A proto-Data Processing Center for LISA

Cécile Cavet, Antoine Petiteau, Maude Le Jeune, Eric Plagnol, Etienne Marin-Martholaz, Jean-Baptiste Bayle
Francois Arago Center, APC, Univ. Paris Diderot, CNRS/IN2P3, CEA/Irfu, Obs. de Paris, Sorbonne Paris Cité, 13 rue Watt, 75013, Paris, France
E-mail: ccavet@apc.in2p3.fr

Abstract. The LISA project preparation requires to study and define a new data analysis framework, capable of dealing with highly heterogeneous CPU needs and of exploiting the emergent information technologies. In this context, a prototype of the mission’s Data Processing Center (DPC) has been initiated. The DPC is designed to efficiently manage computing constraints and to offer a common infrastructure where the whole collaboration can contribute to development work. Several tools such as continuous integration (CI) have already been delivered to the collaboration and are presently used for simulations and performance studies. This article presents the progress made regarding this collaborative environment and discusses also the possible next steps towards an on-demand computing infrastructure. This activity is supported by CNES as part of the French contribution to LISA.

1. Data Processing Center
The LISA Data Processing Center (DPC), as described in [Amaro-Seoane et al.(2017)], is the entity that receives calibrated data (level 1 data) from the Science Operations Center (SOC) at ESA, processes them to identify gravitational wave sources and their parameters, and sends the results (level 2 and 3 data) back to the SOC. It is also tasked with identifying transient events to provide the outside community with alerts to search for electro-magnetic counterparts. The DPC will be delivered by the LISA Consortium under the responsibility of France [Amaro-Seoane et al.(2017), CNES Phase 0].

In order to begin the DPC activities and to provide the consortium with the tools to advance the LISA detector definition, the LISA proto-DPC [DPC] has been initiated in 2014. These tools will be used for the implementation of data analysis softwares, for the development of the LISA simulation package and to enhance the collaborative environment of these tasks.

1.1. Goal
The observations LISA will make will be the first of its kind (i.e. the simultaneous observation of a possibly large number of GW sources) and will explore a new way of observing the Universe. Therefore innovative data analysis (DA) techniques have to be implemented. During the mission operation, the DPC will have large fluctuations of CPU load due to the need of rapid processing of transient events and the regular reprocessing of the data with optimized DA techniques, new calibrations, consistency checks, etc. Moreover the source event rate is uncertain and can vary from a few events to a few tens thousand events per year. All these factors make the dimensioning of the pipeline and of the necessary resources a difficult challenge.
Some basic methods have been developed during past Mock LISA Data Challenges (MLDCs), demonstrating that the LISA DA challenge should be within reach. However details have to be studied using MLDCs of increasing complexity both for gravitational wave source and instrument modeling coming from the recent results of LISAPathFinder and from the implementation of future technological developments.

During the development of the DPC as well as during the operation phase, the pipelines will strongly evolve to integrate updated developments in a short loop cycle. For the operation phase, the processing of the data is expected to be done on a daily and weekly basis and should ensure non-regression on short timescales.

Because of all these complexities, new challenges have to be handle using innovative IT technologies such as virtualization and DevOps (see following sections). In addition, due to the very long term nature of these activities and the rapidly evolving IT solutions, the DPC has to be kept flexible and easily upgradeable until the end of the mission.

1.2. Why now?

There are a number of reasons pushing for an immediate start of the DPC:

- A framework to start collaborative work on DA and simulation is needed on the short term.
- An infrastructure to support the DA challenges has to be implemented soon.
- An infrastructure to support the end-to-end simulations that will be used to produce realistic data, evaluate mission performances and assess the industrial proposals is needed by 2018.
- A structure hosting the various softwares used during the development of LISA, in particular performance management tools, is also necessary in the near future.

The proto-DPC will be the framework that will support the LISA simulations and the next data analysis collaborative activities such as MLDCs.

2. Continuous integration

Continuous integration (CI) has become a standard development practice in the recent years, answering some of the needs that have emerged with Agile [Agile] and DevOps [DevOps] movements in the software development community.

Besides its trending methodological context, CI is now embodied into software systems which provide tools to help software development and collaborative work. In this section, we focus on the description of such CI environments, highlighting how they can help and impact the collaborative software development, in the context of the LISA DA pipeline construction.

2.1. Automatic tests

The main ingredient in CI is the development of a suite of non-regression tests, which can be automatically and systematically run by the CI system. Those tests can address multiple problems that are usually encountered in big software projects such as compatibility between versions, performances, validations, etc.

A schematic view of the CI cycle is given in Figure 1. The CI system is fed by code commits coming from a version control system (VCS) which hosts the source code and its version history. Commits automatically trigger the execution by the CI system of the suite of non-regression tests, starting with code compilation, building and installation on the CI environment. Test results are stored in the CI database, and exposed by the CI web interface. At any moment, developers can check those results, correct and improve the source code if needed, or retrieve a working version of the code (possibly associated with its working environment in a DevOps style project).
Figure 1. The CI system supports automatic code building and testing. By checking the test results using the web interface, developers can improve their codes. The integration in the production environment can be done either by cloning the CI environment or using software containers.

The automation of those tasks provides great benefits, especially for projects with a high number of interdependent sections of code that rapidly evolve, such as DA pipelines. Scientific developers are able to keep improving any part of the source code, even during operation phases, as the CI system guarantees the provision of a working version of the pipeline.

2.2. Towards good practices
The second main advantage of CI is more subtle. One has to realize that the CI system is by default without any intelligence: it is merely a robot that runs what has been implemented. CI relies on tests that the developer has to design and provide to the system. Those tests have to be sensitive to the particular (possibly evolving) objectives that the code must fulfill (precision, speed performance, lack of side effects, etc.): they have to be well-written and kept up-to-date. This is where the weakness and the force of the CI system resides: the system by itself strongly impacts the development methods adopted by the developer by suggesting (but not imposing) good practices such as a test-driven development (TDD).

Another kind of tests that can be use to feed the CI system is source code syntactic analysis. On this aspect, one can use the standard publicly available test rules (like Python PEP 8 standard coding rules) which enforce code readability.

As a first version of such a CI environment for the LISA DPC, some of the most popular software solutions at this time, namely Jenkins [Jenkins] for CI, and SonarQube [SonarQube] for code quality checking have been chosen.

3. Virtualization
Virtualization is a major new technique that has a big impact on the way hardware and configurations are managed.

3.1. Full virtualization and cloud computing
Full virtualization is a technology that allows to isolate an entire guest Operating System (OS) from the host OS using an hypervisor (Xen, KVM ..., see Fig. 2). The Infrastructure-as-a-Service (IaaS) cloud solutions, such as Amazon Web Services (AWS) and OpenStack, provide a stack of components including dynamical managers of hypervisors. Cloud computing is very
interesting for data analysis due to the fact that instantiation of infrastructures provide on-demand virtual resources: virtual machines with virtual CPU, memory and network; virtual but persistent disk volumes; object storage. In other words, one can arbitrary request hardware configurations instead of buying CPU units. Started by the AWS company in 2006, public IaaS cloud infrastructures have also emerged in the academic community. Researchers have now access to the European e-Infrastructures [EGI] and from a national point of view, to the French federation [FG-cloud] to which the Computing Center of IN2P3 (CC-IN2P3) takes part.

The dynamic provision of cloud’s virtual machines has resonated with the DevOps approach. Indeed, system administrators usually develop scripts to automatize complex workflows in order to facilitate service and software deployment. With cloud computing, this configuration automation step also opens a window on the usage of on-demand resources: with an automatic configuration, it is easier to deploy services and codes on virtual instances. Several configuration frameworks (SlipStream, Heat, Juju ...) are dedicated to the cloud while others (Ansible, Puppet, Chief ...) were developed for general platforms (bare metal or virtual machines).

These tools will be put in use for the LISA DPC in order to handle CPU fluctuation needs, and to pool hardware resources from all partners.

3.2. Containers

Recently, the re-discovery of Linux containers has given rise to an innovative way to encapsulate and share codes and services. Containers are based on a specific Linux process (namespaces) which allows to isolate one task execution from another (see Fig. 3). The emergence of several solutions (LXC, Docker, rkt, Singularity, etc.) has changed developer workflow on code prototyping and production phase. As for virtual machines in the cloud, an ecosystem of components is provided to handle specific properties of containers. Furthermore, hubs and catalogs of images have been created to offer a central place for sharing images between users. In a production environment, orchestration of containers is managed by a specific tool (Mesos/Chronos, Swarm, Kubernetes) which allows to combine cluster and cloud infrastructure in a hybrid way [Poncet et al.(2016)].

4. How to contribute?

Our philosophy is to provide the LISA community with a suited stack of tools, which should help the consortium simulation and analysis code development and execution. In order to do
so, we aim at providing such tools in the near future, to support the mission preparation phase and initialize an improvement *virtuous circle* within the LISA community.

We are looking forward to getting feedback from the LISA data scientists willing to start using these tools. The current entry point is the DPC website [DPC] which gathers all the information. In particular, it provides technical details on how to use code hosting and continuous integration platform. It will be updated with new information through the DPC development.

**Acknowledgments**
The proto-DPC activity is supported by the François Arago Centre (Paris) and CNES. CC acknowledge the financial support from the UnivEarthS Labex program of Sorbonne Paris Cité (ANR-10-LABX-0023 and ANR-11-IDEX-0005-02). JBB was supported by CNES.

**References**
[Amaro-Seoane et al.(2017)] Amaro-Seoane, P., Audley, H., Babak, S., et al. 2017, arXiv:1702.00786
[Agile] The Agile manifesto: 
http://www.hristov.com/andrey/fht-stuttgart/The_Agile_Manifesto_SDMagazine.pdf
[CNES Phase 0] CNES, “NGO Ground segment - Phase 0 report”, CNES Report, DCT/ME/EU - 2014.0015466, 2016
[DevOps] DevOps: http://radar.oreilly.com/2012/06/what-is-devops.html
[DockerHub] DockerHub: https://hub.docker.com/u/elisadpc
[DPC] Proto-DPC platform: http://elisadpc.in2p3.fr/home
[EGI] EGI Advanced Computing Services for Research: https://www.egi.eu/
[FG-cloud] FG-cloud: http://www.france-grilles.fr/services-catalogue/fg-cloud/
[Jenkins] Jenkins: https://jenkins.io/
[Poncet et al.(2016)] Poncet, M., Faure, T., Cavet, C., Petiteau, A., et al. 2016, BiDS016
http://esaconferencebureau.com/custom/16X05/bids/ALL/D2_1210_Poncet.pdf
[SonarQube] SonarQube: https://www.sonarqube.org/