters:
                                    ITEM       STATUS  MAX_RESTARTS  RESTARTS   NAME
                                    Service    up             3                          0                erp8t.ser
                                    Subnet     up                                                           222.117.10.0

                                    Node_Switching_Parameters:
                                    NODE_TYPE    STATUS       SWITCHING    NAME
                                    Primary            up               enabled            fpdd2       (current)
                                    Alternate          up               enabled            fpdd1
```

2) Stop package erp8p.pkg on fpdd1, and restart it on fpdd2

```
cmhaltpkg –n fpdd1 erp8p.pkg
cmmodpkg –e erp8p.pkg
```

At this point, erp8p.pkg should start on node fpdd2. If not, you then do the following:

```
cmmodpkg –e –n
cmrunpkg –n fpdd2 erp8p.pkg
```

3) How to halt a node?

```
cmhaltnode –v –f node_name
```

When you halt a node, the cluster daemon cmcld is terminated. All packages run on this node are aborted or failed over to other nodes. If packages fail to halt, the node halt operation is failed too. –f option is to force node halt, even if there are packages running on the node.

4) How to halt a cluster?

```
cmhaltcl –v –f
```

Halt a cluster will halt all cluster daemons on every node. –f option is to force cluster halt, even if there are packages running on nodes. If packages failed to halt, the halt cluster operation is failed too.

5) What to do if a reboot does not form a cluster?

You need to make sure that two nodes are starting within 10 minutes time difference. This time limit is defined in cmclconfig.ascii (AUTO_START_TIMEOUT). Next, verify cluster status by command “cmviewcl –v”. If the cluster is “down” or “unknown”, run cmruncl to manually form a cluster. Please note, if your cluster is running on any node, you can not run cmruncl. This may cause data integrity corruption.

If one of the nodes panic, the other node should form a cluster by itself. After the failed node rebooting, it should joint the cluster automatically. However, the package will not swing back to this node. MC/ServiceGuard prevents this happening for the reason that the failed node may have serious problem. It is your responsibility to fix problems, if any. Then manually swing back package at your wish (see procedure 2).

6) What would happen if heartbeat fails?

If heartbeats fail, whoever grabs the lock disk (two nodes cluster) will form a single node cluster. Both erp8p.pkg and erp8t.pkg will run on the "winner" node. The "loser" node will panic dump. Again, after rebooting, panic node joins the cluster. But the package will not swing back.

7) How to get cluster environment information?

- cmscancl

  This command will gather system software/hardware information from all nodes within the cluster. Including, LAN, network configuration and status, LVM configuration, file systems, info from cluster binary file, package configuration info, etc. The default output file is /tmp/scancl.out.

- cmviewconf

  This is a simplified version of cmscancl. It only shows cluster, node, and package configuration info.

- cmgetconf

  This command can get cluster and package configuration info into an output file. This info is based on your current running cluster and package. You can use the output file as a template.

Modify an Existing VG Group Minor Number

As I mentioned before, an MC/ServiceGuard controlled VG group minor number must be unique among all nodes within the cluster. This section describes the steps necessary to convert a VG group minor number so that it is unique. Assume you have a VG called vg01 on FPDD1 with minor number 0x010000. You want to change it to 0x110000.

1. unmount the file system attached to vg01
2. deactivate vg01: vgchange -a n vg01
3. create a map file for vg01: vgexport -p -m /tmp/vg01.map vg01
4. export vg01: vgexport vg01
5. Create a new group file with minor number 0x110000:

```
mkdir /dev/vg01
mknod /dev/vg01/group c 64 0x110000
```

6. import vg01: vgimport -m /tmp/vg01.map vg01 /dev/dsk/c*t*d*

Note: Include all disk devices that belong to vg01, including PVlinks.

7. Duplicate vg01 to FPDD2

   - copy map file (vg01.map) to FPDD2
   - make new /dev/vg01/group file on FPDD2
   - import vg01 on FPDD2 via vg01.map file

   Note: you don't need vg01 present on FPDD2, while doing the above steps.
   You will get some warnings though, but it is ok.

8. Activate vg01 on FPDD1: vgchange -a e vg01

9. mount file system belongs to vg01

Modify a non-MC/SG VG to a MC/SG VG

From time to time you may want to convert an application that was not controlled by MC/ServiceGuard, to an existing package, or a new package.

1. unmount file systems that belong to the non-MC/ServiceGuard VG
2. deactivate the VG: vgchange -a n vg_name
3. make the VG for MC/SG: vgchange -c y vg_name
4. Create VG's map file: vgexport -p -m vg.map vg_name
5. Create /dev/vg_name/group file on 2nd node. The minor/major number must be the same as of /dev/vg_name/group file on the 1st node.
6. Copy vg.map file to 2nd node then import it:

   ```
   vgimport -m /tmp/vg.map /dev/dsk/c?t?d?
   ```

   Note: include all devices that belong to the VG, including PVlinks.

7. Activate the VG on 1st node in exclusive mode: vgchange -a e vg_name
8. Mount the file systems on 1st system.
9. Delete this "old" VG from /etc/lvmrc file on 1st node. **Important!!!**
10. Include this VG in package control file so when the package starts, it will activate the VG in exclusive mode. The control file must be updated on both systems. Don’t forget to compile a new binary file.

Change MC/ServiceGuard controled file system size

If you don’t have Online JFS on your cluster environment, you need to follow the steps:

1) Turn off package monitor flag

1) Stop application that uses the file system, but do not shutdown the package.

3) Increase LV size using lvextend

3) fuser –c /file_system to see any processes still using the file system

4) umount file system

6) Increase file system size using extendfs

7) mount /dev/vg??/lv?? /fs_dir

MC/ServiceGuard controlled file systems are not listed in
/etc/fstab, you need to use LV full name when mount the file system.

8) Start application

9) Turn on package monitoring flag

LVM configuration changes during exclusive activation

Suppose I want to extend the mirroring (2-way to 3-way) or I want to create a new logical volume, how will the other nodes(s) know about the changes? Do I have to bring down the cluster? Do I need to import the VG again?

You only need to vgimport again if you add or remove a LV, or add or remove a PV. The reason for this is that the device files on the other nodes need to be updated. If you only change the traits of the LV (mirroring, bad block relocation, size, etc), you don't need to vgimport. In either case, you don't need to bring the cluster down.

For example:

VG01 is activated (exclusive) on node1. Node2 already imported VG01. On node1, we add a new LV (lvol5) to the VG01: “lvcreate -L 100 –n lvol5 /dev/vg01”. We do this while the cluster is up and while the VG01 is activated exclusively on node1. This will create new LV device files: /dev/vg01/lvol5 and /dev/vg01/rlvol5, on node1.

Node2 has no idea that lvol5 was created. You need to tell node2 about this change:

```
On node1: vgexport -p -m vg01.map vg01
On node 2: copy vg01.map from node1 to /tmp/vg01.map
On node 2: vgexport /dev/vg01
On node2: vgimport -m /tmp/vg01.map /dev/dsk/c?t?d?
          Note: include all devices that belongs to vg01, including Pvlinks.
```

You can do this while the VG01 is still exclusively activated on node1. It was not necessary to activate the volume group on node2.

Disclaimer

Motorola does not represent that any information contained in this document nor any associated presentations or discussions with Motorola employees is considered official documentation for the products referenced. This document is purely for information sharing within the industry and Motorola does not warrant any aspect of the content or information provided.