ReproVIP Report #2.1.1
Application Deployment Strategies

Emmanuel Medernach[1], **Gaël Vila**[2], **Axel Bonnet**[2], **Sorina Camarasu-Pop**[2]

[1] Univ Strasbourg, CNRS, IPHC UMR 7178, Strasbourg, France
[2] INSA-Lyon, UJM-Saint Etienne, CNRS, Inserm,
CREATIS UMR 5220, U1294, Lyon, France

Abstract. *This paper reports achievements of the ReproVIP project, regarding the deployment of the targeted applications on the available infrastructures to ensure reproducibility. One publication has been produced in this framework and submited for evaluation at ACM REP'24*

Résumé. *Ce document rapporte des avancées du projet ReproVIP concernant le déploiement des applications ciblées dans le cadre du projet sur les infrastructures disponibles pour assurer la reproductibilité. Une publication a été réalisée dans ce cadre et soumise pour évaluation à ACM REP'24*

Contents

1. Introduction

1.1. Context

As mentioned in previous reports [1], a scientific result depends on (i) a methodological choice (*e.g.*, a parameter set), applied on (ii) a given pipeline (*e.g.*, a scientific application), executed within (iii) a computing environment (comprising both software dependencies & hardware architectures). Within this study we focus on the deployment of scientific applications as a means of improving **reproducibility at the environment** level.

We consider *reproducibility* as one's ability to get *identical outputs* when applying the same treatments to the same set of inputs. The causes for the lack of reproducibility on the computational environment level are mainly related to the library dependencies and their evolution over time, but also to numerical instability due to floating point arithmetic issues (rounding errors, hardware and compiler optimizations). For a deterministic application, there may be different sources of variability:

- the source code of the application itself
- versions of all its (build and runtime) software dependencies
- the compiler and compilation options, or the interpreter
- the hardware architecture
- parallelisation (when applicable), which is beyond the scope of this study.

Approaches to mitigate the extent of environment-introduced variability often rely on package managers or containers [2][1]. There exist different types of package managers, such as (i) OS-level (*e.g.*, yum or apt), (ii) user-level (*e.g.*, pip or conda), or (iii) functional user-level (*e.g.*, Nix or Guix). Containers allow developers to package and run an application and its dependencies — including all configuration files and dependent libraries — in a portable environment. The most popular container technologies are currently Docker and Singularity (renamed Apptainer).

The Virtual Imaging Platform (VIP) deploys and executes pipelines on the distributed and heterogeneous computing resources of the EGI e-infrastructure. Most of VIP computations use the legacy EGI high throughput computing (HTC) nodes available to the biomed Virtual Organization through the Dirac interware. HTC clusters are administrated independently and jobs have standard user rights, with no possibility of using OS-level package managers or Docker containers as such. Available alternatives are Singularity, udocker[2] and access to a central software distribution solution based on the CernVM File System (CVMFS)[3]. VIP jobs have to run seamlessly on all available computing resources, regardless of their architecture and configuration. Application deployment on the computing resources is thus a key element, for ensuring both successful execution and a certain level of reproducibility. Current VIP applications are deployed using either static built executables or udocker containers stored on CVMFS. The use of local clusters and Cloud resources, in addition to HTC, is also possible and increasing in VIP. Figure 1 summarizes the VIP ecosystem.

[1] See also `https://epcced.github.io/2020-12-08-Containers-Online`

[2] `https://github.com/indigo-dc/udocker`

[3] `https://cernvm.cern.ch/fs/`

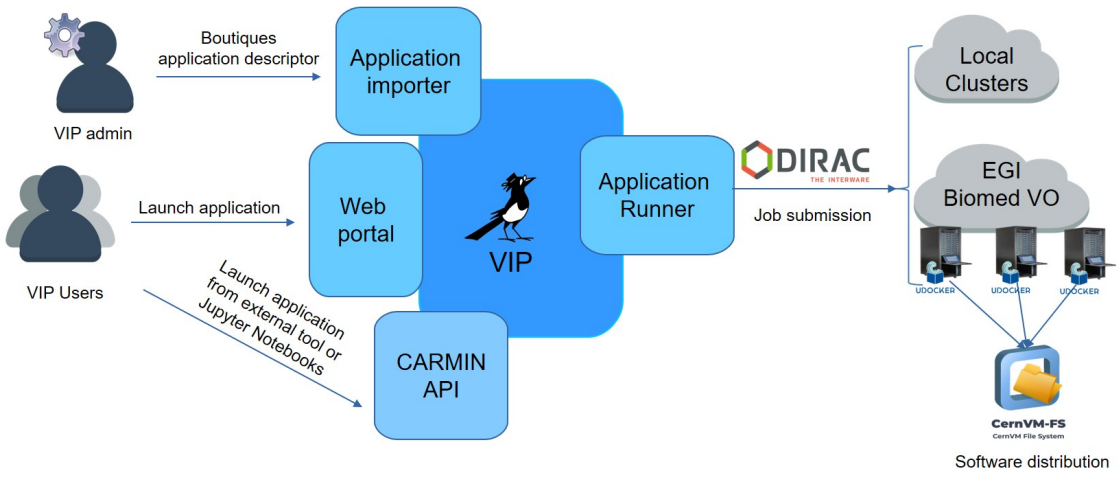

Figure 1. VIP ecosystem

1.2. Objectives

In this context, one of the aims of the ReproVIP project is to evaluate different application deployment technologies with respect to the:

- Reproducibility impact at the environment level
- Applicability to the targeted infrastructures and use-cases.

This deliverable gives the global overview of our study and then concentrates mostly on the second point. The reproducibility impact at the environment level has been published in [3].

2. Materials & Methods

2.1. Infrastructures

For VIP usage we target the EGI infrastructure, but since EGI is a production infrastructure, we used Grid'5000 [4] in our study. Grid'5000 is a large-scale testbed deployed in France for experiment-driven research in all areas of computer science. It provides access to a large amount of resources highly reconfigurable and controllable, which allowed us to adopt solutions available on EGI, such as CVMFS. The computation experiment on Grid'5000 is summarized in Figure 2.

2.2. Deployment strategies

Given the large adoption of containers by various platforms, such as VIP, Neurodesk[2], BIDS apps[5] or Boutiques[6], their choice among the studied technologies seemed straightforward. In addition to containers, we also considered Guix, which is a package and system manager allowing to build reproducible computational environments with the help of modules that accurately document the software building chain and its dependencies. Moreover, Guix packages could be exported as Docker containers or other types of archives.

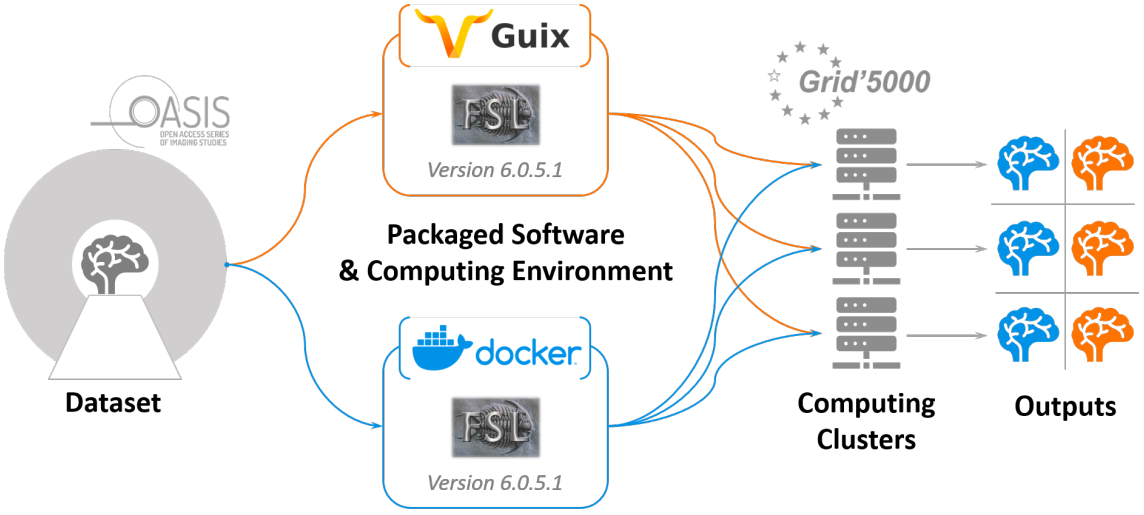

Figure 2. Summary of the Grid'5000 experiment

2.2.1. CVMFS

CVMFS provides a scalable software distribution service. CVMFS is implemented as a POSIX read-only file system in user space. Files and directories are hosted on standard web servers and mounted in the universal namespace /cvmfs. The STFC Scientific Computing Department at RAL maintains a CVMFS server for the EGI communities, among which biomed. CVMFS is currently the main software-distribution solution for the biomed community using the EGI HTC resources.

VIP has been using CVMFS in production for the last few years. We see it as an efficient and scalable solution to make software available on EGI computing nodes, in addition to other software packaging and distribution strategies, such as containerization or Guix packages. This is also the way CVMFS has been used in this study.

In addition to the CVMFS uploader maintained by RAL, we also installed a local CVMFS server in a virtual machine on the SCIGNE[4] infrastructure. This server was mainly used for jobs running on the Grid'5000 infrastructure, but we were also able to use it for EGI jobs. On Grid'5000 nodes, we installed and configured a CVMFS client to access files on our CVMFS server. For VIP jobs running on EGI, we configured them to use our local CVMFS server instead of the default one.

2.2.2. Docker

For the Docker deployment, we used the FSL Docker image vnmd/fsl_6.0.5.1 provided on Docker Hub. On Grid'5000 nodes we were able to to install and use Docker (package docker-desktop-4.20.1-amd64.deb), while on EGI HTC resources we use udocker. We cloned the udocker repository (version 1.3.1) on the CVMFS biomed uploader where we used it to pull the Docker image and to create a container stored in the biomed CVMFS folder. Jobs running on EGI HTC nodes cloned the same udocker repository and used

[4] https://scigne.fr/

udocker for executing the container.

2.2.3. Guix

We wrote FSL Guix modules to compile FSL and all its dependencies in a reproducible manner. In Guix, all packages are installed in separate directories. We adapted the compilation scripts to reference these paths. Package compilation with Guix was done using the guix pack command and using relocatable packages options.

Our Guix build sever was a virtual machine deployed on a local cloud infrastructure (SCIGNE). The packages were build on our Guix build server with gcc compiler version 9.5.0 and then deployed using our local CVMFS server.

FSL Guix modules are available at: gitlab.in2p3.fr/reprovip/reprovip-guix

2.3. Use-case

We chose to focus on the FSL FLIRT application, a software tool frequently used as a building block in neuroimaging analyses (and which had previously been deployed through VIP on EGI as an executable compiled on CentOS). For this study, FSL was executed on the OASIS-I dataset comprising 148 brain scans from 39 healthy subjects.

3. Reproducibility Experiments

Using the Grid'5000 infrastructure, we studied the effect of nine different CPU models using both Docker and Guix and we compared the resulting hardware variability to numerical variability measured with random rounding. Results showed that **hardware, software, and numerical variability led to perturbations of similar magnitudes** — although uncorrelated — suggesting that these three types of variability act as independent sources of numerical noise with similar magnitude. The effect of hardware perturbations on linear registration remained moderate, but might impact downstream analyses when linear registration is used as initialization step for other operations.

The study will be presented at the 2024 ACM Conference on Reproducibility and Replicability (ACM REP '24) . The preprint can be downloaded below:

https://hal.science/hal-04006152

4. Discussions and conclusions

We used the Grid'5000 infrastructure to study the variability in results obtained with a neuroimaging application packaged with Docker and Guix and executed on nine different CPU models. The setup was conceived as close as possible to the one used by VIP on EGI in production conditions.

Docker and Guix, the two packaging solutions used in our experiments, are known to mitigate software variability. From a computational bit-wise reproducibility point of view, experiments conducted in this study show that the two packaging solutions lead to similar conclusions: results are bit-wise reproducible when using the same packaged executable on equivalent micro-architectures. However, the Docker image is a black box providing little or no information on how the executable was built (both on the compilation process,

and on software dependency stack). In contrast, Guix enforces full transparency on both compiling and runtime environments, requesting for the description and availability of all dependencies. The Guix package is thus more complex to produce than a Docker image, but, once available, variations can be easily built by modifying compiler options or by using other versions of dependent packages. Moreover, Guix allows to save a manifest file, creating a profile which contains provenance metadata, and able to rebuild the exact same image at any future time.

Bit-wise reproducibility would require freezing all software, compiler and dependency versions, using no compiler optimizations and disabling dynamic architecture support. While this can be important in some cases, we can also question the pertinence of such an approach on a larger scale and the longer term. Computing environments cannot be frozen eternally and scientific results are subject to certain degrees of variability that need to be taken into account.

If we take into account the applicability of the studied solutions to the targeted infrastructure, we need to consider that Guix is currently not available on the EGI HTC nodes. Guix packages need therefore to be deployed on CVMFS using relocatable packages options. However, deploying all the Guix generated files on CVMFS may be challenging because of their large number[5]. For this reason, the current Guix-CVMFS based solution is available in VIP in test mode only. The adoption of Guix as package manager on top of the CVMFS software distribution solution could help solve these issues.

References

[1] Gaël Vila, Morgane Des Ligneris, Axel Bonnet, et al., "ReproVIP Report #1.1.1 Selected Metrics and Reproducibility Results," Tech. Rep., CREATIS Université Lyon 1, Feb. 2023.

[2] Angela I. Renton, Thanh Thuy Dao, David F. Abbott, et al., "Neurodesk: An accessible, flexible, and portable data analysis environment for reproducible neuroimaging," *bioRxiv*, 2022.

[3] Gaël Vila, Emmanuel Medernach, Inés Gonzalez, et al., "The Impact of Hardware Variability on Applications Packaged with Docker and Guix: a Case Study in Neuroimaging," Submitted at https://acm-rep.github.io/2024/, Feb. 2024.

[4] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, et al., "Adding virtualization capabilities to the Grid'5000 testbed," in *Cloud Computing and Services Science*, Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan, Eds., vol. 367 of *Communications in Computer and Information Science*, pp. 3–20. Springer International Publishing, 2013.

[5] Krzysztof J Gorgolewski, Fidel Alfaro-Almagro, Tibor Auer, et al., "Bids apps: Improving ease of use, accessibility, and reproducibility of neuroimaging data analysis methods," *PLoS computational biology*, vol. 13, no. 3, pp. e1005209, 2017.

[6] Tristan Glatard, Gregory Kiar, Tristan Aumentado-Armstrong, et al., "Boutiques: a flexible framework to integrate command-line applications in computing platforms," *GigaScience*, vol. 7, no. 5, pp. giy016, 2018.

[5]Large catalogs stress the CernVM-FS transport infrastructure and mai require splitting into nested catalogs