IBM Systems MAGAZINE

# Doubly DEPENDABLE

## Linux and open-source HA build on mainframe's strengths

Mainframes are renowned for their dependability, providing a stable and reliable platform for unmatched availability to mission-critical business services. Linux* and open-source software (OSS) in general also have a reputation for providing significantly above-average quality. For years, Linux has been widely trusted with mission-critical services, which is reflected in the worldwide deployments of Linux on System z*.

**Lars Marowsky-Brée** is a SUSE Distinguished Engineer and the founder of the Linux Foundation HA Working Group. He serves as the architect for High Availability and Storage at SUSE.

To increase the availability of mission-critical workloads, efforts have been made to greatly reduce the likelihood of component failure. Systems have long been combined to form clusters in which their capacity beyond the immediate runtime requirements of a workload is used to compensate for individual components' faults. This is in contrast to clustering for high performance computing (HPC), where added capacity is used to boost workload performance. While some overlap in the technology exists, high availability (HA) and HPC have different priorities and goals.

For HA clusters, redundant components must be added intelligently so the redundancy they provide improves the availability of the workload cluster hosts. The architecture must include at least one level of redundancy for every potential component failure, be it in hardware or software. Components without redundancy but mandatory for service delivery are called single points of failure (SPoF). It's not always cost-

effective to remove all SPoFs. Ultimately, the risk vs. cost trade-off is a business decision.

Especially on System z, this quality and redundancy have been built deep into the architecture and are transparent to the OS running in a virtual instance. On other architectures, it becomes the task of the OS and its middleware to:

- Combine individual nodes into a more dependable cluster
- Identify faulty components
- Handle recovery by switching to a backup network interface or a different storage path, or migrating the service from one node to another

An HA cluster stack is a software suite capable of such management. Linux features one of the most advanced, comprehensive and fully OSS implementations, provided primarily through a combination of the Corosync (membership and messaging) and Pacemaker (policy-driven resource management) projects. Growing from modest two-node heartbeat clusters in the 1990s, this stack became the de facto standard for OSS HA and has been widely adopted by Linux distributions.

While mainframes have a high degree of this functionality built into their hardware and firmware—making application-visible hardware faults extremely rare—clustering the individual instances still has advantages. The cluster stack provides hard consistency guarantees and coordinates access to shared resources. This is used by cluster-concurrent file systems such as OCFS2 and GFS2. It not only protects against hardware failures but also monitors the software and initiates recovery accordingly through restarts of the services or rebooting the instance.

## The Bigger Picture

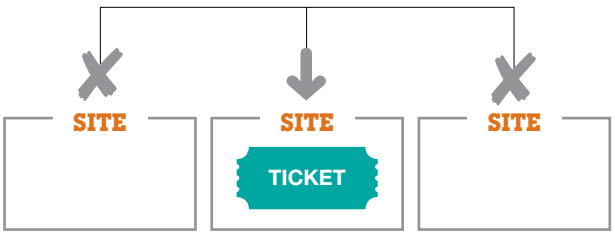
Even the most reliable hardware and data center cannot possibly cope with all eventualities when disasters strike. Therefore, a global organization must be able to compensate. With such scenarios, the focus shifts from single-component failures to a systemic view of the infrastructure. The commonly proposed mitigation

strategy is to build data centers at geographically dispersed locations. However, distance translates to unavoidable latency due to the laws of physics (speed of light), and typically higher costs for network bandwidth.

Traditional local clusters—able to exploit low latency between nodes—are coupled tightly with synchronous coordination and replication of data with coherency guaranteed at multiple levels in the stack. It isn't feasible to maintain such tight coupling across distant geographies.

Hence, geographic clustering is most commonly implemented as an asynchronous, loosely coupled active/passive scenario, meaning only one site is active for a given workload. The other sites are passive replication targets, only becoming active after failure of the primary site. Because data replication is asynchronous, such a failover will generally incur a minimal loss of the most recent transactions. This is usually deemed preferable to not providing any service at all.

**TOP-LEVEL MANAGERS** allocate "tickets" on which resource hierarchies depend




Only one site is granted a specific ticket at a time

## The Next Level

Geographic clustering is available as an OSS extension to Corosync/Pacemaker clusters. It's assumed each site is a largely autonomous cluster itself, taking care of local storage, recovery, fencing, failure detection and resource hierarchy management. The new component only coordinates recovery at the next level.

To achieve this, two or three of these sites are coupled together in a "cluster of clusters." The top-level manager arbitrates the allocation of so-called "tickets" upon which the resource hierarchies depend. This ensures only one site is granted a specific ticket at any given time, and thus allowed to activate the resources this protects.

This arbitration is handled via an implementation of the PAXOS algorithm, and the project managing the tickets is called the booth. All sites vote on which site is to be granted a ticket and the majority vote wins. The system supports arbitrator sites that aren't true clusters themselves but only participate in the voting process.

To allow automatic and safe allocation to a given site, it must first be safely established that the previous owner has fully relinquished a given ticket. This is easy if all sites are up and connected. Should a site become permanently disconnected, however, the booth process running there will notice and revoke the ticket locally by force.

To speed up the recovery process, Pacemaker will immediately fence all nodes hosting resources that depend on the ticket. This is necessary to make the cleanup time predictable. Because a disconnected geographical cluster doesn't allow for fencing to be acknowledged by remote sites, automated failover is only possible via a timeout-based mechanism. The timeout must allow for a disconnect to be detected and the node-fencing process to complete. Once this time has expired, the surviving majority can re-grant the ticket to one of the remaining sites.

Some organizations implement a policy where this isn't deemed adequate. For these scenarios, or for those without a third tiebreaker site, a manual process of revoking and granting tickets is provided.

The changes required to Pacemaker's core are minimal. The tickets are represented as newly added clusterwide attributes. A new constraint type is added to allow resources to depend on them. This is fully supported in Pacemaker and the cluster resource manager shell. The Web interface "hawk" also displays ticket ownership and provides a dashboard of multiple clusters.

## Ongoing Improvements

While this framework lays the foundation for building geographically distributed clusters exclusively based on OSS technology, the OSS community continues to explore and implement further enhancements.

Storage replication is one such area. While the Distributed

**Mainframe-based infrastructures can profit from advanced OSS HA cluster technologies available on Linux.**

Replicated Block Device included with Linux can handle storage replication on a per-host level, resource agents are needed to properly interface with third-party storage arrays to tie these into larger disaster recovery concepts.

Network access is critical. While a proof-of-concept implementation to automatically update dynamic DNS records exists, it's desirable to add integration with dynamic routing protocols, such as OSPF or BGP4, to make the network layer itself aware of changes. One potential solution for this requirement is integrating with routing software.

Synchronizing configuration to avoid divergence between the sites is another area of interest because manual replication is error-prone. The HA stack could replicate cluster-relevant configurations, such as the cluster information base or external configuration files, as well as apply transparent transformation to account for differences between the sites.

A key focus of industry contributors, including distribution vendors, ISVs, consulting partners and key customers, is to develop best current practice whitepapers to guide implementation in the enterprise of the whole stack—from the OS, the HA cluster software, the application workload and operating procedures. This feedback also informs future development of the OSS HA stack.

## Open-Source HA and Linux

Mainframes and Linux benefit from leveraging each other's strengths. Mainframe-based infrastructures can profit from advanced OSS HA cluster technologies available on Linux. Open-source HA and Linux on System z have come a long way since their humble beginnings.

Today, Linux has earned its place as a trusted enterprise OS. Multinode HA clusters on Linux are widely deployed and reliable with a proven track record. Customers and solution providers are actively seeking to deploy Linux-based, open-source HA solutions for even their highest-tier infrastructure deployments, and the architecture to implement these now exists.