High-Availability Using Open Source Software

Luka Perkov
Iskon Internet d.d.
Garićgradska 18, 10000 Zagreb, Croatia
E-mail: luka.perkov@iskon.hr

Nikola Pavković
Ruđer Bošković Institute
Bijenička cesta 54, 10000 Zagreb, Croatia
E-mail: nikola.pavkovic@irb.hr

Juraj Petrović
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb, Croatia
E-mail: juraj.petrovic@fer.hr

***Abstract*—Availability, the degree to which a system is in a specified, operable state, is a critical criterion in overall information systems' quality. Since network services are the core infrastructure components for any modern business, their uninterrupted functioning becomes the critical issue in sustaining the business itself. Deployment of a production ready system to ensure high availability is not an easy task. Additionally, implementation of highly available software solution using existing resources, without or with minimal investment is a common requirement in many business environments. Various commercially available solutions exist on the market, but the open source community also provides a number of mature, production-ready tools and techniques aiming at the same market. Comparing the TCO of both solutions, the open source approach represents the most economical, but nevertheless reliable solution.**

This paper gives an overview of the most commonly used tools and techniques to ensure high availability of information systems' network services using only open source software.

I. Introduction

High availability is based on the use of redundant computers organized into a structure that is called a computer cluster. Computer clusters can be categorized into three groups:

- High Availability clusters (HA),
- Load Balancing clusters (LB) and
- High Performance Computer clusters (HPC).

A HA cluster consists of a minimum of two computers, which are referred to as nodes. If the event of complete node failure occurs, the service provided by a HA cluster should continue to operate normally and recover to the initial state successfully upon node recovery. In LB clusters, a small number of front-end nodes divides the work among a large number of back-end nodes that perform actual service provision. Small computation tasks are equally distributed between nodes in a HPC cluster. HPC clusters are used mainly for a wide range of scientific calculations, from bioinformatics to meteorology, wherever there is a need for high computing power that average computers do not and can not provide.

By purpose, as well as in appearance from the outside, the HA and LB clusters are quite similar. Their goal is to provide high availability of network services, regardless of the backend node failures. Both are fully transparent to users – from the outside, the cluster seems like only one computer instance that reliably and continuously provides the service. Differences between these two cluster types are contained in the internal system architecture and the underlying software stack providing particular functionality.

II. HA overview

When designing a HA system one should use a combination of loosely coupled components to achieve the desired goal. These components are called resources and could be perceived as HA cluster building blocks. Three main resources, where HA implementation would be advisable, are:

- storage,
- application and
- operating system.

Storage HA selection has the most impact on HA cluster performance and should be chosen wisely. It's main goal is to prevent data loss and to provide data to other cluster resources in an uninterrupted manner. DBMS (Database Management System) are the most common applications with some form of HA functionality built in. Operating system HA is achieved through OS virtualization. With help of OS virtualization technology, it is possible to create highly available virtualized systems, in a way that a virtualized system is not aware of the underlying HA infrastructure. These resources, as well as some of the available software solutions, are further discussed in the following sections.

In order to automate the failover process in a HA cluster it is recommended to use resource management software tools instead of writing custom scripts or tools for resource management. Later described resource managers support a wide range

of platforms, architectures and resources. Furthermore, they have been thoroughly tested and deployed in a large number of production environments.

III. Storage

Since data is considered as the most valuable asset, it should be preserved in the event of failure. Even though some file systems provide support for distributed storage and node recovery, they are not discussed in this paper. Instead, here described technologies are presented as *virtual* storage block devices to the host OS. This setup makes it possible to choose from variety of file systems that can be used on any storage device.

A. DRBD

DRBD (Distributed Replicated Block Device) is a distributed storage system used to provide functionality similar to RAID 1. It's designed to run over the network and can be used by no more than three nodes to provide data redundancy. In a nutshell DRBD provides:

- primary and secondary, as well as dual primary node setup,
- synchronous and asynchronous operation mode,
- automatic recovery after any type of failure and
- advanced resynchronization management.

DRBD architecture is shown in fig. 1.

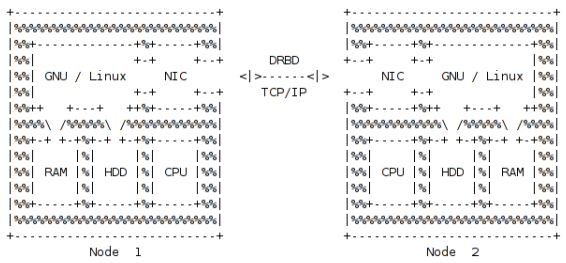

Figure 1. DRBD architecture

B. Open-iSCSI and iSCSI Enterprise Target

iSCSI protocol allows initiators (clients) to send SCSI requests to SCSI targets (storage devices) on remote servers. iSCSI allows organizations to consolidate storage into data center storage arrays while providing hosts with the illusion of locally-attached disks. One target can provide iSCSI service to many initiators. iSCSI architecture is shown in fig. 2.

Open-iSCSI provides iSCSI protocol initiator implementation for the Linux kernel, while iSCSI Enterprise Target can be used as iSCSI protocol target implementation.

IV. Application

DBMS are one of the few applications that can provide independent but also incomplete HA support. Although DBMS HA implementations often provide node recovery, they are almost always seen in LB cluster setup only if internal application HA is used. In the following sections PostgreSQL and MySQL basic HA clustering options are discussed.

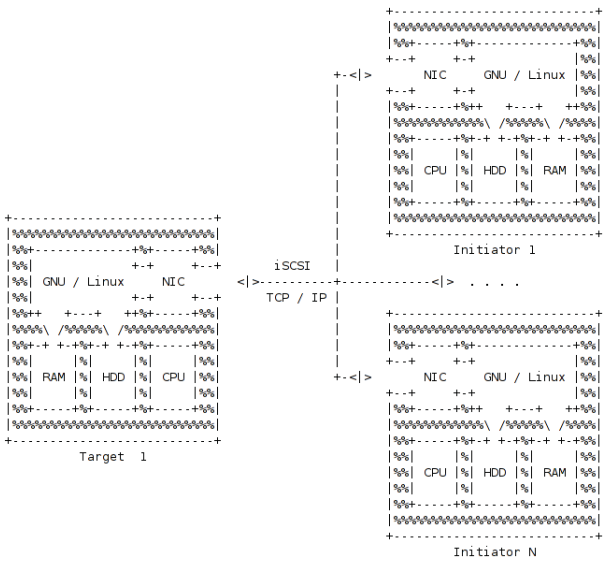

Figure 2. iSCSI architecture

A. PostgreSQL

PostgreSQL DBMS supports write ahead log (WAL) archiving on slave nodes that can be used for HA deployments. This solution will only satisfy a non demanding user. However, PostgreSQL has a variety of third party tools and plugins one can choose from in order to enhance HA clustering options. Some of them are:

- pgpool-II,
- SkyTools,
- Slony-I and
- PGCluster.

pgpool-II offers wide range of options, like connection pooling, replication, load balancing, limiting exceeding connections and parallel query execution, to fine tune PostgreSQL cluster performance. Most of PostgreSQL clustering solutions are based on asynchronous operation mode.

B. MySQL

MySQL DBMS provides both asynchronous and synchronous replication out of the box. MySQL uses asynchronous replication by default – which is more convenient for many reads, while synchronous replication should be used in databases with many writes. Asynchronous replication allows transactions to be written on master node only and in synchronous replication mode data can be written on any node in the cluster.

V. OS Virtualization

OS virtualization must be used in order to provide full OS HA. Linux has been supporting various types of emulation and virtualization methods for a long time. Only in a last couple of years these solutions have become mature enough to be used in mission-critical environments. One of the features that all of described solutions provide is *live* migration, which is

convenient when migrating virtual machines (VM) between physical nodes.[1]

In the following sections Xen, KVM and OpenVZ OS virtualization solutions with their fundamental concept of operation are described.

A. Xen

Xen can perform full OS virtualization on systems that support virtualization extensions in hardware, but can also work as a hypervisor on machines that don't have the virtualization extensions. Xen is placed directly on the hardware and treats each OS used to deploy Xen as a guest VM.[2] It implements a model known as paravirtualization that provides better performance over other virtualization techniques.

Xen architecture is shown in fig. 3.

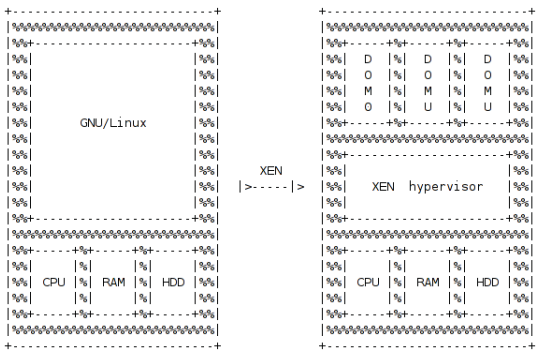

Figure 3. XEN architecture

B. KVM

KVM is a hypervisor contained in the mainline Linux kernel. KVM can turn the standard Linux kernel into a hypervisor by loading the appropriate kernel module. Therefore, optimizations to the standard Linux kernel components benefit both the hypervisor and the Linux guest operating systems.

Because KVM was designed after the advent of hardware assisted virtualization, it did not have to implement features that were provided by hardware. By requiring hardware support rather than optimizing with it if available, KVM was able to design an optimized hyphenation solution without requiring to support legacy hardware or requiring modifications to the guest OS.

KVM architecture is shown in fig. 4.

C. OpenVZ

OpenVZ is a container-based virtualization solution for the Linux OS. This model uses a patched Linux kernel that can only run guest Linux OS in isolated containers. All virtualized containers have to be compatible with the Linux kernel version that the host runs on. However, because it doesn't have the overhead of a true hypervisor it is very fast and efficient. The

[1]During live migration procedure, the memory of the virtual machine is copied to the destination without stopping its execution.

[2]Xen VM used for management of Xen and other VMs is called DOM0, while all other VMs are referred to as DOMU.

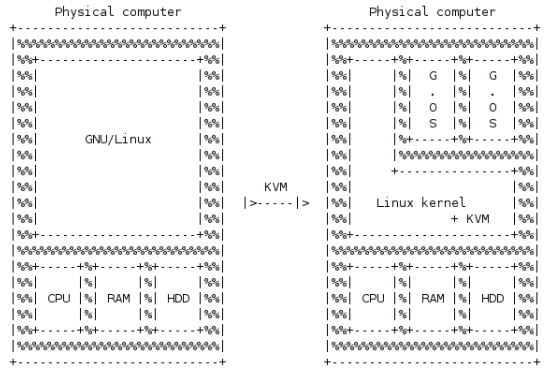

Figure 4. KVM architecture

obvious big disadvantage is the single kernel model, which leads to some functionality limitations of virtualized containers but also raises some security issues.[3]

OpenVZ architecture is shown in fig. 5.

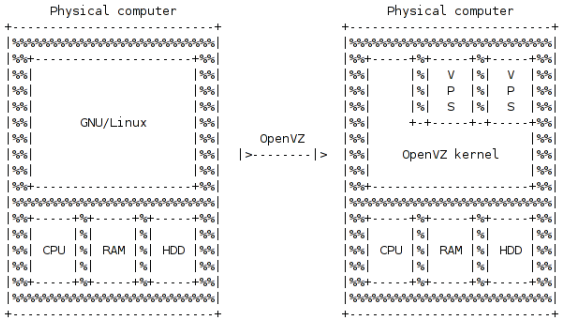

Figure 5. OpenVZ architecture

VI. HA CLUSTER RESOURCE MANAGEMENT

In order to fully automate recovery by detecting hardware or software failures it is essential to use appropriate tools which provide advanced cluster node and resource management operations. Only when adequate HA cluster resource management (CRM) tools have been deployed, cluster management is both adjustable and extensible. In the following section, Pacemaker CRM tool which can meet requirements of most HA cluster implementations, is described.

A. Pacemaker

Pacemaker achieves maximum availability for defined cluster resources by detecting and recovering from node and resource-level failures by making use of the messaging and membership capabilities provided by cluster infrastructure.[4] At the highest level, pacemaker cluster architecture consists of three pieces:

- core cluster infrastructure providing messaging and membership functionality (eg. Corosync or Heartbeat),

[3]In case a OpenVZ virtual machine is compromised, further kernel-level penetration could provide the attacker with means of gaining control of other VMs hosted on the same physical computer

[4]Pacemaker is a continuation of the CRM that was originally developed for Heartbeat but has since become its own project.

- non-cluster aware components and
- a brain, processing the events from the cluster and configuration changes.

Pacemaker makes no assumptions about desired environment, which allows to provide support for practically any redundancy configuration including Active/Active, Active/Passive, N+1, N+M, N-to-1 and N-to-N. Pacemaker's key features are:

- detection and recovery of node and service-level failures,
- storage agnostic, no requirement for shared storage,
- resource agnostic, anything that can be scripted can be clustered,
- supports STONITH for ensuring data integrity,
- supports large and small clusters,
- supports both quorate and resource driven clusters,
- supports practically any redundancy configuration,
- automatically replicated configuration that can be updated from any node,
- ability to specify cluster-wide service ordering, colocation and anti-colocation,
- unified, scriptable, cluster shell and
- support for advanced services type:
  - clones: for services which need to be active on multiple nodes or
  - multi-state: for services with multiple modes (eg. master/slave, primary/secondary).

Example Active/Pasive HA cluster setup is shown in fig. 6.

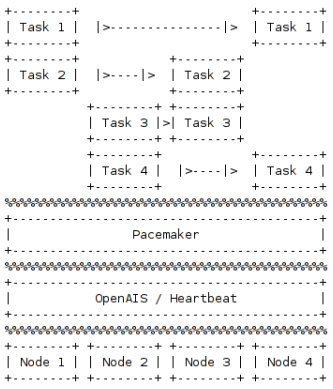

Figure 6. Pacemaker Active/Pasive HA cluster example

VII. Conclusion

Using a variety of Open Source tools it is possible to develop and maintain fully functional HA cluster that meets the requirements of business. Unfortunately this is not a simple task, at least not when developing a cluster with non standard configurations. If carefully deployed, Open Source HA cluster will have equal, if not better performance of its commercial alternatives on the same hardware platform. In the long term, time invested in research, development and testing custom cluster deployments has more benefits than purchasing expensive of the shelf cluster solutions.

References

[1] *http://www.drbd.org*
[2] *http://iscsitarget.sourceforge.net*
[3] *http://www.open-iscsi.org*
[4] *http://www.postgresql.org*
[5] *http://www.mysql.com*
[6] *http://www.xen.org*
[7] *http://www.linux-kvm.org*
[8] *http://wiki.openvz.org*
[9] *http://www.linux-ha.org*
[10] *http://clusterlabs.org*