A real time, Linux Container system to monitor transient events with INTEGRAL IBIS telescope

Bruno Luigi Martino[*] [1], **Ugo Zannoni**[2], **Giorgio Patria**[3]
[1] *CNR-IASI: Institute for Systems Analysis and Computer Science*
Via dei Taurini 19, 00185 Roma, Italy
bruno.martino@iasi.cnr.it
[3] *IAPS/INAF: Institute for Space Astrophysics and Planetology*
Via Fosso del Cavaliere 100, 00133 Roma, Italy
ugo.zannoni@iaps.inaf.it
[4] *ITIS G. Galilei*
Via Conte Verde, 51, 00185 Roma, Italy
giorgio.patria@gmail.com

This paper describes the IT infrastructure in operation at the Rome IAPS/INAF designed to ensure the continuity of service of a monitoring system actually in development phase devoted to the search for the electromagnetic counterpart of gravitational events. Thanks to a collaboration agreement, in case of detection of Gravitational Waves by LIGO / Virgo experiments, a software application is activated to search for interesting electromagnetic events in the data archive of the INTEGRAL SPI/IBIS satellite instruments. That application must be able to react to the trigger even in case of hardware or networking problems of the local IT system.

XII Multifrequency Behaviour of High Energy Cosmic Sources Workshop
12-17 June
Palermo, Italy

[*]Speaker.

1. INTEGRAL and GW

Following the official announcement of the first detection of a gravitational wave by the LIGO Interferometer (Laser Interferometer Gravitational-Wave Observatory), a historic result obtained by the LIGO/Virgo collaboration, a new era for astrophysical research opens up.

A great result has been achieved: the transition from the discovery phase of gravitational waves to the birth of gravitational astronomy. Until August 1, 2017, the Virgo experiment had collaborated only in the analysis of the data collected by LIGO, as it was waiting for the completion of some technological updates of the system, which is now in the Advanced Virgo configuration. Starting from this date it has become part of the network of 2 advanced LIGO detectors (participating in the observation campaign called O2).

Having three interferometers instead of two (2 in LIGO, 1 in Virgo) allows a better localization of the source of the gravitational waves, as well as the measurement of their other parameters, such as the polarization state. This allows to have more precise information in order to be able to direct the tools of other observatories to the region of the sky involved and to identify the possible electromagnetic counterpart of a gravitational event, improving the accuracy of its localization.

From now it becomes crucial to identify and characterize what produces gravitational waves, investigating in each band of the electromagnetic spectrum, from radio waves up to gamma rays. Thanks to a specific agreement between the research groups of the LIGO and Virgo interferometers, when a possible gravitational signal is detected, the IAPS researchers are warned and obtain information on the estimate of the position in the sky from which the gravitational wave originates.

A subset of these events could be expected to be associated with a GRB (the information remains private until it is published by the LVC collaboration). The imaging instruments on-board INTEGRAL(IBIS/ISGRI, IBIS/PICsIT, SPI, and the two JEM-X modules) [1] have been exploited to attempt the detection of electromagnetic emissions associated with Gravitational Wave events over 3 decades in energy (from 3 keV to 8 MeV). IBIS and SPI instruments are equipped with active shields to reduce the cosmic-ray induced background. The event count-rate provided by these shields can be used to detect transient γ-ray phenomena over the full sky, making the Anti Coincidence Shield (ACS) of SPI and the Veto system of IBIS detectors with a competitive sensitivity for such purpose.

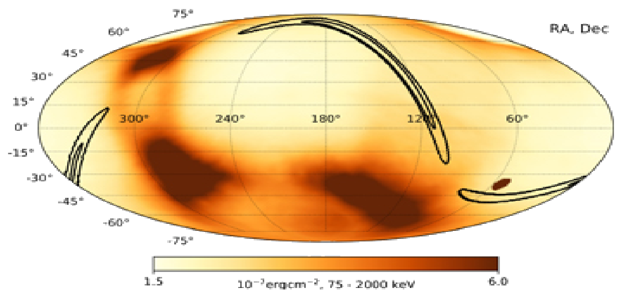


Figure 1: INTEGRAL 3 sigma upper limit during GW170104 [2]

At IAPS Institute is in develop state an application, as part of the " Searching for gamma counterparts of gravitational events' project in INTEGRAL satellite data, and creating an archive/

database for cross-calibration of high-energy instruments" (J. Rodi), with the goal of searching for coincident events with LIGO/Virgo triggers inside INTEGRAL data archive (non consolidated data too).

This is an application that must work without interruptions that can withstand unforeseen events and/or malfunctions, making it an ideal candidate to be managed by a high-availability computing system.

2. High Availability clusters

In order to implement highly reliable computing systems it is necessary to adopt redundancy mechanisms. As a first approximation, the duplication of some hardware components allows to obtain a good behavior of critical computing environment in case of unexpected destructive events. Among the replicable subsystems needed to build up a computer system we can highlight:

- power supply
- network interface
- disk storage
- ECC based Ram

When it is not possible or not economically convenient to install redundant hardware components, the primary objective becomes that of making the recovery time of the service, in the presence of malfunctions, as short as possible. Some possible solutions to the problem are:

- failover/failback strategies
- H24 management
- spare parts availability

A high availability cluster is a group of hosts that act like a single system and provide continuous uptime. The choice of having more physical machines devoted to the management of one or more services is not always the best solution because a virtualization environments make it easier to implement and is able to remove the hardware dependencies.

A virtual machine [3] is a completely virtualized environment that only abstracts the physical hardware; it works with its own BIOS, virtualized network adapters, disk storage, CPU and so on.

While a virtual machine abstracts the hardware at the application level, abstraction in a container takes place by exploiting the characteristics of the operating system. Each type of container technology has a defined and stated purpose that bound its scope. Each VM runs not just a full copy of an operating system, but a virtual copy of all the hardware that the operating system needs to run. This is one of the major reasons why containers are often used for running specific applications: less CPU and memory usage. VMs take up a lot of system resources, then our architecture is built on a Linux Containers architecture [4] (LXC).

All the containers in an HA cluster must have access to the same shared storage because this is the only way to make possible a consistent context switch. When an host goes down cause an hardware or software failure it is possible to migrate the relative service from the original host to a backup one or, at least, switch on a new server accessing of same applications data.

Containers and virtual machines have very similar objectives: the isolation of an application and its dependencies to obtain an autonomous unit able to work anywhere.

The main difference between them is the architectural approach; a container:

- uses the host kernel
- can't boot a different OS
- can't load it's own modules
- it's a bunch of processes visible on the host machine (VMs are opaque)

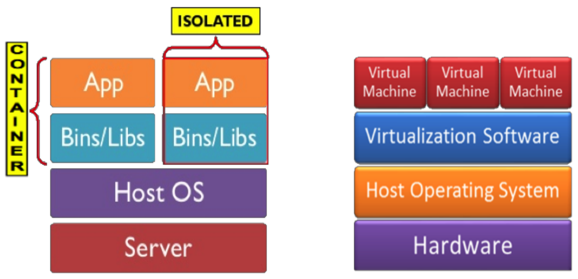


Figure 2: VM vs containers

Container orchestration is the automated arrangement, coordination and management of containers in their clusters. An orchestrator manage different aspects of container life-cycles, such as placement and initial deployment, scaling and replication, and so on. Among the most popular systems can be mentioned:

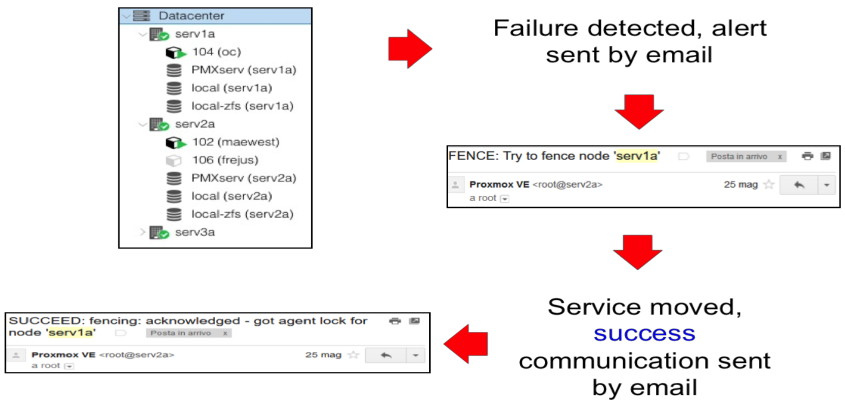
- Proxmox Virtual Environment is a complete server virtualization management solution, based on KVM and LXC container virtualization
- Rancher is trying to take what a container OS should be, everything in RancherOS is a Docker container
- Kubernetes , developed by Google, simplifies the containerized applications management in a clustered environment)

Proxmox VE is open source based on the Debian distribution and can be installed as a standalone system; alternatively, advanced users can install it on top of an existing Linux system (detailed knowledge about Proxmox VE is necessary). A Proxmox container can be managed throughi VE WEB interface by connecting an internet browser to 8006 port. In Proxmox VE new containers can be created using a set of templates; by default they aren't available after a new installation.

To let system use them, the user must issue the update command (pveam update). Available templates list can be retrieved using the WEB console. A new container creation process starts running the Create CT wizard; after the selection on CT name, ID number and password a template can be choosen. Usually, for performance reasons, the root image disk is stored in a local disk but an HA system must use a shared file system to allow the machine state migration.

3. Building an HA system

We can define the availability of a service during a certain time interval as the ratio between the total time a service can be used and the duration of the interval. It is normally expressed as a percentage of uptime in a given year. Virtualization environments like Proxmox VE make it much easier to setup High Availability systems [5] because they remove the “hardware” dependency. They also support to setup and use redundant storage and network devices. So if one host fail, simply start those services on another host within the cluster. Proxmox VE provides a software stack called HA-manager, which can do that automatically, able to detect errors and handle failovers situations.



FENCE: Try to fence node 'serv1a'
Proxmox VE <root@serv2a> 25 mag
a root

SUCCEED: fencing: acknowledged - got agent lock for node 'serv1a'
Proxmox VE <root@serv2a> 25 mag
a root

Figure 3: Fencing sequence

To setup a clustered HA system we must at least reach the following requirements:

- at least three cluster nodes (to reach a reliable quorum)
- shared storage for VMs and containers
- hardware redundancy (everywhere)
- use reliable “server” components

The quorum configuration in a failover cluster determines the number of failures that the cluster can sustain. To assemble a complete HA system, the required containers must be assigned to the

Proxmox HA manager. The cluster configuration database handle the “quorum”; it knows which node of the cluster are actives at any given time. Let’s try to imagine a system without quorum mechanism, e.g. a two- nodes cluster. If there is a network problem, the two nodes can’t communicate; without quorum, both nodes can operate independently (even in case of failure) and take disks ownership. This situation is called Split-Brain. This case must never happen because a contemporary execution means a concurrent access on a shared storage and a potential corruption of the critical application data. In a three-node cluster the majority of vote is 2 votes but because there is three nodes, you can lose a node and the cluster keep working. For this reason we have chosen to implement a 3-component monitoring system

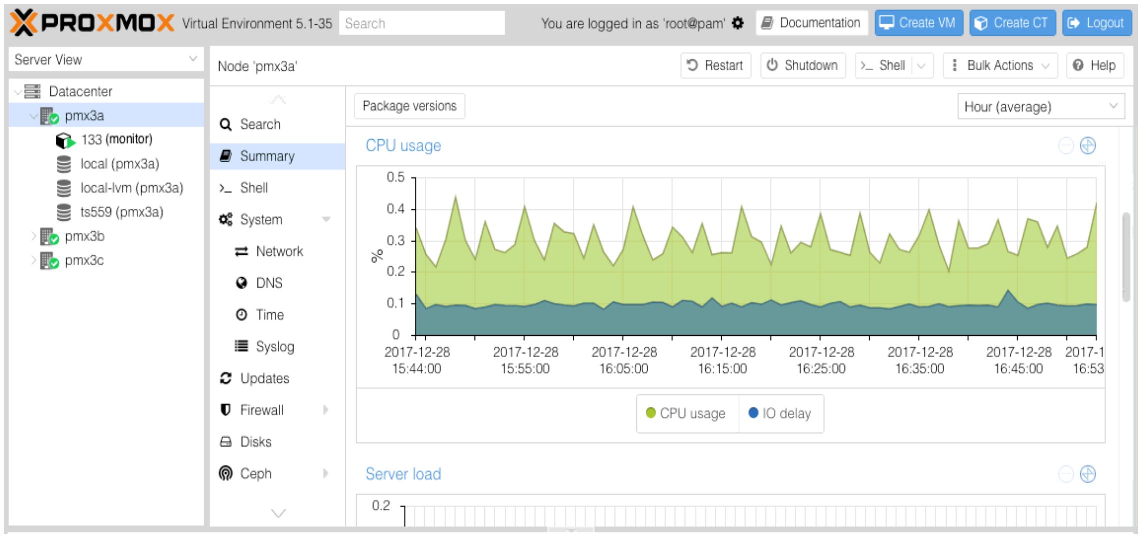


Figure 4: WEB console status summary

4. Conclusions

The adopted solution provides a high availability system with a short development time. Thanks to this our monitoring system is able to work without interruptions and in a reliable way. The choice of Proxmox was however also guided considering its extremely fine granularity in assigning physical resources to containers. Another advantage of this architecture is focused in its completely automatic operation; no manual intervention is needed to manage malfunctions because failover is very effectively handled. Any hardware anomalies (for example the occurrence of disk I/O errors, errors in network communications or delay in response to system interrupts) are reported to the system administrator via an e-mail message.

The choice of the operating system to be installed in a new container is made simpler by a set of predefined templates, both for general purpose distributions (like Debian, CentOS or Ubuntu) or oriented to the management of specific services such as turnkey (light insulated containers in a safe way).

Through the WEB console it is possible to control the start, stop and migration of containers between the hosts composing the cluster as well as verify the activity status of the resources such as CPU, central memory, number of cores etc. etc. Even the resources assigned can be redefined at any time without problems.

5. Acknowledgement

Thank you very much to Franco Giovannelli, Pietro Ubertini and the LOC. A very special thanks to our friend Memmo Federici, who has never really left us. Thank you Memmo!

References

[1] C. Winkler et al., *The INTEGRAL mission*, *A&A 411* [2003]

[2] V. Savchenko et al., *INTEGRAL detection of the first prompt γ-ray signal coincident with the Gravitational Wave event EVENT GW170817*, *Astrophys.J. 848 no.2*, [2017]

[3] R. Gareth et al., *Evaluation of containers as a virtualisation alternative for HEP workloads*, *Journal of Physics: Conference Series 664* [2015]

[4] *LXC, linux conainers*, *URL http://linuxcontainers.org/.* [Accessed: 2017-06-02]

[5] A. Reuther et al., *Scheduler Technologies in Support of High Performances Data Analysis"*, *MIT* [2015]