HPEC: High Availability by Design

Read About

Info

curtisswrightds.com

Email

ds@curtisswright.com

Introduction

A fault tolerant system must have the ability to continue processing data even when a hardware failure occurs. This is accomplished by building duplicate hardware of all critical components of the system, eliminating single points of failure such as the processors and power supplies. When a component fails, the software must be able to detect the failure and handle the switching of the hardware and re-routing of the data flow. As the next generation of embedded defense systems become even more complex with more computational power, memory, data, and speed, the problem of designing effective fault tolerant systems also becomes more difficult. Fortunately, the High Performance Computing (HPC) world has developed a set of mature, proven methodologies and tools to support High Availability (HA) clusters. By definition, availability refers to a level of service provided by applications, services, or systems. Highly available systems have minimal downtime, whether planned or unplanned.

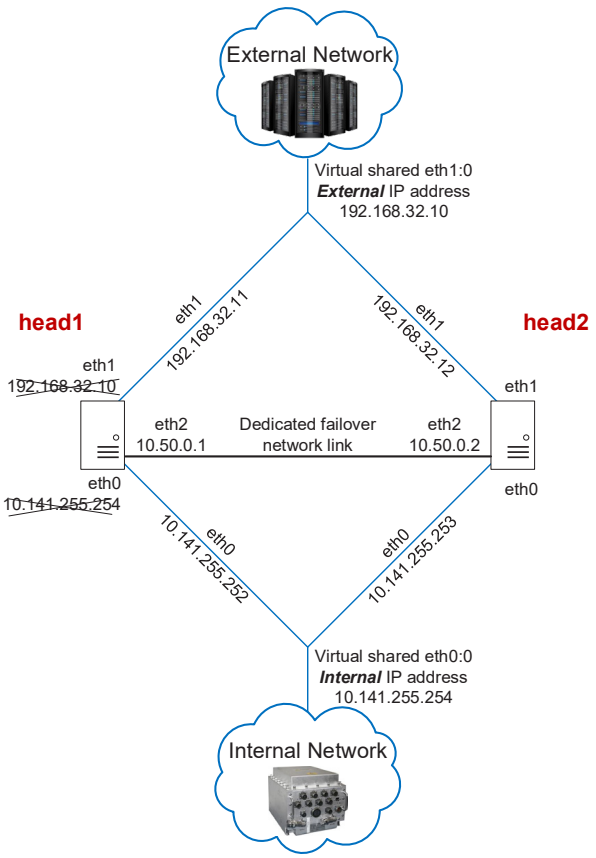

Figure 1: Layout of a Two-Head Failover Network

Dissecting the HPEC Cluster

Let's review the concept of a cluster as it pertains to a High Performance Embedded Computer (HPEC). A cluster normally contains a number of processors interconnected by a high-speed interface such as 10/40 GbE Ethernet or InfiniBand®. In the world of OpenVPX™, this is known as the data plane. In addition to CPUs, the system could also contain GPUs connected via PCI Express® (PCIe). A cluster also has an internal network, known as the control plane in OpenVPX systems, and usually includes one or multiple GbE switches or an InfiniBand fabric.

The Cluster Manager

Because it controls all of the other nodes in the system, the head node is the most important piece of the cluster. The head node is the only node connected to an external network; think of it as the router between the internal and the external networks. The head node is the master of the cluster because it controls all the other devices including the compute nodes, switches, and power distribution units. The head node can be one of the processors in your embedded box, a stand-alone computer, or perhaps even an encrypted data storage device. It provides vital services to the rest of the system including central data storage, user management, DNS and DHCP services. The regular, or compute nodes, are configured automatically by the node provisioning feature of a cluster manager, such as Bright Computing's Cluster Manager. When the system is started, the compute nodes are PXE-booted with software images that are stored on the head node, ensuring a node can always start in a "known state". This also ensures software changes can easily be undone since these changes are only made once in the software image on the head node. Generally, this removes the need to log directly into the compute nodes. More importantly, this node provisioning system eases the administration of the compute nodes making it a trivial task to replace an entire node in the event of a hardware failure. The downside is that the head node is the single point of failure for the entire system.

The "Failover"

The solution to this potential issue is a high availability setup which is a typical configuration in HPC systems. This is achieved by replicating the head node to a second processor or computer, which will be referenced as the passive head node. In the HA setup, the passive node continuously monitors the active head node. When it detects that the active head node has failed, the passive node will assume control and take possession of the resources, services, and network addresses; this is called a "failover". Whenever possible, the same services are run on both the active and passive nodes which results in moving fewer services in the event of a failover. The services running on both nodes include CMDaemon (for node provisioning), DHCP, TFTP, NTP, and DNS. During normal operation, the passive head node is provisioned by the active node. In a typical HA setup only the NFS and user portal are migrated from the active node to the passive, therefore saving time if a failover occurs.

Each head node in an HA setup typically has as, at a minimum, an external and an internal network interface, each configured with an IP address. In addition, an HA setup shares two virtual IP interfaces, one for the external network and one for the internal network, between the two head nodes. However, only one node can host the address and its interface at any given time. During normal operation, the shared IP addresses and their interfaces are hosted on the active head node. On failover, the interfaces migrate and are hosted on the newly active head node. When remotely logging in to the head nodes for an HA system, users should use the shared external IP address for connecting to the cluster, ensuring connection to the active node. Similarly, inside the cluster, nodes should use the shared internal IP address to reference the head node. For example, nodes will mount the NFS filesystems using the shared virtual internal interfaces to ensure the imported filesystems can still be accessible in the event of a failover. Shared interfaces are implemented as alias interfaces on the physical interfaces (e.g. eth0:0). Again, they become active when a head node becomes active, and are deactivated when a head node becomes passive. Please refer to Figure 1 for the layout of a two-head failover network.

The Heartbeat Connection

In addition to the normal internal and external network interfaces on both head nodes, the two head nodes should also be connected via a direct dedicated network connection; refer to “eth2” in Figure 1. This interface, known as a heartbeat connection, enables the two head nodes to monitor each other’s availability. The monitoring is usually achieved with a regular heartbeat-like signal between the nodes such as a ping. If the signal times out, it implies a head node is dead. When setting up this interface, a simple UTP cable should be used between the Network Interface Controllers (NIC) of the two head nodes. This prevents a possible disruption from using a switch between the two nodes.

The HA setup also requires some type of shared storage between the two head nodes to preserve information after a failover sequence. The shared filesystems are only available on the active head node which is another reason users should log in using the virtual address. Logging in directly to the passive node could result in confusing behavior due to unmounted filesystems. The shared storage can be either Network Attached Storage (NAS) or Direct Attached Storage (DAS). With NAS, both head nodes directly mount a shared volume from the external device. Using DAS, both head nodes share access to a block device usually through a SCSI interface. Although the block device is visible and can physically be accessed simultaneously on both head nodes, the filesystem on a block device is typically not designed for simultaneous access. Attempting to simultaneously access the filesystem will most likely cause file corruption.

Identifying a Dead Node

Because of the risks involved in accessing a shared filesystem simultaneously from two head nodes, it is vital only one head node is active at any given time. Hence before a head node switches to the active role, it must receive confirmation the other node is in passive mode or is powered off. The conundrum remains: how do you tell when the active node is actually down? Is the active head node really down or is there a communication breakdown between the two head nodes? Because the “brains” of the cluster are communicatively “split” from each other (called the split brain situation), it is impossible to use only the direct interface to determine a course of action. On one hand, the head node could have completely crashed resulting in the lack of response. On the other hand, the direct interface between the head nodes might have failed and the active node is merrily working with the rest of system but has received notices that the passive head node has split from the system.

To deal with this uncertainty, a passive head node when facing this situation will transition into a “fencing” mode. The passive node does not try to kill the active node, but tries to obtain proof using other methods to determine if the active node is really down. While trying to decide the state of the active head node, the passive head node tags all subsequent actions as a backlog of actions to be executed later. If the head nodes are able to re-establish communications with each other before the passive node deems the active node dead, the fenced-off backup is compared and synced until the nodes are once again consistent. If the active head node is still communicating to the rest of the system, a failover should not be initiated. In a worse-case scenario, the failover could render the system unusable.

The STONITH Procedure

One technique used by Bright Computing’s Cluster Manager to reduce the chances of a passive head node unnecessarily powering off the active node is to perform a quorum procedure. When the active head node does not respond to any of the periodic checks for a period longer than the “dead time” in seconds (set by the user), the active head node is declared dead and a quorum procedure is initiated. The passive node polls all the processors in the system to confirm if they are in communication with the active head node. If more than half of the nodes confirm they are unable to communicate with the active head node, the passive node initiates the STONITH (Shoot The Other Node In The Head) procedure and becomes the active node. This is one instance where the messenger does not get shot. The STONITH procedure performs a power-off operation on the current active head node and then verifies the power is off. For automatic failover to be possible, power control must be defined for both head nodes.

The HA system can employ an automatic failover as described above or be configured for a manual failover. During a manual failover, the operator is notified of the problem and is responsible for initiating the failover procedure. Since no automatic power off is done, the operator must confirm the unresponsive node is powered off. The capability also exists to switch between automatic and manual mode during setup and while the system is running.

Could manual intervention be needed during an automatic switch over? The short answer is yes. Consider the situation where the active head node “is not dead yet”, meaning it is not fully functioning but not totally dead. “Mostly dead” is where the active head node passes some of the tests such as ping, status, etc. STONITH is the method used to guarantee a safe failover. However, STONITH sometimes fails to complete a clean shutdown when acting on a mostly dead active head node, which can result in potential problems with the filesystem or database states, or maybe the zombie node will remain in the middle of the transactions. If the system is set up for an automatic failover configuration, and a mostly dead head node is detected, the system will not automatically power down the active dead head. It will remain in the zombie state until it recovers, or the operator performs the STONITH manually to put it out of its misery. Before performing the shutdown, the operator must examine the situation for damage risk.

Another great feature for embedded systems is the ability to support HA for the compute nodes in the system as well as the head nodes. Similar to the head nodes, power control is needed for all regular HA nodes in order to carry out automatic failover for the ordinary nodes. Regular nodes use failover groups to identify nodes (two or more) to form a HA set. During normal operation, one member of the failover group is active, while the remaining ones are passive. For regular nodes, the heartbeat checks are performed using the regular internal network and not a separate failover network. The active head node performs the checks, and if a regular active node is deemed dead, it is powered-off through STONITH and a replacement node is brought online.

Managing Your HA Embedded System

At this point, you are probably intrigued by the concepts, but maybe a little intimidated at the thought of trying to configure and manage an HA embedded system. This is where the OpenHPEC™ Accelerator Tool Suite from Curtiss-Wright flies in to the rescue. Because of the similarity of the hardware, the maturity of the tools, and the larger installed user database, importing tools from the HPC community into your HPEC designed with the OpenHPEC Accelerator Suite can reduce both cost and time to deployment. The Bright Cluster Manager is best-in-class in the HPC world for setting up and maintaining clusters, including HA clusters. For example, Figure 2 shows how easy it is to setup an HA cluster for your system.

Bright Computer’s installation wizard starts by guiding you through the process of installing your cluster from bare boards to a full development system in a matter of minutes. By answering a few questions about the system you are building, Bright’s Cluster Manager configures all of the resources such as custom kernels, disks, and networks.

Using the image-based provisioning, kernel images can be maintained for different board types in the system, including GPUs. Adding, deleting or moving a board to another slot becomes as simple as a click. Bright Computing’s Cluster Manager can also simulate crashes to allow you to fully test your new HA capability. The OpenHPEC suite also features Allinea Forge, used in over 70% of the HPC installations, to provide true system level debugging and profiling. These tools form a fully integrated HPEC development environment that is fully tested, validated, and benchmarked. One solution loaded on one installation DVD with one part number from one company – Curtiss-Wright.

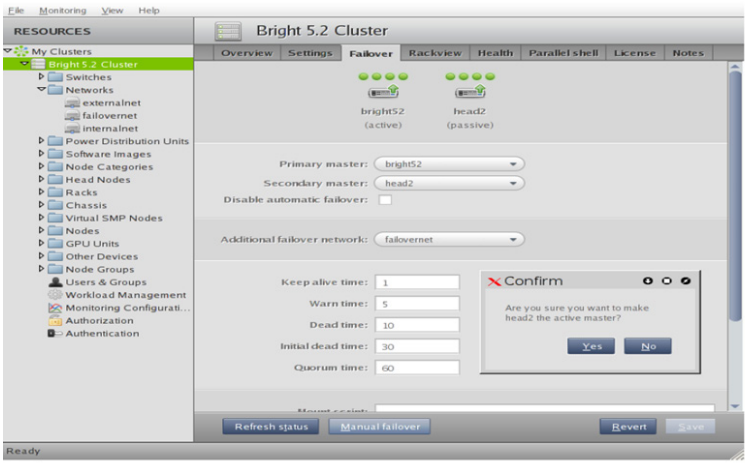

Figure 2: HA Cluster Setup

Authors

Tammy Carter, M. S.
Sr Product Manager
Curtiss-Wright Defense Solutions

Learn More

White Paper: Deploying HPC Tools into Embedded Software Development to Save Time and Money Products

Article: Turbocharge HPEC System Design with HPC Development Tools

Products

- OpenHPEC Accelerator Suite
- OpenHPEC Math Libraries
- CHAMP-XD1 (VPX3-482): 3U VPX Intel Xeon D DSP Processor Card
- CHAMP-XD2 (VPX6-483): 6U VPX Intel Xeon D DSP Processor Card
- CHAMP-GP3 (VPX6-492): 6U OpenVPX GPGPU Processor Card with NVIDIA Maxwell
- VPX6-6902: 6U OpenVPX Serial RapidIO and Ethernet Switch