

LXC and Dockers: migrating OSA software on Linux containers

Bruno Luigi Martino*¹, **Giorgio Patria**², **Francesco Reale**³, **Memmo Federici**⁴

¹ *CNR-IASI: Istituto di Analisi dei Sistemi ed Informatica*

Via dei Taurini 19, 00185 Roma, Italy

bruno.martino@iasi.cnr.it

² *ITIS G. Galilei*

Via Conte Verde, 51, 00185 Roma, Italy

giorgio.patria@gmail.com

³ *CNR-ISC: Istituto per i Sistemi Complessi*

Via Fosso del Cavaliere 100, 00133 Roma, Italy

francesco.reale@sic.rm.cnr.it

⁴ *IAPS-INAf: Istituto di Astrofisica e Planetologia Spaziali*

Via Fosso del Cavaliere 100, 00133 Roma, Italy

memmo.federici@iaps.inaf.it

This paper describes the behavior of a containerized linux Dockers system and the advantages arising from it's use. It shows how using this approach is possible to run in the Windows environment the package OSA for the analysis of the INTEGRAL satellite data (Off-line Scientific Analysis) normally used in Linux / MacOS environments.

Frontier Research in Astrophysics - II

23-28 May 2016

Mondello (Palermo), Italy

*Speaker.

1. The Integral satellite

Integral is an ESA mission in cooperation with Russia and the United States launched on 17 October 2002 from the Baikonur Cosmodrome in Kazakhstan aboard Proton; with a mass of 4,100 kg, Integral was one of the the heaviest ESA spacecraft launched at that time; it stands 5 meters high and has a diameter of 3.7 meters.

INTEGRAL [1] is the first space observatory that can simultaneously observe objects in gamma rays, X-rays and visible light. Its principal targets are violent explosions known as gamma-ray bursts, powerful phenomena such as supernova explosions, and regions in the Universe thought to contain black holes.

The INTEGRAL spacecraft carries four scientific instruments:

- The gamma-ray spectrometer SPI (foreground cylinder)
- The gamma-ray imager IBIS (big square in the back)
- The two X-ray monitors JEM-X (1 & 2, the two circles between IBIS and SPI)
- The optical monitoring camera OMC (small conical tube at the left of JEM-X)

After a comprehensive review of the current operational status and the likely scientific return from the mission, the SPC decided to extend the operation of INTEGRAL from 1 January 2017 to 31 December 2018.

2. Off-line Scientific Analysis software

The INTEGRAL data analysis is usually performed applying the standard s/w provided to ISDC by the single PI led Instrument Teams. This s/w is named: Offline Science Analysis (OSA). OSA is continuously maintained and updated at ISDC taking into account the instruments, spacecraft and orbit evolution. The latest released version is named OSA10, being the 10th revision from the satellite launch [2]. The binary OSA software packages were built on Mac OS X platforms and on the Scientific Linux 32 and 64 systems. OSA is broken down into the following components:

- Off-line Scientific Analysis Software (OSA SW)
- Instrument Characteristics (OSA IC)
- 'High-energy' Catalog (OSA CAT)
- Test Observations, for example to check the correct installation of the INTEGRAL Data Analysis System (OSA TESTDATA)

The analysis is achieved by four independent instrument specific analysis chains (scripts). With the help of graphical user interfaces it is possible to alter the analysis parameters to customize the analysis runs to personal needs.

3. Virtualization

There are many operating situations that make if not necessary definitely appropriate the use of virtualization techniques. Virtualization software abstracts the underlying hardware by creating an interface to virtual machines (VMs), which represent virtualized resources such as CPUs, physical memory, network connections, and block devices. To simplify the concept, it comes to software applications that create an environment able to simulate the hardware of different computer architectures. An operating system called “guest OS” can then be installed into this environment while the operating system installed on the physical computer is the “host OS”. There are several ways to implement virtualization. Two leading approaches are full virtualization and para-virtualization.

Paravirtualization is virtualization in which the guest operating system (the one being virtualized) is aware that it is a guest and accordingly has drivers that, instead of issuing hardware commands, simply issue commands directly to the host operating system. This also includes memory and thread management as well, which usually require unavailable privileged instructions in the processor.

Full Virtualization is virtualization in which the guest operating system is unaware that it is in a virtualized environment, and therefore hardware is virtualized by the host operating system so that the guest can issue commands to what it thinks is actual hardware, but really are just simulated hardware devices created by the host.

Additionally, virtualization can be enhanced with the help of dedicated system hardware.

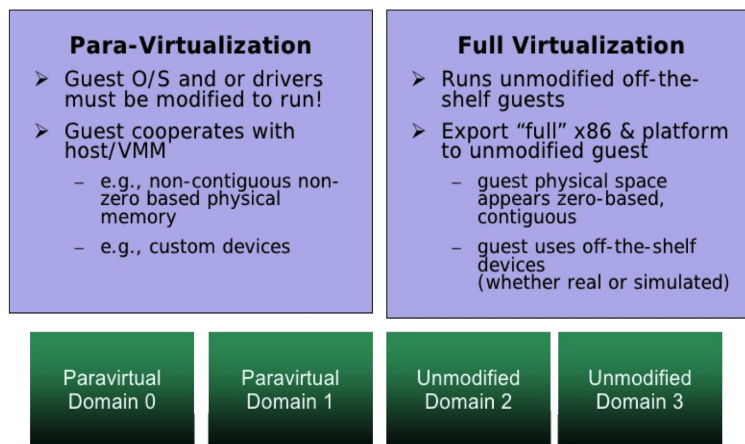


Figure 1: Para Virtualization VS Full Virtualization

4. Virtual machines

A virtual machine (VM) is a software computer that, like a physical computer, runs an operating system and applications. A virtual devices give the same functionality as physical hardware but provide benefits in terms of portability, manageability, and security. Specialized software, called a hypervisor, emulates the PC client or server’s CPU, memory, hard disk, network and other hardware resources completely, enabling virtual machines to share the resources. Virtual machines do

not require specialized, hypervisor-specific hardware. Virtualization does, however, require more bandwidth, storage and processing capacity than a traditional server or desktop if the physical hardware is going to host multiple running virtual machines. VMs can easily move, be copied and reassigned between host servers to optimize hardware resource utilization. Because VMs on a physical host can consume unequal resource quantities – one may hog the available physical storage, while another stores little.

5. Linux containers

Containers and VMs are similar in their goals: to isolate an application and its dependencies into a self-contained unit that can run anywhere [3]. The main difference between containers and VMs is in their architectural approach.

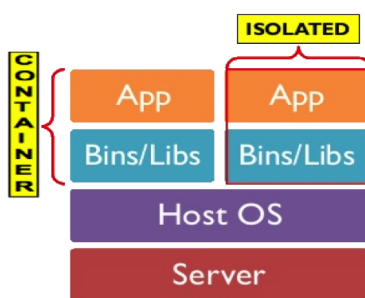


Figure 2: Linux container architecture

A container:

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

Linux container's building bricks are *Control Groups (cgroups)*:

- Each subsystem has hierarchy (Memory, CPU, Block I/O ...)
- Hierarchies are independent

and *namespaces*:

- Provides processes with their system own view
- Each process is in one namespace of each type (pid, net, mnt ...)

Cgroups limits “how much” you can use, Namespace limits “what you can see” (and use)

6. LXC and Dockers

As industry moves beyond the Virtual Machine consolidation paradigm, several types of containers have come to prominence. Two flavors in particular currently enjoy the share of deployments on the Linux operating system: LXC and Docker [4]. LXC provides a “normal” OS environment that supports all the features and capabilities available in the Linux environment. Behaves very much like a traditional VM and thus offers a lower barrier to entry and does not require changes to the application being deployed. Cloning and snapshots are supported and they provides full root access. An LXC container

- is a folder in `/var/lib/lxc`
- is described by a small config file
- requires sysadmins “elbow grease”

A Docker container

- packs an application and its dependencies
- contains everything it needs to run: code, runtime ...
- applications works regardless from the environment

Summarizing: LXC is used for complete Operating Systems packaging, Docker is used for applications deployment

7. Using dockers

Docker installation is embarrassingly easy, you just need to issue (inside an ubuntu linux environment) the command: `sudo apt-get -y install docker.io`

```
Adding group `docker' (GID 117) ...
Done.
Setting up ubuntu-fan (0.9.0) ...
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for systemd (229-4ubuntu4) ...
system@xsrv1:~$
```

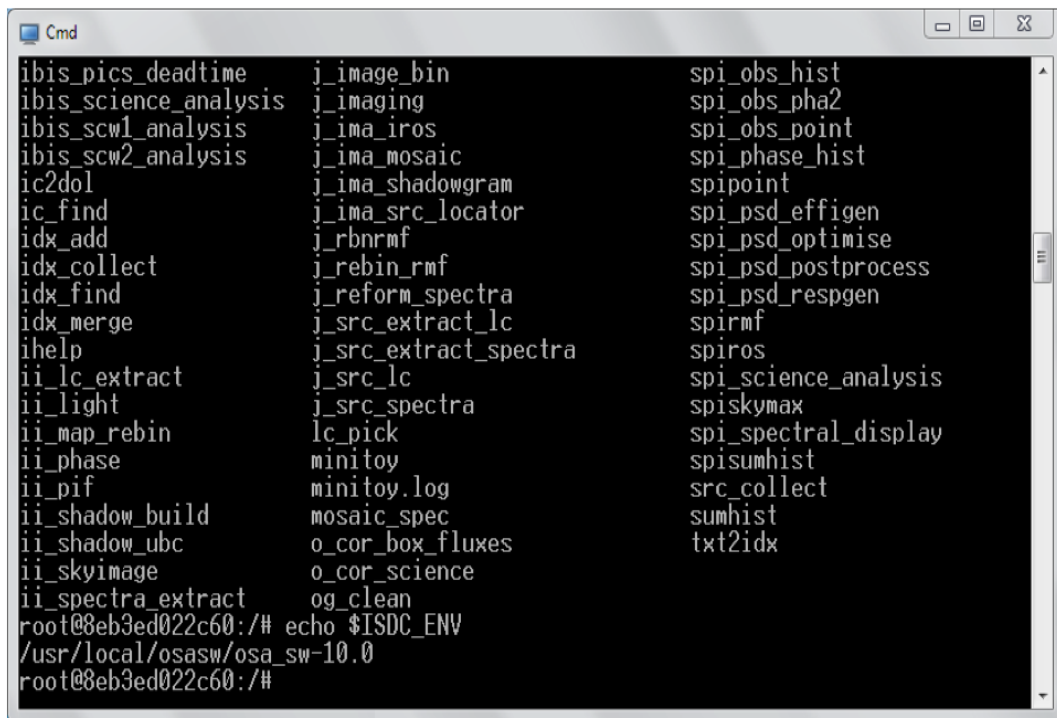
Figure 3: Example of a Docker installation

The setup of a new Docker container is made by downloading the required distribution (e.g. *ubuntu* from repository just entering the command: `sudo docker pull ubuntu`), after that the container is ready to be started: `docker run -i -t ubuntu /bin/bash`. Each time you will use `docker run` command you choose an image that already exist (e.g. the *ubuntu* image). If an image isn't already present on the host then it'll be downloaded from Docker Hub Registry. Docker can also build new images reading the assembling instructions from a text document (Dockerfile). Docker in Linux environment can run natively and to be able to run in Windows or MacOS environment requires

the use of *boot2docker*. Boot2docker is a lightweight Linux distribution based on Tiny Core Linux made specifically to run Docker containers. It runs completely from RAM, weighs 27MB and boots in 5s. At time of writing of this article Docker has launched a beta of Docker for Mac and Docker for Windows, two software packages that represent significant software architecture changes made to optimize Docker for an OS-native experience. While users could previously use Docker on Windows and Mac, the packages remove additional layers of dependencies and offer native, tightly integrated user experience. Docker can now be installed as a native Windows or native Mac app, launched and utilized from a system toolbar just like any other packaged application.

8. OSA and Dockers

The installation of OSA package is directly supported in unix-like platforms like Linux or MacOS, however, are not uncommon problems with different versions of libraries or compatibility issues. In addition, the recompilation of OSA from source code is a road often steep for the same reasons. We thought that the OSA installation Docker environment would be a great way to verify their actual effectiveness. An OSA version made in this way would have also allowed to be used also in machines inside Microsoft Windows environment.



```

Cmd
ibis_pics_deadtime      j_image_bin           spi_obs_hist
ibis_science_analysis j_imaging             spi_obs pha2
ibis_scw1_analysis     j_ima_iros           spi_obs_point
ibis_scw2_analysis     j_ima_mosaic         spi_phase_hist
ic2dol                 j_ima_shadowgram    spipoint
ic_find                j_ima_src_locator   spi_psd_effigen
idx_add                j_rbnrmf             spi_psd_optimise
idx_collect            j_rebin_rmf         spi_psd_postprocess
idx_find               j_reform_spectra    spi_psd_respgen
idx_merge              j_src_extract_lc     spirmf
ihelp                  j_src_extract_spectra spiros
ii_lc_extract          j_src_lc             spi_science_analysis
ii_light               j_src_spectra        spiskymax
ii_map_rebin           lc_pick              spi_spectral_display
ii_phase               minitoy              spisumhist
ii_pif                 minitoy.log          src_collect
ii_shadow_build        mosaic_spec          sumhist
ii_shadow_ubic         o_cor_box_fluxes    txt2idx
ii_skyimage            o_cor_science
ii_spectra_extract     og_clean
root@8eb3ed022c60:/# echo $ISDC_ENV
/usr/local/osasw/osa_sw-10.0
root@8eb3ed022c60:/#

```

Figure 4: OSA running in a CMD Windows

The OSA installation inside a container Docker did not give particular problems as carried out in a Linux OS selected from those directly supported by ISDC.

9. Conclusions

Using the Docker approach:

- the distribution of scientific software is made extremely simple. Once the working environment has been developed, it is possible to use the container virtually anywhere
- there is no performance loss because thanks to the absence of the virtualization layer the container's processes are the same than the host machine
- there is an effective independence from the host system
- the easiness of containers management make their start, stop and migration actions very useful in case of hardware failure or implementation of HA systems

Acknowledgments

Pietro Ubertini, Franco Giovannelli and LOC

References

- [1] Winkler et al., *The INTEGRAL mission Astronomy and Astrophysics*, 411, L1, [2003]
- [2] ISDC/OSA , *Introduction to the INTEGRAL Data Analysis*, URL <http://www.isdc.unige.ch/integral/download/osa/doc/10.2/> [Accessed: 2015-05-22]
- [3] LXC, *linux conainers*, URL <http://linuxcontainers.org/>. [Accessed: 2015-05-20]
- [4] E. Mazzoni et al. Docker experience at INFN-Pisa Grid Data Center, *Journal of Physics: Conference Series* 664 (2015) 022029. [2015]