Acta Polytechnica CTU Proceedings **1**(1): 278–282, 2014 doi: 10.14311/APP.2014.01.0278

HAVmS: Highly Available Virtual Machine Computer System Fault Tolerant with Automatic Failback and Close to Zero Downtime

Memmo Federici[1], Carlo Gaibisso[2], Bruno L. Martino[3]

[1] *Istituto di Astrofisica e Planetologia Spaziali, INAF IAPS Via fosso del Cavaliere 100, 00133 Roma, Italy*

[2] *Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", IASI-CNR Viale Manzoni 30, 00185 Roma , Italy*

[3] *Associated INAF IAPS*

Corresponding author: memmo.federici@iaps.inaf.it

Abstract

In scientific computing, systems often manage computations that require continuous acquisition of of satellite data and the management of large databases, as well as the execution of analysis software and simulation models (e.g. Monte Carlo or molecular dynamics cell simulations) which may require several weeks of continuous run. These systems, consequently, should ensure the continuity of operation even in case of serious faults. HAVmS (High Availability Virtual machine System) is a highly available, "fault tolerant" system with zero downtime in case of fault. It is based on the use of Virtual Machines and implemented by two servers with similar characteristics. HAVmS, thanks to the developed software solutions, is unique in its kind since it automatically failbacks once faults have been fixed. The system has been designed to be used both with professional or inexpensive hardware and supports the simultaneous execution of multiple services such as: web, mail, computing and administrative services, uninterrupted computing, data base management. Finally the system is cost effective adopting exclusively open source solutions, is easily manageable and for general use.

Keywords: HAVmS - high availability - fault tolerant - open source - multitask.

1 Introduction

HAVmS effectively solves the problems related to the "robustness" of computer systems wholly embracing the concept of high availability. A system in high availability, HA in what follows, must ensure the continuity over time of the provided services, which, in case of fault, must be restored in the shortest possible time. HAVmS, through an accurate design of the Hw and Sw, significantly reduces the faults and their negative effects on the provide services.

HAVmS has been designed mainly keeping in mind the following requirements: cost effectiveness, ease of management and, above all, the ability to automatically implement restoring strategies of the provided services without any interruption.

All the above requirements have been met by an accurate choice of the available Open Source solutions meeting our targets and their integration with the Sw specifically conceived by the authors. In particular, the last of them makes the system unique in its kind and competitive with analogous commercial solutions.

HAVmS is made by two servers, one active and the other dormant, but from the point of view of users, as well as from that of applications, the system is seen as a single server.

The fault tolerance of the system is ensured by a continuous synchronization of the two servers. This synchronization keep the data and the states of all the virtual machines, VMs in what follows, perfectly aligned.

Every service runs on a different VM each of which has its own RAM, storage and some computing resources. Each VM is independent of the others. When a fault occurs, the sleeper server is awakened and automatically takes the place of the broken server. Once the fault is fixed, it will be enough to reconnect the repaired server and automatically the failback procedure restores the proper functioning of the system including its HA capabilities.

1.1 The context

HAVmS is a general purpose system, this is one of its strengths. Among the potential areas in which the system proved its effectiveness, there is the acquisition and processing of data from space missions.

The main abilities required to HAVmS by this particular context are:

- managing of processes that require uninterrupted data analysis and data acquisition;

- storing the results of the analysis for long periods of time;
- making these results available to the scientific community,

in an effective, continuos and reliable way.

In fact, scientists in this area are interested in performing their analysis and tests on spatial observations as soon and on as much data as possible.

Furthermore, missions such as INTEGRAL (International Gamma-Ray Astrophysics Laboratory) Winkler, et al. (2003), have to maintain for a long period of time the results of the scientific analysis performed on the whole data set and usually released, as surveys, about every year.

Finally a further important requirement that has been met is making the management of the whole system, as far as possible, easy and automatic, thus minimizing the costs related to human resources devoted to the management itself.

1.2 The alternatives

In designing, HAVmS we deeply analyze and consider as points of reference the more widespread alternatives currently implementing the concept of HA.

All these alternatives, in general, suffers from the problem of the adoption of often expensive Sw and Hw architectures.

More in details these are, from our point of view and with respect to the particular operation context we are considering, the main drawbacks of the considered solutions:

- Windows Server Failover Clustering (WSFC):

Requires the purchase of a quite expensive Sw license (when this paper is written, about 900 Euros per processor). All servers in the cluster must be absolutely identical. HA is guaranteed by a mirroring system, replicating the status of the severs every 5-10 minutes: if there is a fault during a replica, the replica itself may fail and the running jobs are aborted with a high probability.

- VMware vSphere:

Requires the purchase of a quite expensive enterprise license (when this paper is written, about 4000 $). Requires professional Hw and an additional server to handle the nodes in the system. The setup is not simple.

- Red Hat Cluster Manager (Red Hat Enterprise Linux Server):

Requires the purchase of an annual license (when this paper is written, about 500 $). In case of fault the system must be restarted, as a consequence all runs in progress are stopped. Does not support an automatic failback mechanism.

2 Our Solution

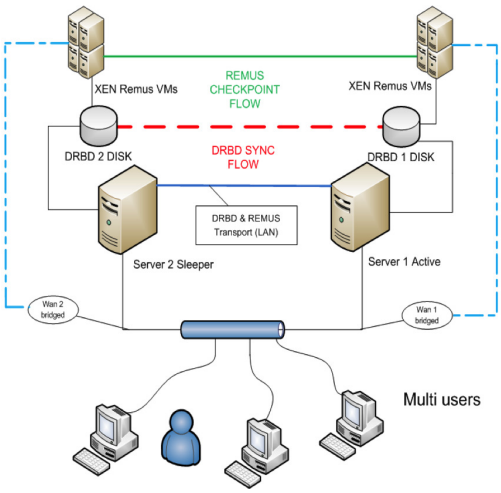

Figure 1: HAVmS block diagram

Fig.1 shows the general architecture of HAVmS. Continuity of service is provided through an automatic Failover (activates the dormant server in case of fault of the active one) and Failback (restores the initial system configuration and operation) mechanisms.

The software components implementing these mechanisms are:

- XEN VMs manager, a manager of virtual machines;
- DRBD (protocol D), a synchronizer of block devices;
- Remus, provides transparent, comprehensive high availability to ordinary VMs running on XEN;
- some components, designed and developed by the authors especially conceived, among the others, to implement the automatic failback and system startup.

The system is based on the services provided by Xen VMs manager. Each VM has its own IP address by which it is accessible from the outside. As previously stated, one server is active, the other one is dormant. The two servers are connected through a LAN at 1 Gb/sec. Remus (in green) and DRBD (in red) seamlessly (once every 40 msec) replicates the state of the active server on the dormant one. In particular, Remus replicates the states of VMs and simultaneously sends a trigger to DRBD, which, in turns, synchronizes

the block devices (hard disks) of the two servers. As a consequence the two servers are aligned once every 40 msec. Users connect to the VMs through their unique IP address. VMs makes it possible to install different operating systems, on which different softwares and services can be run, such as: Matlab, IDL and Web servers, compilers and so on.

3 Servers Software and Hardware Architecture

Both servers have the same software architecture, already introduced in the previous section.

The whole architecture leans on Linux Ubuntu Server 10.04 64 (with a kernel customized to support XEN).

In what follows the single components of the architecture are briefly described.

3.1 Xen

Xen hypervisor is a virtualization platform licensed under the GPL developed at the Computer Laboratory of the University of Cambridge. Xen is included in all major Linux distributions and increasingly adopted by commercial solutions. One of the its most interesting feature is the ability to effectively control the requests of access to physical resources coming from VMs through a paravirtualization mechanism. This mechanism guarantees a minimum decay of performance due to virtualization, since requests of access coming from VMs are mainly executed on the physical computing resources.

Paravirtualization requires a customized version of the Ubuntu kernel.

3.2 DRBD

DRBD (Distributed Replicated Block Device) is a distributed storage system for the GNU/Linux platform, usually adopted by HA clusters. DRBD is responsible for the synchronization of data between the servers: one of them is identified as primary, the other one as secondary. When the primary server fails, a management process promotes the secondary one to the role of primary. When the fault is fixed, the system may reestablish the roles initially assigned to the servers, after a resynchronization of the data storage devices. This synchronization is particularly effective since only those blocks that were changed during the outage are resynchronized.

3.3 Remus

Remus is part of XEN and implements the HA concept by replicating on the dormant server the state of all the VMs machine running on the active server. This snapshot occurs once every 40 msec (this value can be changed at setup time); at the same time Remus sends a trigger to DRBD for the synchronization of the hard disks. In this way the active server is constantly aligned, both for what concerns data and computational aspects, with the dormant server.

In case of fault of the active server, the dormant one is awakened and becomes immediately active, thus avoiding any interruption in the provisioning of services. Moreover, the TCP/IP protocol guarantees the correct transmission of data packets, even when the connectivity is temporary interrupted.

The architecture required to effectively support our HA solution, does not expect the use of professional HW. In fact, as will be evident from the following description of our experiences in the field, an entry level solution with the following characteristics turns out to be absolutely appropriate to the achievement of our goals:

- I7 8 cores Intel Processor
- 8/16 Gb Ram DDR3
- 2 Tb Hard Disk
- 2 network interfaces

3.4 Failover

The typical faults of a operating environment such as the one here considered here, can be attributed mainly to two categories:
lack of connection, malfunctioning of some VMs.
In case of a fault, the failover process is automatically triggered, which:

1. stops the DRBD synchronization between servers;
2. if running, stops the execution of Remus on the active server;
3. awakens the dormant server. The execution of VMs is consequently resumed from the last committed checkpoint (at most 40 msec before).

3.5 Failback

Unfortunately, Remus does not support any kind of automatic failback.

The Sw developed for HAVmS compensates for this deficiency and gives it its unicity.

This process, which is automatically triggered and implemented, restores the initial configuration and operational capabilities of the system once the fault has been fixed, included its HA functionality.

4 Practical Applications

The development of HAVmS was determined by some practical needs. Below we briefly describe the two applications that currently rely on the services offered by the system.

4.1 The HA data storage system of INTEGRAL

The Laboratory of Distributed Computing at IAPS (INAF) is in charge of AVES Federici et al. (2012), the cluster devoted to the analysis of data collected by INTEGRAL. AVES is connected to Data Storage Subsystem (DSS) a dedicated storage system Martino and Federici (2011) adopting HAVmS. DSS automatically downloads from INTEGRAL Science Data Center (ISDC) Courvoisier et al. (2003) the data collected by the satellite, backups this data and makes them available to AVES (16 TB to rise). As a consequence of a fault of DSS the download is interrupted and AVES as no longer access to the shared data. This might have severe consequences on the ongoing activities since, in turn, may cause a data misalignment and a crash of the running analysis. HAVmS avoid these risks and their unwanted consequences.

4.2 Continuity of service at IASI

HAVmS guarantees the continuity of the service provided by IASI IT division, among them: attendance control, centralized computing, storage and backup, printing services management. A different VM is allocated to each service.

5 Further Potential Applications

In the following we describe two potential fields of application of HAVmS we are investigating. These fields differ substantially from one another. Each of them is representative of a significant and large family of applications.

5.1 High End solution

Large structures, such as hospitals or national administrative offices should provide a relevant number of services that are usually hosted on individual servers. These servers, to ensure an acceptable level of reliability, should be equipped by systems for the backup of the status of the provided services and the data they deal with.

This is therefore a privileged context within which to exploit the characteristics of HAVmS. A quick market survey has shown that our solution can meet the above mentioned reliability requirements with not negligible costs but still low if compared with those of similar commercial solutions. In fact, one of the adoptable Hw solutions could be the following: Motherboard Supermicro Xeon MP series X9Qxxx, equipped with 4-socket Xeon processors (32 cores), 120 GB of RAM, 4 ethernet interfaces etc.

This configuration may provide up to 30 services, each with its own VMs and IP address, with a cost of approximately 5000 Euros per server.

5.2 A little gem

In this section we describe a solution adopting HAVmS that fundamentally, both in terms of computing power and cost, differs from the High End solution, which is a good example of the great versatility of our HA solution.

This solution has been designed to deal with the monitoring of atmospherical and geological events by the use of sensors in hard to reach sites.

The resulting system, which has a cost of about 300$ when this paper is written, is composed by three Raspberry PI collecting analog and digital signals from sensors and transferring them by a radio link to a remote control station. Thanks to the low power consumption, the system can be powered by a small photovoltaic panels. Two of the three Raspberry's are allocated to HAVmS; the third one, which is initially turned off, is switched on in case of failure of any one of the first two. This is obtained by a Hw signal sent through LAN (wakeup on LAN). The awakened Raspberry takes the place of the broken one thus guaranteeing the continuity of service, included those offered by HAVmS.

This solution substantially limits the number of maintenance interventions, thereby reducing the relative costs.

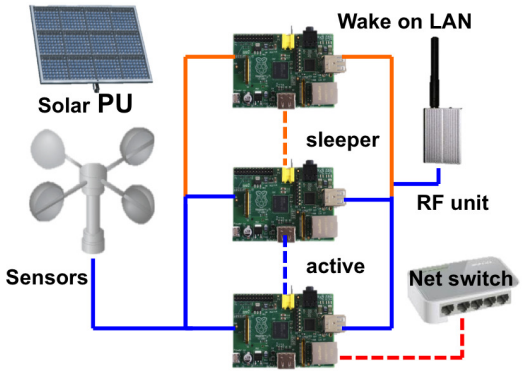

Figure 2: The Raspberry PI three node HAVmS

6 Further Work

We plan to improve the HA capabilities of HAVmS by adding to the current configuration a third server in

Wake on LAN. In case of fault this third server is automatically awakened by an Hw signal sent through the LAN and takes the place of the broken one. This automatic mechanism should not require any intervention by the system manager. With this solution, the system maintains its characteristics of HA also in case of fault of one server.

To improve the effectiveness of HAVmS in relation to the use of network resources, multiple network connections can be combined in parallel (bonding) to increase throughput beyond what a single connection could sustain, and to provide redundancy in case one of the links fails.

7 Conclusions and Discussion

HAVmS is a Highly Available Fault Tolerant general purpose, recyclable system based on the use of VMs, assuring continuity of operation and no interruption in services providing in case of fault.

HAVmS is cost effective since it only adopts open source solutions.

It is also extremely versatile, providing all operating systems supported by XEN. Paravirtualization dramatically reduces the negative impact of virtualization on the performance of the whole system.

The system supports automatic failover and failback within times close to zero. Automatic failback is an exclusive feature of HAVmS.

Acknowledgement

The authors thanks to: Giuliano Sabatino, Fabio Guglietta.

References

[1] Winkler, C., et al.: The INTEGRAL mission. Astron. Astrophys. 411, L1L6 (2003)

[2] M. Federici, et al. 2012, AVES: A high performance computer cluster array. Exp Astron (2012) 34:105121

[3] B.L. Martino and M. Federici "An high availability data storage subsystem for the INTEGRAL data analysis", Mem. S.A.It. Vol. 83, 377 2011

[4] Courvoisier, T.J.-L., et al.: The INTEGRAL science data centre (ISDC) for the INTEGRAL satellite scientific data analysis. Astron. Astrophys. 411, L53L57 (2003) doi:10.1051/0004-6361:20031172

DISCUSSION

BEALL JAMES's Question: Do you plan to use the Raspberry PI machine for computation?

MEMMO FEDERICI's Answer: Not in the traditional sense because RB is equipped with small amount of RAM (only 256 Mb) and little computing power. There are some applications that see this Hw used in small clusters with low consumption. We plan to use RB to make some services such as Web servers and print servers. This small device is very versatile and lends itself also to the process control.