

The Telecom System View

Volume 1.5 Issue 14

Linux Clustering Software for Telecom

By Andy Cress

High Availability, or low Mean Time Between Service Outage (MTBSO) is absolutely essential in the telecom market. One extremely important method of achieving this availability is by the use of clustering at the system level.

Abstract

Products intended for the telecom sector must provide high availability features that will protect against hardware and software failures. This industry has set the hallmark for availability and the effectiveness of methods used to achieve it. Existing products under non-Linux OSs attempt to meet this demand. However, these proprietary products are counter to the open architecture now sought by most companies. Thus, this paper discusses several Linux-based clustering products that are appropriate for Telecom.

You are invited to comment or participate in the publication. Your added insight will be incorporated into subsequent releases of this publication. Please contact Andy Cress at Andrew.R.Cress@intel.com.

Contents

- 1.0 Introduction
- 2.0 Cluster Architecture & Requirements
- 3.0 Linux HA Clustering Products
- 4.0 Glossary
- 5.0 Changes to Subject Material

1.0 Introduction

Telecom Applications (Billing, Provisioning, IVR, Directory Services, HLR, etc.)			
Telecom Services and Protocols (SS7, H323, SIP, MGCP, etc.)			
Clustering Software (Fail-over, Load-Balancing, Data replication, etc.)			
Operating System (e.g. Linux)			
System Hardware (systems or blades)			
NICs	Disks	CPUs	Memory

The Linux operating system has grown in function and in market share recently, and more and more companies are interested in using Linux in a Telecom or enterprise environment where high availability is a requirement (at least 99.999%). One of the key components to high availability is clustering software or middleware that keeps the critical services up despite any software or hardware failures. There are quite a few clustering software products available under Linux, and this paper strives to evaluate which of these are best suited for use in a Telecom environment.

Clustering is a broad term used to encompass fail-over (for hardware and software faults), load balancing, distributed messaging, cluster-aware applications, etc. Most of the products evaluated for this paper were primarily focused on HA fail-over, while a broader look at clustering requirements is outlined in section 2.

Often Telecommunications Equipment Manufacturers (TEMs) have their own proprietary solution for high availability clustering which is integrated with their telecom services software. Some TEMs have begun to port their software to Linux. This paper focuses on standard off-the-shelf clustering products.

2.0 Cluster Architecture and Requirements

2.1 A Linux Clustering Overview

Linux Clustering is a broad term that encompasses various type of clustering products that have quite different purposes. There are several functional categories of clustering, depending on the purpose of the environment and the problem being solved.

Purpose of Clustering	High Performance Computing	Scalability	High Availability	Server Consolidation
Goal	Maximize floating point computation performance	Maximize data I/O performance	Maximize service availability	Maximize ease of management
Description	Lots of systems working together on a single compute-based problem. Measured by the number of FLOPs (floating-point operations) per second it can do.	Lots of systems working on similar tasks, distributed in a defined fashion based on system load characteristics, adding more systems as capacity increases. Network-oriented , or Data-oriented .	Redundancy and failover for fault tolerance of services. Two types of services: Stateless , or Stateful	Central management of resources dedicated to disparate tasks. Treats the cluster as a single manageable unit.
Attributes	- Loosely coupled nodes	- Loosely coupled nodes - Distributor component	- Tightly coupled nodes - Heartbeat traffic between nodes and between services	- Tightly coupled nodes - Have clustered filesystem
Synonyms	FLOP farm, Parallel computing	Load Balancing, Web farm, Parallel database	Failover, Fault Tolerance	Single System Image (SSI),
Examples	Beowulf, MOSIX, OSCAR, Rocks, Ganglia, LAM-MPI, ...	LVS, TurboLinux, LSDB, commercial cluster DB products, ...	Commercial HA clustering products, ...	OpenSSI Clusters, OpenGFS, Lustre, Oracle CFS, ...

Many of the clustering products available fit into more than one of the above categories. For instance, some products include both failover and load-balancing components. Also, some scalable clustered databases also provide both scalability and HA failover functions. Many SSI products that would fit into the Server Consolidation category also provide HA failover functions.

For the Telecom environment, these categories can be prioritized in terms of needs and relevance as follows (most to least):

1. High Availability - High Availability is top priority because it is needed for call control and protocols, and since no open source solution meets all the HA requirements
2. Scalability
3. Server Consolidation
4. High Performance Computing

There are several hardware choices that affect the choice of clustering software. The goal of the clustering software should be to handle as wide a variety of these hardware choices as possible.

- Shared Storage or Non-shared Storage devices
Shared RAID storage is often needed for data-oriented HA applications, so it should be supported by the clustering software,

but other applications do not use shared storage, so non-shared or separate storage devices should also be supported (see 'Shared Nothing'). Some servers are even diskless, with EEPROM or network boot.

- **Blades or standard Systems**
Compact-PCI, ATCA, or similar chassis provide separate blades for CPU and IO processing, so IO ports can be shared, and heavy IO-bound functions can provide better service with blades under cPCI. Other functions do not require this level of IO, and gain more processing power for less cost by using standard PCI rack systems. Many clustering software implementations run on both bladed and standard system architectures.
- **Redundant Network ports**
Features like Link Aggregation or Multipathing can be applied at the driver level for certain network adapters, and this provides additional fault resiliency that is complimentary to, but transparent to, the clustering middleware. The clustering middleware needs to be able to fail-over network ports regardless of the Link Aggregation/Multipathing capability in the network driver.

High Performance Computing

Products in this category focus on maximizing compute performance for floating-point operations. It is primarily used by academic and scientific communities. This is well-suited to Commercial-Off-The-Shelf server products, and serves the purpose of large compute-based floating-point problems.

Scalability

Products in this category focus on maximizing data I/O performance with heavy load. The cluster needs to be able to distribute the application load across N systems with near-linear scalability. This is usually accomplished with some form of load balancing. This increases the performance of the cluster as a whole. A key feature of this category is a distributor module that receives requests and sends them to the appropriate cluster member. There are two sub-categories:

- **Network-oriented scalability**, with network throughput most important.
These usually do not need to save state information, but usually do heartbeat between the distributor and nodes or applications as an HA by-product of the scalability goal.
- **Data-oriented scalability**, with data transactions being most important. Some advanced products may split a single unit of work into pieces. Clustered databases and Parallel databases would fit in this category, although they may also have additional adjunct functions in other categories. Some databases accomplish recovery for failed nodes with server logic, while others add logic to the client (requestor) for intelligent retries.

The clustering software can have an impact on CPU and network bandwidth utilization with its own communication, so this must be contained in order to maintain the overall performance of the services, especially as the size of the cluster grows.

High Availability

Products in this category are focused on maximizing service availability. There are two sub-categories:

- **HA Stateless**, with no saved state information, such as most web services. Some stateless HA products have the service already running, while others start a new instance of the service as part of a failover process. Some stateless clustering products may not save state at the server, but depend on the client to preserve his current state.
- **HA Stateful**, with state information to be saved. This allows sessions to be maintained across a failover. Stateful clustering products save the state information on the server side, within the cluster.

It is critical for these products to be configured in an environment where there is no single point of failure, not even power. The key measurement of availability is the amount of downtime over a period of time. More advanced products can achieve 99.999% availability, which is less than 5 minutes of service downtime per year. Fast failover is a key component of high availability, so that the failover does not consume the allotted yearly downtime. Note that failure detection and failure recovery are two separate and important parts of the failover latency.

Server Consolidation

Products in this category are focused on maximizing the ease of management of the cluster. A cluster filesystem and other underlying administrative functions allow the administrator to manage the cluster as if it were a single system without reference to specific cluster administration steps. For remote administration, every possible administration function needs to be performed remotely (or automatically) from a network operations center. Errors should be reported to the network operations center in a standard format (such as SNMP). Note that these functions do often overlap with other clustering products with goals in the other categories.

Related Links for some open-source examples in each category:

OSDL CGL Clustering Links: <http://developer.osdl.org/cherry/cluster-links/>

Service Availability Forum, interfaces: <http://saforum.org>

High Performance Computing: Beowulf <http://www.beowulf.org/>, MOSIX <http://www.mosix.org>, OSCAR <http://oscar.sourceforge.net>, Ganglia <http://sourceforge.net/projects/ganglia>, NPACI Rocks <http://www.rocksclusters.org/Rocks/>, PVM http://www.csm.ornl.gov/pvm/pvm_home.html, Portland CDK <http://www.pgroup.com/products/cdkindex.htm>, LAM-MPI Parallel Computing <http://www.lam-mpi.org>, see also LinuxHPC.org projects <http://www.linuxhpc.org>

High Availability (stateful): Most of these are commercial packages, see section 3.0 below.

High Availability (stateless): Linux HA Heartbeat project <http://linux-ha.org/heartbeat/>
Open HA Cluster project <http://sourceforge.net/projects/open-ha-cluster/>
SGI FailSafe <http://oss.sgi.com/projects/failsafe/>

Scalability (network): Linux Virtual Server <http://www.linuxvirtualserver.org/>

Scalability (data): Linux Scalable Database project <http://sourceforge.net/projects/lsdproject/>

Server Consolidation: OpenSSI Clusters for Linux* <http://sourceforge.net/projects/ssic-linux/>
OpenGFS <http://sourceforge.net/projects/opengfs>
Lustre <https://projects.clusterfs.com/lustre/>
Oracle CFS <http://oss.oracle.com/projects/ocfs/>

2.2 Architectural Goals for Telecom Cluster Software

The architectural goals for cluster software in the telecom environment are different than compute-bound clusters such as Beowulf and Mosix, and are also different than web-based server clustering, due to the nature of the applications and services to be clustered.

There are a variety of types of telecom applications that need to use clustering. Here are some of the key application types:

1. Stateful Core services & applications: call control, protocols (MGCP, SS7, H.323, SIGTRAN, SIP, etc.)
These services and applications require a mix of CPU, memory, and IO resources, but very little disk storage. This is the most important type for High Availability clusters.
2. Storage/Data-driven applications, such as: Billing, Provisioning, HLR/VLR, etc.
These applications require large memory configurations, due to the fact that the databases need to be kept in memory for performance reasons.
3. Value-added (Edge) services, such as Unified Messaging, IVR, Voice Recognition, etc.
These applications have less stringent availability requirements because they are not part of the core network, but have unique CPU, memory and IO requirements.
4. Stateless data network applications (e.g. Network Management, Mail, DNS, Web services)
These applications are not part of the core telecom network, but may appear as infrastructure for the telecom service providers. They have short transactions that do not require maintaining states across the cluster, and tolerate large failover latencies. Some clustering packages only fit this type, which is not usually sufficient for the telecom environment.

The most stringent of the above types would be the stateful core services, with the storage type also being important to ensure that those different requirements are met. It may be that different clustering software would be used for each type of clustering above, but a cluster software choice that met each of the requirements would be ideal.

Across these choices, clustering software vendors share a common set of requirements established by the telecom market. Given the environment, clustering software vendors should pursue the following architectural goals:

- Remote Administration
Every possible administration function needs to be performed remotely from a network operations center. Errors must be reported to the network operations center in a common format.
- Service failure detection and recovery latency requirements for clustered failover
The clustering software's baseline failover needs to be tunable down to fine grain latency of less than one second, with latencies of 500 milliseconds or lower are preferred. Note that failure detection, isolation, and service recovery are separate and important parts of the failover latency. Saving additional state information would not be considered part of this baseline latency. Telco core services need the shortest latency, but there are some transaction-based functions in the edge services where accuracy is more important than latency time.

- Support varied types of network applications with varied availability requirements
Each application will need tunable resource parameters for availability management. Some applications have a very low tolerance for availability changes, while others will prefer to allocate resources differently. A tiered or tunable approach to providing cluster services is desirable to meet these varied requirements.
- Scalability
The cluster needs to be able to distribute the application load across N systems with near-linear scalability. This is usually accomplished with some form of load balancing. This increases the performance of the cluster as a whole.
- Performance
The clustering software can have an impact on CPU and network bandwidth utilization with its heartbeat and membership algorithms, so this must be optimized in order to maintain the overall performance of the telecom services, especially as the size of the cluster grows. The cluster management traffic, such as heartbeats, should have the capability to be designated on a separate network, but the CPU and memory overhead of inefficient heartbeats can impact the telecom applications & services.
- Ease of integration of clustered applications & services
Service providers want their choice of network applications; therefore, application vendors want ease of integration between clustering software implementations. Standards such as JDK, CORBA, POSIX, LSB, SAForum HPI & AIS, etc. should be employed where appropriate to minimize the effort required to move applications from one vendor's clustering middleware to another's.
 - Stateless applications & services
Integration of stateless applications should be as transparent as possible. The clustering vendor may provide packaged kits for common applications.
 - Stateful applications & services
More integration is required with stateful applications, for check-pointing, etc. These are often custom-developed applications, so the clustering package needs well-defined cluster APIs.

We can categorize these goals into three cluster-awareness categories:

1. Clustering middleware that is aware of virtual services managed by it.
This tier includes features like basic service fail-over & restart when a node fails, and usually involves an active/standby cluster configuration. Advanced products also would include some scalability features.
2. Stateful services that are modified to be cluster-aware.
This tier adds features like application checkpointing, application messaging, and some advanced products also have application shared-memory. This usually involves some code changes to the applications & services to add these features.
3. Single System Image
This tier adds features such as a cluster filesystem that makes the filesystem transparent over the cluster, and other advanced services that make the cluster transparent to most of the applications.

Here is a sample cluster software architecture diagram, showing the various software components that may be part of the clustering software solution.

Applications						Cluster Event Notification Service
Cluster Membership (heartbeat, failover)	Cluster Storage Service	Fault Management Service	Cluster Load Balancing Service (scalability)	Application Management	Message Service	
UDP, TCP/IP, etc. (protocols)			Virtual Interface Architecture (VIA)			
LAN/WAN (media)			Server Area Network (SAN), cPCI processors, InfiniBand			

A **cluster membership** service includes the functions to recognize and manage the nodes' membership in the cluster. Part of this is normally implemented with some kind of heartbeat protocol to recognize when a node has failed and must be removed from the active cluster.

A **cluster storage service** includes the replication and retrieval of cluster management data as well as application data. An integrated cluster database is often used for the cluster management data. Application data may take the form of checkpointed state information, or more general data records or files that should be distributed throughout the cluster.

A **fault management service** includes functions to recognize hardware and software faults within the node and often includes fault prediction mechanisms. The fault severity is determined and an appropriate event is generated.

A **cluster load-balancing service (scalability)** includes functions to distribute the service load across the cluster to achieve better aggregate performance with increased load (scalability). Some type of distributor module is needed to accomplish this.

An **application management** interface includes functions and interfaces to allow applications to access the scalability and fail-over capabilities of the cluster. Portability, flexibility, and ease of integration with various applications are the key goals.

A **message service** allows fast and reliable message passing between nodes in the cluster. The messages have primarily cluster management content. The message protocol may be linked with the heartbeat protocol, or it may be different.

A **cluster event notification service** ensures that a management station, which is usually remote from the cluster, can receive meaningful notification for cluster membership and cluster fault events. Enough information must be provided so that a support service call can be generated. The notification method can vary depending on what is recognized by the enterprise management station.

2.3 High Availability Clustering Requirements

While the other types of clustering may have readily available open-source or commercial products from a variety of sources, the most difficult type of clustering to implement for the Telecom environment is High Availability, of which HA stateful failover is the most complex sub-type. There are several points of failure in a telecom system, and the clustering software must take each into account in order to increase the availability of the clustered services. Hardware failures, such as NICs, disks, or entire nodes must be monitored, and extensible facilities must be provided to detect software process failures in both services and applications. Also, there are differing requirements when dealing with CompactPCI systems and regular rack-mount PCI systems that affect clustering. CompactPCI systems utilize intelligent adapters and fail-over can often occur within a chassis, whereas rack-mount PCI systems may need to fail-over between chassis more often. So, fail-over between chassis is a requirement for rack-mount PCI systems.

The OSDL Carrier Grade Linux forum has defined some requirements for Linux clustering also. The intention of these requirements is to provide common building blocks in open source for vendors to assemble and use in a clustering solution on Linux. See http://www.osdl.org/docs/carrier_grade_linux_requirements_definition_version_20.pdf

In order to provide uninterrupted service in a telecom environment, HA fail-over software must provide certain required features. Below are the features that were used for this evaluation of existing HA fail-over products on Linux. Fail-over features do not include all of the functionality needed in a telecom environment, but these are some of the key differentiators. The fail-over features listed fall into one of the following feature categories: Membership, Application Management, Storage, Fault Management, Scalability, Messaging, and General. The requirements below are primarily from a high availability clustering perspective, with some features from scalability and SSI clusters also included in the Storage and Scalability categories to gauge the product's versatility.

Feature Category / Feature	Description
1) <i>Membership</i> IP Failover Latency	Almost all clustering software packages can do a failover of a TCP/IP address, so this is a common baseline. The telecom environment requires that services fail-over rapidly, without discernable interruptions. A lowest-common metric for rapid failover is the fail-over of a virtual IP address. Stateful application failover latency will be longer than this, but its tunability will depend on the same factors measured here. The IP fail-over should not take the IP address out of service for more than 1 second if one NIC or an entire node/chassis fails. This means that failure detection via heartbeats or other means must have tunable variables to allow minimum detection time, and any takeover/recovery timing values, if present, must be minimized. This tuning must reduce the total fail-over latency, yet avoid false fail-overs.
2) <i>Membership</i> Cascading failover	If one node fails, and a standby node takes over for it, then that new node fails, there needs to be the ability to 'cascade' the fail-over so that another node can take over at this point. Ensure a consistent view of a cluster from every node, so that no one node has sole management/view responsibility

	<p>Automatic cluster membership when a node is added to the cluster network with the clustering software installed</p> <p>Automatic cluster reconfiguration on node shutdown and heartbeat failures</p>
3) <i>Membership</i> N:M cluster heirarchy	<p>With more than 2 nodes, it becomes necessary to designate which nodes can provide fail-over for the other nodes. N means the number of nodes active in the cluster, and M means the number of nodes that are standby for fail-over, such that they can provide fail-over for any or a set of the clustered nodes (N:M). Some software packages can have only one standby (M) for each active set of nodes (N), which is called N:1. If all nodes are symmetrical, so that they are both active and can serve as spares, the heirarchy is labelled N:0. In order to support cascading failover, at least N:M is required.</p>
4) <i>Membership</i> Heartbeat method	<p>Heartbeat media can be via LAN, WAN (e.g. Serial) and/or Disk. One or more media can be used, but specialized types of media available only from one vendor should be avoided.</p> <p>The heartbeat should have tunable timing.</p> <p>The protocol used should give speed and reliability.</p> <p>Efficient heartbeat over a large number of nodes (at least 8).</p> <p>Do fault detection, location and isolation (fail-over) in less than 1 second.</p> <p>Allow membership in multiple logical clusters on the same physical network</p> <p>Multipathing support to allow transparent failover for network interfaces</p> <p>Some method to bypass standard network layers to improve efficiency of the cluster protocol.</p> <p>Standard network layers would be used to support decentralized distributed applications or other functions that don't have stringent recovery time constraints.</p>
5) <i>Membership</i> Membership method	<p>Each cluster software package must have a method to determine whether it has the right to become a member of the cluster or to perform cluster management functions. Many different implementations have been used with varying degrees of performance, scalability, and reliability. Some applications are better suited for one membership method.</p>
6) <i>Application Management</i> Stateful application failover (checkpointing)	<p>Stateful applications require state and/or session information to be preserved across a failover. Protocol services are a good example. Usually a clustering API is required so that the service or application can checkpoint its state information via cluster software. The stateful application can then fail over without losing its session and other state information. A example stateful Telco application in open-source is OpenH323/OpenGateway.</p>
7) <i>Application Management</i> Stateless application failover	<p>These applications do not have a requirement of state information to be preserved across a failover. Web servers are a good example. The clustering software should include either a generic redirector or standard service kits to easily configure the cluster for the most common applications and services.</p>
8) <i>Application Management</i> Application health	<p>Application heartbeat or other health monitoring mechanism. Since some software faults may cause an application to be out of service, while the node itself appears healthy.</p>
9) <i>Application Management</i> Features	<ul style="list-style-type: none"> • Support cold, warm, and hot process replication • Support a mechanism to define application dependencies • Generate alerts to an event notification service • Application failure recovery: <ul style="list-style-type: none"> ○ detect a client program's failure ○ automatically restart a client program at failure recovery ○ bring the client program back to the state just before a failure occurred • Support an enterprise management interface for application and service installation, configuration, monitoring, and updating. • Support policy-based application management (rules) <p>Some products monitor application health through instrumenting the application/service to an API, and some use hooks in the OS or libraries to make the management and monitoring transparent to the application. Either approach would meet this requirement.</p>
10) <i>Storage</i>	<p>Many core telecomm functions either would either not use databases, or use in-memory</p>

Shared nothing	databases that would not use shared storage, so the clustering software must support configurations in which there is no common shared storage. This is the default storage configuration for Telco applications.
11) <i>Storage</i> Shared SCSI storage	For OSS application functions, telecom systems will require large amounts of data. Using more than one SCSI host connected to the same external RAID enclosure provides the needed redundancy without replicating large amounts of data. The clustering software needs to support this configuration through appropriate locking and storage health indicators, as well as multihost support in the kernel.
12) <i>Storage</i> Shared FC storage	For OSS application functions, telecom systems will require large amounts of data. Using more than one Fiber Channel host connected to the same RAID enclosure provides the needed redundancy without replicating large amounts of data. The clustering software needs to support this configuration through appropriate locking and storage health indicators.
13) <i>Storage</i> Diskless nodes	Some Telco servers are diskless, with EEPROM or network boot. The cluster software should be flexible enough to work without local disk storage.
14) <i>Storage</i> Lock Manager	Locking resources between nodes that are in the cluster is required, to allow multiple nodes to compete for and access cluster resources.
15) <i>Storage</i> Cluster Database	A replicated or cluster-wide database is required to maintain internal membership data, to store checkpointed state information, and optionally to store more application data. If this database is maintained in memory it provides a performance advantage.
16) <i>Storage</i> Network Replication	If the node does not have direct access to shared storage, network media can be used for remote storage and redundancy. Projects such as DRBD and NBD allow Linux to redirect or replicate block storage requests across the network. Facilities such as Network-Attached Storage (NAS) devices can also provide this functionality.
17) <i>Storage</i> Cluster File System	A clustered file system provides both journailling and distributed features. A clustered file system should be transparent, so that it appears just like a single-node file system. There are commercial and open-source alternatives. Some open-source projects are: OpenGFS, Oracle CFS, Lustre, and OpenSSI.
18) <i>Fault Management</i> Remote Administration	Need to support a Command-Line Interface (via network and/or serial) for automation of the cluster administration functions for a large number of nodes from a remote management station. The ability to support multiple platforms and OSs with the same clustering product would be an extra benefit.
19) <i>Fault Management</i> Event Notification	Policy-based fault management – rules that govern event notification Node failures or shutdowns – should result in a notification (related to membership) Cluster configuration changed – should result in a notification (related to membership) Process failures or shutdowns – should result in a notification Exhaustion of system resources – should result in a notification Extensible to additional unique cluster events that the customer desires Support accepted standard management frameworks and instrumentation - SNMP, CIM, TMN, HTTP, etc. (SNMP at a minimum)
20) <i>Fault Management</i> Detect hardware faults	Cluster software vendors may support several types of hardware platforms. IPMI and the newer SAForum Hardware Platform Interface from www.saforum.org yield platform-independent APIs for cluster fault management services to utilize. Detect all link failures, even the redundant links, and let the application decide whether to fail-over or not.
21) <i>Scalability</i> Number of nodes	Often HA clusters are formed in service groups of 2 to 8 nodes. HA clusters of 3 or more nodes are needed so that planned service times can still provide fail-over redundancy. Support for as many as 64 nodes would also be desirable for scalability clusters. Sometimes HA clusters and scalability clusters are combined in the same service groups, especially if the cluster software product supports both.
22) <i>Scalability</i> Resource Efficiency	Minimize the amount of CPU, memory, and network bandwidth that the cluster service requires for management of the cluster. If the cluster management resources are minimized, it

	allows more application load capacity on each node, and thus better scalability.
23) <i>Scalability</i> Load Balancing	<p>Many clustering vendors use the open-source Linux Virtual Server (LVS) code in order to support load balancing. LVS can be used along with the fail-over products, or the clustering vendor may include built-in load balancing support as part of their product.</p> <p>Monitor each node in the cluster using server statistics</p> <p>Statistics should be gathered from a common interface for individual nodes. This could be provided by a protocol like HTTP, SNMP, DMI, or CIM.</p> <p>Provide server node CPU, memory, disk I/O, network I/O, process stacks/queues, and other OS resource metrics to applications</p> <p>Extensible for user-defined resource metrics</p> <p>Allow applications to select from a set of balancing algorithms (rules)</p> <p>Support gradual quiescence and shutdown of a server/node</p> <p>Generate threshold alerts to an event notification service</p> <p>Provide overload and bandwidth protection</p> <p>Ability to designate servers in a cluster as backup/standby for faulted servers in the cluster</p> <p>Support multiple platforms and OSs in a cluster (desired, but optional)</p> <p>Web-based cluster monitoring and global administration, supporting replication of rules to multiple nodes</p>
24) <i>Messaging</i> Messaging service	<p>Peer-to-peer messages</p> <p>Broadcast/Multicast messages to cluster members</p> <p>Publish and subscribe functions</p> <p>Synchronous and asynchronous messages (acknowledged or not acknowledged)</p> <p>Priority messages, so that a heartbeat message does not get queued up behind an informational message, for instance.</p> <p>Guaranteed message delivery</p>
25) <i>General</i> Rolling upgrades	The clustering software should provide for taking a system out of the cluster, performing planned upgrades or maintenance on it and bringing it back into the cluster, so that the next system can then be upgraded. When the cluster software itself is upgraded, the cluster must support back-compatibility with the previous cluster software version.
26) <i>General</i> Split-Brain condition	The cluster software design should either <u>avoid</u> and/or <u>handle</u> the condition in which two nodes (or groups of nodes) believe that they are each supposed to take over a cluster-wide service. Handling split-brain may sometimes result in a STONITH algorithm, suicide, or both. Avoiding split-brain has to do with the membership design methodology.
27) <i>General</i> Multipathing support	When there are redundant network and storage paths to the same sets of devices, sometimes the device drivers handle multipathing and switchover on failure, but if not, the clustering software should handle switching across redundant network/storage paths to a given destination device.

3.0 Linux HA Clustering Products

3.1 Tier 1 Linux HA Clustering Products

In my experience, only a few of the clustering products available commercially, and none of the currently available open-source products in the Linux community meet the HA requirements as specified in section 2.3. These requirements are useful for product evaluation and also for future development, both commercially and in open-source projects. Some requirements, such as rapid failover, are difficult to assess without actual testing or demonstration. Those products that do meet the requirements specified in section 2.3 above can truly be labelled 'Tier 1 Linux HA Clustering Products'. Each clustering product has its specialized features and strengths.

One of the key metrics tested for each product is rapid failover (including detection + recovery), and this was measured in the simple case by using a node outside the cluster to send ICMP packets every 10 milliseconds to the clustered IP address, and turning off the active node for that IP address, showing how long the IP resource was out of service when a failover occurred. Note that the CORBA metric is measured with application echo messages rather than ICMP echo packets. As stated in section 2.3, a requirement of IP failover recovery within 1 second was set for the telecom environment. Also, this failover recovery latency should not be tuned so low that it will cause false failovers. Due to the fact that very few of the clustering products achieved this metric, products that have an IP latency of less than 5 seconds can be included in the Tier 1 category, since incremental improvements may bring these products better latency.

Below is a chart comparing some of the important features of the 'tier 1' clustering products for Linux and non-Linux OSs. The features from section 2 included in this chart are key differentiators among HA clustering products in the Telco environment. Either a clustering customer can fill in the information in the columns, or a clustering product vendor could respond with their view of how these requirement are met for comparison.

KEY PRODUCT FEATURES							
IP failover latency							
Stateful Application failover API							
Stateless Application failover kits							
Event Notification (SNMP traps)							
Remote administration							
Size of cluster (# nodes)							
Cascading failover							
N:M hierarchy							
Shared Nothing support							
Shared Storage support							
Rolling upgrades							
Load Balancing support							
Resource usage: memory							
Resource:usage network							

4.0 Glossary

Glossary

This table defines terms used in this paper that are not defined elsewhere. Many other terms are defined in the following CGL Definitions table.

Checkpointing	A function that allows applications to synchronize state information and user data at intervals or checkpoints, so that a stateful failover can be performed. This can be done via a cluster API that the applications use or via agents/redirectors that capture application data in some transparent way. See also [Data Checkpointing] below.
Loosely-coupled nodes	Fewer logical connections and dependencies between nodes. If each node operates independently with very little synchronization or dependency between nodes, this label is used. This obviously implies a 'shared nothing' storage hardware configuration.
MTBSO	Mean Time Between Service Outage. The interval of time in which the cluster can provide continuous service. Failures can occur, as long as redundancy allows the service to continue uninterrupted.
Shared Nothing	The lack of shared storage or shared memory in the physical hardware configuration of the cluster. Each node contains all of the hardware needed to participate as a member of the cluster.
Split-Brain condition	The condition in which two nodes (or groups of nodes) believe that they are each supposed to take over a cluster-wide service.
STONITH	Shoot-The-Other-Node-In-The-Head. A method to handle the confusion of Split-Brain conditions, so that an active node forcefully takes another node down.
Tightly-coupled nodes	More logical connections and dependencies between nodes. If each node needs to be synchronized (via checkpointing) and/or participate in locking between nodes, this label is used. This could be either a 'shared nothing' or a shared storage hardware configuration.

CGL Definitions of terms for High Availability and Clustering

Revised 11/20/2003

http://www.osdl.org/docs/cgl_definition_of_terms_for_high_availability_and_clustering.pdf

Contributors

Chacron, Eric (Alcatel)
 Chen, Terence (Intel)
 Cherry, John (OSDL)
 Dake, Steve (MontaVista)
 Fleischer, Julie (Intel)
 Haverkamp, Mark (OSDL)
 Kukkonen, Mika (OSDL)
 Marclay, Bernard (HP)
 Marowsky-Brée, Lars (SuSE)
 Minyard, Corey (MontaVista)
 Williams, Michael (Ericsson)

Introduction

1/22/2004

Page 11 of 16 Pages

Copyright (c) 2004 by Intel Corporation. This document may be distributed in whole or in part without permission, as long as the author and copyright information is included in said distribution. Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder. Linux is a Registered Trademark of Linus Torvalds. Other company, product, or service names are the property of their respective owners.

This document contains definitions of general terms used in describing high availability (HA) and clustering in the context of carrier grade systems. Definitions of terms commonly used in a particular design and implementation of a HA/clustering solution are not in the scope of this document. Angle brackets [] are used in definitions to denote other terms defined within this document.

Definition of Terms

Application	A set of [processes], running on a computer [system], that provide a service to the [users] of this [system]. Application is usually referred to the non operating system portion of the software in a [system].
Availability	Availability is the amount of time that a system [service] is provided in relation to the amount of time the system [service] is not provided. System [service] downtime could be the result of system failures (unscheduled downtime) or for things like upgrades, system relocation, or backups (scheduled downtime). A system [service] is provided if the [service] is functioning at an acceptable level of performance or scalability. Availability is commonly expressed as a percentage (see [five-nines] or [six-nines]). Percent Availability = (time service is provided / total time) X 100
Cluster	Two or more computer [nodes] in a [system] used as a single computing entity to provide a [service] or run an [application] for the purpose of [high availability], [scalability], and distribution of tasks.
Communication	The exchange of information between [processes]. These [processes] can be running on the same [node] (intra-node) or on different [nodes] (inter-nodes). The information includes [events] and [messages].
Data	Numerical or other information represented in a form suitable for processing by a [process].
Data Checkpointing	The mechanism by which [application] state is transmitted from an active [service unit] to one or more standby [service units].
Event	A [communication] with or without data which notifies a set of zero or more [processes] that something took place. It travels within a [node] and/or can be between [nodes]. Events use a publish/subscribe interface. A [subscriber] informs the event mechanism that it wishes to receive a certain event. A [publisher] posts an event to the event mechanism to be delivered to all [subscribers] of that event. This way the [publisher] and [subscriber] are decoupled, they do not have to directly know about each other, just about the event. Events may be asynchronous or synchronous. A [publisher] posting a synchronous event will block or be informed when all [subscribers] have received the event. The [publisher] of an asynchronous event will not block waiting for delivery or be informed when the event is delivered to any [process].
Event Service	A publish/subscribe event service that manages [events]. [Events] may be grouped into named channels and handle attributes such as priority, ordering, retention times, and persistence.
Fail-back	The process to migrate back to a [node] after it has been [repaired]. It can be controlled or automatic.
Fail-over	Ability to automatically switching to a [redundant] [node], [system], or [network] upon the [failure] or abnormal termination of the currently-active [node], [system], or

	[network]. Failover happens without human intervention.
Failure Detection	A failure is ultimately caused by an unmasked [fault] in the [system] and thus showing through to the [user]. Failure detection is the process, usually from external view, to detect a [failure] of the [service] the [system] is providing.
Failure	The inability of a [system] or [system] component to perform a required function within specified limits. A failure may be produced when a [fault] is encountered. Examples of failures include invalid data being provided, slow response time, and the inability for a [service] to take a request. Causes of failure can be hardware, firmware, software, network, or any other reason that interrupts the [service].
Fault	An error in a computer [system] or the [service] it provides. A fault may be masked and not impact the [application] or the [service] it provides. A fault can also be classified as transient or permanent. A fault is often associated with a [system] defect in the software or hardware. A fault can also be caused by something happens external to the [system].
Fault Confinement	Equivalent to [fault isolation].
Fault Detection	Ability to detect an abnormal condition (device failure, temperature error, etc.) in the [system].
Fault Diagnosis	The localization of a [fault] to its repair unit.
Fault Isolation	Ability to protect the rest of the [system] from the effects of a [fault].
Fault Prediction	Detecting or forecasting [faults].
Fault Tolerance	Ability for a [system] to mask a set of [failures] from occurring and not impact the [service] it provides.
Five-nines	Five-nines is measured as 99.999% [service] [availability]. It is equivalent to 5 minutes a year of total planned and unplanned downtime of the [service] provided by the [system].
Group Multicast	The sending of a single [message] to a set of destination [processes].
Hand-over	Equivalent to [switch-over]
High Availability	The state of a [system] having a very high ratio of [application] uptime compared to [application] downtime. Highly available systems are typically rated in terms of number of nines such as [five-nines] or [six-nines].
Lock Service	The lock [service] is a distributed lock [service], suitable for use in a [cluster], where [processes] in different [nodes] might compete with each other for access to a shared resource. A lock [service] may provide the following capabilities: exclusive and shared access, synchronous and asynchronous calls, lock timeout, trylock, deadlock detection, orphan locks, and notification of waiters.
Message	A [communication] with [data]. A message may contain attributes of the [communication] such as source, destination, time stamps, and authorization information, etc; it may also contain [application] specific information.
MTTF	Mean Time To [Failure]. The interval in time which the [system] can provide [service] without [failure].
MTTR	Mean Time To [Repair]. The interval in time it takes to resume [service] after a [failure] has been experienced.
Node	A single computer unit, in a [network], that runs with one instance of a, real or virtual, operating system.
Network	A connection of [nodes] which facilitates [communication] among them. Usually, the

	connected nodes in a network use a well defined [network protocol] to communicate with each other.
Network Protocols	Rules determining the format and transmission of data. Examples of network protocols include TCP/IP, UDP, etc.
Node membership	The mechanism by which [node] registration, un-registration, and [failure detection] is managed. A [node] is deemed to be a member if it has registered with the [cluster] successfully. A [node] is deemed to be a non-member if it has not registered with the cluster. A detected [failure] may make the [node] a non-member, depending on node membership policy. A [node] can either gracefully un-register to depart from the membership or this node fails and be evicted from the membership by force. The node membership also handles authorization to join the [cluster] or not.
Performance	The efficiency of a [system] while performing tasks. Performance characteristics include total throughput of an operation and its impact to a [system]. The combination of these characteristics determines the total number of activities that can be accomplished over a given amount of time.
Process	A single instance of a software program running on a single [node].
Process group	A collection of processes registered within [cluster] software.
Process group membership	The mechanism by which a [process] registration, un-registration, and [failure detection] is managed. A [process] is deemed to be a member if it has registered with the [process group] successfully. A [process] is deemed to be a non-member if it has not registered with the process group. A [detected] failure may cause the [process] to become a non-member, depending on process group membership policy. A [process] can either gracefully un-register to depart from the membership or this [process] fails and be evicted from the membership by force. The process group membership also handles authorization to join the membership or not. Process group membership depends upon [node membership] if process group membership is available on multiple [nodes]. Process group membership is used to execute application [failover] policy.
Publisher	A [process] that sends [events].
RAS	[reliability], [availability], and [serviceability]
Recovery	To return a failing component, [node] or [system] to a working state. A failing component can be a hardware or a software component of a [node] or in the [network]. Recovery can also be initiated to work around an occurred [fault]; ultimately restoring the [service].
Redundancy	Duplication of hardware, software, or network components in a [system] to prevent having a [Single Point of Failure].
Reliability	The continuation of [service] in the absence of [failure]. Reliability is commonly measured as the [MTTF] of a [system].
Repair	The process to remove a [fault].
Replication	A component, [node], or [system] which is configured identically to a base component, [node] or [system] for the purpose of [fault tolerance], [performance], or ease of [service].
Scalability	How well a solution to some problem will work when the size of the problem increases. In the CGL context, the scalability is defined as the ability of a [system] to provide the same level of [high availability] performance when the work load of [service] increases. The solution to increase the [system] or [service] scalability can be

	software or hardware oriented.
Service	A set of functions provided by a computer [system]. Examples of Telco services include media gateway, signal, or soft switch types of applications. Some general examples of services include web based or a database transaction types of applications.
Service Unit	A collection of one or more software [processes] that provide [service] to a [user].
Serviceability	The capability for a [system] to be maintained and updated. Often, serviceability is measured by how easy a maintenance task can be performed or how quickly a [system] [fault] can be tracked down and repaired so that the [system] can resume the [service].
Single Point of Failure	Any component or [communication] path within a computer [system] that would result in an interruption of the [service] if it failed.
Six-nines	Six-nines is measured as 99.9999% [service] [availability]. It is equivalent to 30 seconds a year of total planned and unplanned downtime of the [service] provided by the [system].
Subscriber	A [process] that receives [events]. A [subscriber] may subscribe to one or many [events]. A subscriber may join and leave an event subscription at any time without involving the publishers.
Switch-over	Ability to switch to a [redundant] [node], [system], or [network] upon a normal termination of the currently-active [node], [system], or [network]. Switch-over can happen with or without human intervention.
System	A computer system that consists of one computer [node] or many nodes connected via a computer network mechanism
User	An external entity that acquires [service] from a computer [system]. It can be a human being, a external device, or another computer [system].

6.0 Changes to Subject Material

“THE TELECOM SYSTEMVIEW” white papers are intended to be living documents that serve to capture the current state of knowledge. From time to time new information will come to light that will cause subsequent issues including changes from previous issues. The volume number will remain the same for a give subject but the issue number will increment. This section will be included at the end of the document and will indicate the reason for a follow-on issue.

Document	Date	Description
Volume 1.5 Issue 1	Feb-8-2001	Initial Release
Volume 1.5 Issue 2	Feb-12-2001	Changed opening of section 4.0.
Volume 1.5 Issue 3	Mar-5-2001	Changed shared storage requirement text in section 2.0
Volume 1.5 Issue 4	Mar 21-2001	Changes to various products in the evaluation list.
Volume 1.5 Issue 5	Apr-23-2001	Added some metrics to Tier 1 and corresponding information.
Volume 1.5 Issue 6	May-9-2001	Updated some metrics, changed some product naming, also fixed header Issue#.
Volume 1.5 Issue 7	July-13-2001	Added VendorA to Tier 1
Volume 1.5 Issue 8	Aug-8-2001	Added comments about a new vendor
Volume 1.5 Issue 9	Aug-23-2001	Edited for improvements in another new product version.
Volume 1.5 Issue 10	Sep-13-2001	Added Architectural Goals, merged all Reqmts to section 2.3, added other vendor comments.
Volume 1.5 Issue 11	Oct-2-2001	Updated Architectural Goals, added two new vendors to other vendor list.
Volume 1.5 Issue 12	Oct-5-2001	Added info about Link Aggregation, shared fiber, and rolling upgrades
Volume 1.5 Issue 13	Sep-26-2002	Added 2 more vendors, more comments on others.
Volume 1.5 Issue 14	Jan-14-2004	Lots of changes to requirements and vendor sections for a new round of testing. Merged with comments from OSDL CGL group.