

The Evolution of The Linux-HA project

Alan Robertson
Linux Technology Center
International Business Machines Corporation
alanr@unix.sh OR alanr@us.ibm.com

Abstract

The Linux-HA project provides software-based high-availability clustering services which are used to improve the availability of many kinds of services. The Linux-HA project is used in thousands of sites across the world, and has been in production use for nearly five years.

Although the project is named *Linux-HA*, the project software is quite portable, and the software runs on other OSes including FreeBSD and Solaris.

This paper will present an overview of the past, present, and future capabilities of the Linux-HA software, including a longer-term vision for a comprehensive framework for high-availability services for the future.

Project Origins

The Linux-HA project started off in 1997 as a mailing list created by Harald Milz to discuss how one might create a set of high-availability capabilities for Linux¹. In the process, Harald created an unusual developer's HOWTO, which explained how one such HA services might work if one were to write them.

At that time, the author worked for Bell Labs, and was investigating Linux as a platform to use in products and for a development environment. As a part of this job, he was asked to investigate what kind of HA capabilities Linux had. The author discovered the HOWTO, and the mailing list, but there was not yet any software. In 1998, the author decided to write a piece of software which would perform one of the core functions mentioned in the HOWTO – the heartbeat function. Naturally, the software was called “heartbeat”. It was a simple design, operating only over serial ports, and providing only detection of server death via its heartbeats. The intent of this software was to provide a core piece of software to get started with, a sort of “seed crystal” for the Linux-HA project to help jump start the project. Although the project has expanded far beyond the initial heartbeat function, the software is still often just called “heartbeat”.

Early Capabilities

The earliest versions of heartbeat added UDP broadcast heartbeats, and could run scripts on startup and on server death. It could negotiate release of resources when a node returned to a cluster, and was tested by hand. It wasn't yet suitable for managing shared storage clusters, because it had no fencing capabilities. Nevertheless, there were a few early users who found its simplicity, ease of configuration and robustness much to their liking. The first production installation of heartbeat occurred in 1999.

1 Linux is a trademark of Linus Torvalds.

Over time, the project added strong authentication of packets, a basic plugin system used for communication and authentication. It also created a mechanism for killing cluster nodes, called STONITH (Shoot The Other Node In The Head), which was subsequently adopted by other HA packages. The addition of the STONITH system provided a robust mechanism for controlling unique access to storage. This brought heartbeat into the category of being able to reliably manage shared storage.

An API was added, a general plugin system was written, and a new IPC system was added. New failover models were added, internal restructuring was performed which significantly improved portability. Two new capabilities (a consensus membership service and the ipfail connectivity monitor) took advantage of the API.

At this point, it became a platform which was interesting to a wide audience including telecommunications vendors. As a result, the project began receiving contributions to help telecommunications (and other HA-centric) applications perform their jobs better. Intel became a serious contributor to the project during this period, because of their interest in telecommunications systems.

Current Capabilities

As of this writing, Linux-HA has the following set of features:

- Two-node HA (failover) clusters
- Active/Active and Active/Passive configurations
- Quick and Easy setup
 - Ability to use standard init scripts as resource agents
 - Integrated IP address takeover
 - Integrated filesystem support
 - Integrated email notification
 - Native integration with DRBD filesystem replication
- Multiple failback models
- API for monitoring and control
- Highly extensible
- Strongly authenticated communications
- Multiple communication media supported (broadcast, multicast, unicast, serial ports)
- Unlimited redundant communication media
- Integrated connectivity monitoring using voting protocol
- System watchdog timer support
- Automated test suite
- Support for OCF standard resource agents
- Sub-second failure detection
- Basic application heartbeat monitoring support
- Support for the SAF data checkpoint API

Two-node HA (failover) clusters

The majority of failover clustering applications are basic two-node configurations. This is the minimum configuration for HA clustering, and can be configured to eliminate all single points of failure.

Active/Active and Active/Passive configurations

In two-node arrangements, if one server provides all the services, and the other one

acts primarily as a backup server, this is called an *active/passive configuration*. If both servers are configured to provide high-availability services, and each will back up the other in a mutual failback arrangement, this is called an *active/active configuration*. Heartbeat supports clusters in both configurations.

Quick and Easy Setup

Linux-HA has a reputation for being very easy to get started with. The average setup for a first-time user takes between a half hour and two days, with the most common reports being approximately a half day. Much of this ease of setup and use comes from avoiding complexity in its conceptual model and design. Beyond that, there are a number of factors which contribute to its reputation for ease of use.

In order to integrate particular applications into an HA clustering arrangement, the simplest method is to write resource agent scripts to control the application. In order to minimize this effort, heartbeat is able to directly use standard Linux (LSB) `init(1)` scripts as resource agents. This means that if the application or service comes with an `init` script to start, stop, and monitor it, then heartbeat can immediately control that application or service with little or no additional work on the administrator's part. No special kits or add-ons or user-written scripts are required in most cases. By following the Linux standard for service control, writing such scripts is easy, since the function of `init` scripts is well-known to most system administrators.

Heartbeat has integrated IP address takeover. That is, it can take over any virtual IP address through a built-in resource agent for IP addresses. This agent is very simple to use, and in most cases all that is required is to tell heartbeat the IP address to take over. It then deduces the correct interface, netmask and broadcast address automatically.

Heartbeat has integrated support for filesystem mounting and unmounting. If a shared disk is required in a particular configuration, then heartbeat can likely support it immediately. It also has special support for IBM's ServeRAID shared SCSI RAID controllers, and for the ICP Vortex RAID controllers.

Heartbeat has easy-to-configure integrated support for providing email notification for the takeover and release of any resource group. This capability is provided in the form of a resource agent which can be added to any resource group for which email notification is desired.

The DRBD (Data Replicating Block Device) service provides a very cost-effective replacement for costly shared disks in many lower-end applications. DRBD provides native support for Linux-HA in its basic out-of-the-box setup.

The Linux-HA project provides the *ldirectord* component. *Ldirectord* provides native support for high-availability load balancers using the standard Linux *ipvs* service provided by the Linux Virtual Server project.

Multiple failback models

Linux-HA provides user-selectable failback control. If the **autofailback** option is enabled, resources automatically migrate to their preferred server when it is available. This is required if an active/active configuration is desired. If it is disabled, then resources only change servers when failures occur. This is generally the preferred

2 IBM is a trademark of the International Business Machines corporation.

option for active/passive configurations.

Highly extensible

Like many open source projects, the Linux-HA project is highly extensible. It provides four primary ways to be extended. First are resource agents which were mentioned earlier. Second, Linux-HA provides an API which can be used to write applications which can monitor and control it. The connectivity monitoring and control application described below is written this way – as is the membership service. Third, Linux-HA has several system exits – ways for users to do whatever action they desire at important times. Last, heartbeat has three kinds of plugins which can be used to provide more extensive customization of the software. These plugins are in the areas of authentication, communication, and fencing (STONITH). In addition, as an open source project, we constantly accept patches for bug fixes and new capabilities, providing the ultimate in extensibility.

Multiple communication media supported (broadcast, multicast, unicast, serial ports)

Linux-HA has a central communications core which uses plugins to implement communication over different media. The currently supported communications media include broadcast, multicast, unicast IP-based communications, and serial ports. In addition, two specialized ping plugins are provided for communicating with non-cluster entities for monitoring connectivity. There is no practical limit to the number of different types or combinations of communications media which can be configured.

Strongly authenticated communications

One way of thinking about a poorly-managed cluster system is to say that “a cluster is a computer whose backplane is the Internet”. Since from a security standpoint, the Internet is probably the most hostile environment known, this is a frightening thought indeed. Unfortunately, it is difficult to guarantee that all systems are well-managed, or stay that way once installed. It is an unfortunate fact that the more reliable a system is, the less quality attention it is likely to get from the administrative staff. In order to deal practically with these circumstances, it is desirable for an HA system to use strongly authenticated communications, in order to be as resistant to attack as possible. In addition, Linux-HA has been subject to several security audits. As a result, Linux-HA has been installed and used quite successfully in some reasonably security-sensitive environments.

Integrated connectivity monitoring using voting protocol

The ipfail module monitors configured ping nodes as has been described earlier, and, using the API and redundant communications, implements a voting mechanism that determines the comparative states of communications in the cluster. It then uses this information to move services off nodes with failed communications and onto those with better connectivity. Since this mechanism can be configured to test communications to any number of systems at any distance away, it is generally superior to methods which only monitor the local state of communication links.

System watchdog timer support

Linux provides a software-based system watchdog timer (softdog), and support for several hardware watchdog timers. The Linux-HA software can be told to register itself

with any of these timers, and should heartbeat or the operating system fail, the system would then automatically reboot.

Automated test suite

Linux-HA is tested extensively before each release with an automated test suite. This suite has contributed substantially to the quality of the Linux-HA system. Like the rest of Linux-HA, it is extensible. As a result, it can be used to test particular customer configurations for the ultimate in customized assurance.

Basic support for OCF standard resource agents

The Open Cluster Framework (OCF) is an effort associated with the Free Standards Group to create a API standard for Linux clustering. They have defined an API for resource agents. This API is very similar to the standard heartbeat resource agent API. After this standard was defined, heartbeat added basic support for OCF resource agents.

Fast failure detection

One of the important factors which HA clustering tries to improve over non-HA systems is mean-time-to-repair (MTTR). For an HA clustering system, the MTTR is significantly influenced by the time it takes the HA system to detect the death of a cluster member. Most HA systems can reliably detect failure in a time between 10 and 60 seconds. The Linux-HA system can be configured to detect failure in less than a second. This feature is critical for systems like certain telephony applications for which availability is critical, and which can fail over quite rapidly once failure has been detected.

Basic application heartbeat monitoring support

Although Linux-HA supports applications without the applications making any particular provisions for their management by an HA system, certain applications (like some telephony applications) can be improved by providing specific HA support. One of the ways in which the Linux-HA project currently supports such applications is through an application heartbeat service, provided by the *apphbd* daemon. Applications which wish to do so may register themselves with *apphbd*, and then periodically provide an application-level heartbeat to *apphbd* to assure it of their continuing sanity. Should they fail to heartbeat in the agreed-upon interval or disconnect unexpectedly, *apphbd* will notify the recovery manager, which can then restart the application. This is the first feature in the Linux-HA package which does not require HA clustering to operate. It works quite well on a single stand-alone machine, or one which is a member of an HA cluster.

Support for the SAF data checkpoint API

In a normal "enterprise" type application, much or all of the state which an application needs to take over from another machine is stored on disk. Therefore, simply providing disk sharing or replication allows other applications to take over quite easily. However, certain applications (networking or telephony applications) have state which changes too fast to write to disk, and/or the time required to error check and restart a disk-based application is too great. For those applications, the Service Availability Forum (SAF) has designed a data checkpoint API. The SAF data checkpoint API provides for the rapid replication of data across a network without involving disks. The Linux-HA project provides an implementation of this API which allows applications which use the API to take over services very quickly. Combined with the fast failure detection, it is an

excellent tool to use in designing availability-critical applications. This API can be used either locally (in conjunction with the application heartbeat (*apphd*) service, or across the cluster (using the *heartbeat* service).

Planned Capabilities for 2004

Although these features provide a significant set of capabilities which meet the needs of many users, still many applications would be helped by some additional capabilities. Fortunately, several companies have seen the need for these capabilities and have stepped with staffing dedicated to creating these new capabilities.

The list of features the project is currently planning on adding in 2004 is detailed below.

- Local resource management and monitoring
- Larger Clusters (new resource management scheme)
- Basic management and command line status utilities
- Improved documentation
- Support failover to second ethernet on same machine
- General system health monitoring (service preconditions)
- “Warm” failover
- SAF event and messaging APIs
- Platform-specific STONITH and watchdog driver support.
- OCF standards compliance – resources and membership

Local resource management and monitoring

The most commonly cited defect in the current Linux-HA software is its inability to natively monitor resources which it manages. That is, if one is managing, for example, an apache web server, heartbeat makes no effort to monitor this web server for correct operation. It simply assumes that it is working correctly. Once this feature is available, heartbeat will provide built-in automatic monitoring of resources without any additional configuration.

Larger Clusters

Another commonly-cited defect is heartbeat's current two-node cluster limitation. Although we expect a significant majority of clusters to continue to be two-node arrangements, there is still a need for larger clusters – particularly for more sophisticated applications. Once this feature is available, heartbeat will support much larger clusters. This is a significant undertaking, requiring a redesign of the cluster resource management layer. The resource monitoring service will integrate with (and provide services for) this new cluster resource management layer.

Basic management and command line status utilities

Currently there are no tools to monitor and few tools to manage the Linux-HA software. It is our plan to add some basic monitoring and management capabilities to Linux-HA.

Improved documentation

Although the current Linux-HA documentation has proved to be reasonably effective, understanding it and deciding how to use it effectively for particular applications can be intimidating for the novice. It is our plan to add new documentation describing how to use it for specific common configurations, thereby decreasing the “intimidation factor” for the package, and giving it a bit more professional appearance.

Support failover to second ethernet on same machine

Although heartbeat is quite willing to fail over to another machine when an ethernet adapter fails, in some cases, such failovers create a noticeable disruption of service. In these cases, it is desirable to fail over to another ethernet adapter on the same machine, thereby minimizing the impact of the failure.

General system health monitoring

As mentioned previously, it is already planned to add the ability to monitor resources. However, it is also desirable to monitor things which Linux-HA doesn't control. For example, one might monitor motherboard temperature, or UPS battery state, or other similar things in order to decide to fail over to another machine whose state might be better. This could be viewed as a generalization of our current connectivity monitoring application (ipfail).

Hot failover

The normal enterprise model of a service is that it is either running or not running on any particular cluster member. However, some kinds of services (like those that use the SAF data checkpoint API), operate in three states: not running, running as primary, and running as secondary. We use the term “hot failover” to refer to these types of applications which actively keep the secondary apprised of state at all times.

SAF event and messaging APIs

The project plans on adding support for additional SAF APIs. The next two APIs to be supported are the SAF event and messaging APIs.

Platform-specific STONITH and watchdog driver support.

Heartbeat has two areas in which it is slightly platform-dependent. It has a fencing component (called STONITH) which needs to know how to reset a particular node in the cluster. It also can make good use of a platform-specific watchdog timer, if one is available. It is planned to add specific support for a few additional IBM platforms in the next year. Of course, we will continue to incorporate enhancements for other platforms as the community provides them.

OCF standards compliance – resources and membership

Although heartbeat has basic support for the OCF resource API, the project currently plans on adding complete OCF resource support, and also support for the OCF membership API.

A Vision for Comprehensive High-Availability Services

Looking at the plans outlined previously, it is apparent that Linux-HA will be competitive with the majority of commercial HA packages. As a result, it will be suitable for a the vast majority of enterprise-level high-availability clustering applications.

Nevertheless, even after all these capabilities become available, there is still room for availability improvement when one considers telecommunications, and other availability-sensitive application areas, and improving availability for machines which are not part of clusters.

As of the present time, there has been little connection between the HA services provided by HA clustering systems like Linux-HA, and those provided by the operating system itself. But, many of the kinds of HA services and monitoring capabilities which HA systems provide for their cluster members are quite useful in standalone (single-machine) environments. Examples of services which fall into this category include the ability to fail over to a redundant ethernet link automatically, and general system health monitoring. Since the Linux operating system is a truly open environment, the possibility for synergy between these HA cluster systems and standalone systems is very real.

In addition, the telecommunications world has certain paradigms which they commonly follow which are of interest in any system for which availability is important, and for which professional staff are responsible.

The remainder of this paper will briefly outline a proposed Recovery Manager based on these concepts which would be useful both for local and clustered computers.

The Recovery Manager

It is the purpose of the Recovery Manager (RM) to accept reports of exceptional events from a variety of sources, and then according to configured policies, perform appropriate recovery actions designed to respond to and recover from failures.

Exceptional events are events which are thought to have an impact on service availability or performance. The range of exceptional events is obviously wide, and can be clarified by a few examples. Exceptional events might include motherboard temperature out of range, low disk space, excessively high paging rate, low UPS batteries, network connectivity failure. It is specifically intended that most exceptional events have both positive and negative forms. For example, in addition to a *connectivity lost* event, there would normally be a corresponding *connectivity restored* event. Not every event has a such a corresponding recovery event. For example, an error indicating a correctable memory error, would not likely have a corresponding positive (or recovery) message.

It is envisioned that the RM would be structured into four basic modules – an event reception module, a filtering module, a policy module, and a recovery action module. Each of these modules is briefly described below.

The event reception module has the function of performing authorization of event sources, receiving the events and passing them on to the policy module.

It is the purpose of the policy module to identify the type of event received, and then route incoming messages either to a filtering module or to a recovery action module

according to the policy defined for that particular message type.

A filtering module is a type of recovery action module which accumulates events until they reach some predetermined threshold, and then, in effect, escalate the problem back to the recovery module, for further classification and processing. Note that this event type must be a different type than the original original message, and probably contain much or all of the information from the original messages which led up to the escalation. Note that a filtering module can take messages which have no natural recovery message (like the ECC RAM hits mentioned earlier), and produce a recovery message after a period of time has elapsed.

A recovery action module is one which takes some action intended to lead to recovery from the original problem. Recovery modules can take a wide variety of forms, from switching to a secondary ethernet adapter, to initiating a backup of a disk reporting errors, to informing the HA system to fail over to another cluster member, to escalating to an administrator via email, to issuing a trouble ticket in a trouble ticketing system, to gracefully shutting down a system. Depending on the success or failure of it's recovery actions, a recovery action module can also create a new event to cause escalation to a new exceptional event, in a fashion similar to a filtering module.

Although this architecture is fairly simple, the ability to create new events out of old ones through filtering or escalation, makes the system very powerful. It is expected that most of these components use plugins to implement filters, policy types and recovery actions. As a result, there is very little such a system cannot be made to do.

As an example of the kinds of actions one might envision such a system performing, an pending failure of disk reported through SMART will be considered.

In an HA cluster system, a policy could be configured to force failover to another system in the cluster. If the data is protected through software RAID, a policy could be configured to simply notify the administrator. If the data is on a desktop system, an automatic backup of the system could be initiated. In the right kind of environment, a recovery action could determine the type of disk, order a replacement, and create a trouble ticket for the impending failure.

The point of this example is not that one will find any particular action given in the example to be a good one, but that one will likely be able to make this work in a very wide variety of environments ranging from desktops, through enterprise servers through HA clusters, through turnkey embedded systems.

Summary

The history and future of the Linux-HA project has been presented. Linux-HA is currently used in thousands of sites all across the world, and is in the midst of significant improvements which will make it usable for the overwhelming majority of enterprise-class server applications. This paper further outlined a proposed recovery manager which is usable in a wide variety of circumstances which will likely improve the availability and manageability of Linux in a meaningful way.