Fully containerized approach for the HPC cluster at FAIR

M. Al-Turany[1], D. Bertini[1], M. Dessalvi[1], S. Fleischer[1], R. Grosso[1], T. Kollegger[1,2], D. Kresan[1*], V. Penso[1] and C. Preuß[1]

[1] IT, GSI Helmholtz Center for Heavy Ion Research GmbH, Darmstadt, Germany
[2] Institute for Computer Science, Goethe-University, Frankfurt am Main, Germany

[*]E-mail: D.Kresan@gsi.de

Abstract. The scientific program of the future FAIR accelerator covers a broad spectrum of topics in modern nuclear and atomic physics. This diversity leads to a multitude of use cases and workflows for the analysis of experimental data and simulations. To meet the needs of such a diverse user group, a flexible and transparent data processing center is required to accommodate all FAIR experiments and users. In this work, we describe the operational approach for the computing cluster at GSI/FAIR that is characterized by an exceptionally minimal host system. This is achieved by installing and running all user applications in containers. The implementation of this approach on a production system hosting approximately 700 users, 80.000 CPUs and 400 GPUs demonstrates its feasibility and scalability. A transparent solution for interactive work in a containerized environment that addresses different levels of user experience is introduced. In addition, users have the opportunity to construct and submit their own containers. The paper will cover also how usage of Spack and CVMFS contributes to the overall efficiency and adaptability of the computing cluster at GSI/FAIR.

1. Introduction

The Facility for Antiproton and Ion Research (FAIR), currently under construction adjacent to the GSI Helmholtz Centre for Heavy Ion Research in Darmstadt, Germany, represents a significant advancement in the field of particle and nuclear physics [1]. FAIR's research agenda spans a wide array of topics within Atomic Physics, Nuclear Structure and Astrophysics, Heavy Ion Collisions, and Physics with Hadrons. By facilitating experiments at the cutting edge of these fields, FAIR will provide unique insights into the fundamental forces of nature and the structure of matter under extreme conditions.

To support the diverse experiments and the complex modern detectors required for their studies, FAIR is developing an on-site data processing center. The massive amounts of data generated by FAIR's detectors necessitate a robust computing infrastructure capable of handling high data rates and performing intricate data analysis with high efficiency. Given the diverse and extensive user community at FAIR, the computing infrastructure must be highly adaptable to various software tools and analysis workflows. This necessitates a scalable HPC cluster that can grow with the increasing demands of research while keeping a low maintenance overhead.

Maintenance has been a critical challenge with previous generations of clusters at GSI. The extensive and diverse user community required the installation of over 4,000 software packages

on the host system, making it virtually impossible to upgrade the cluster's operating system without causing disruptive changes for users. Drawing on this experience, we opted for a containerized cluster design, which will be discussed in this paper.

2. HPC Cluster for FAIR

2.1 Computing resources requirement [2]

Demand on the compute resources for an experiment is mainly driven by detector setup, beam taking conditions and physics of interaction. Compressed Baryonic Matter (CBM) experiment will operate at high beam intensity and interaction rate, which in combination with highly granular detector and trigger-less readout will deliver extremely high data rates on the order of 1 TB/s [3]. In order to be able to store data, events need to be reconstructed and pre-selected in online mode,

Table 1. Number of cores and amount of disk storage which are required by four scientific pillars of FAIR.

	NUSTAR	CBM	PANDA	APPA
Number of cores for offline analysis	9 000	45 000	68 000	11 000
Number of cores for online event reconstruction	7 000	45 000	34 000	-
Storage (TB)	34 250	103 000	60 680	7 037

In addition, around 400 GPUs will be provided

during data taking. In this way, data stream will be reduced to few tens GB/s. Hardware resources needed for such online farm can be taken from table 1.

The experiments will not have dedicated hardware but will share the cluster dynamically. Required amount of compute nodes will be reserved for the time of a corresponding run. The fact that not all experiments are carried out at the same time makes this approach more practicable and help saving computing costs and resources.

2.2 Fully containerized approach

In order to cope with diversity of software and hardware requirements in our user's community, we have decoupled user space from host space and introduced fully containerized approach.

According to this approach, Linux distribution on the host system is kept at minimum number of installed packages and all user applications are executed with Apptainer [4]. The host OS contains:

- Resource Management System (Slurm) [5]
- Container engine (Apptainer)
- HW drivers (InfiniBand, GPU)

System is built such, that it supports submitting of user-defined containers. This offers another degree of freedom in choosing of Linux flavour as well as software installed. It makes such a cluster highly scalable in multiple dimensions, with less maintenance effort.

2.3 Virtual Application Environment

Ready-to-be-used Apptainer image, which mounts externally maintained software stack at run-time, is called Virtual Application Environment (VAE). It is provided as a common solution for user application space. Users also have the opportunity to maintain and deploy their own container images, e.g. for specialized GPU applications.

Usage of Spack package manager for building and installation of the software stack for VAE enables support of following features:

- automatic handling of dependencies,
- multiple package versions,
- multiple micro-architectures, tailored to the cluster hardware,
- mixed toolchains.

CVMFS [6] is used for distribution to the worker nodes. Having its effective caching on the client side, let us operate the cluster in a very effective and stable way.

3. Advanced features

3.1 Login into container [7]

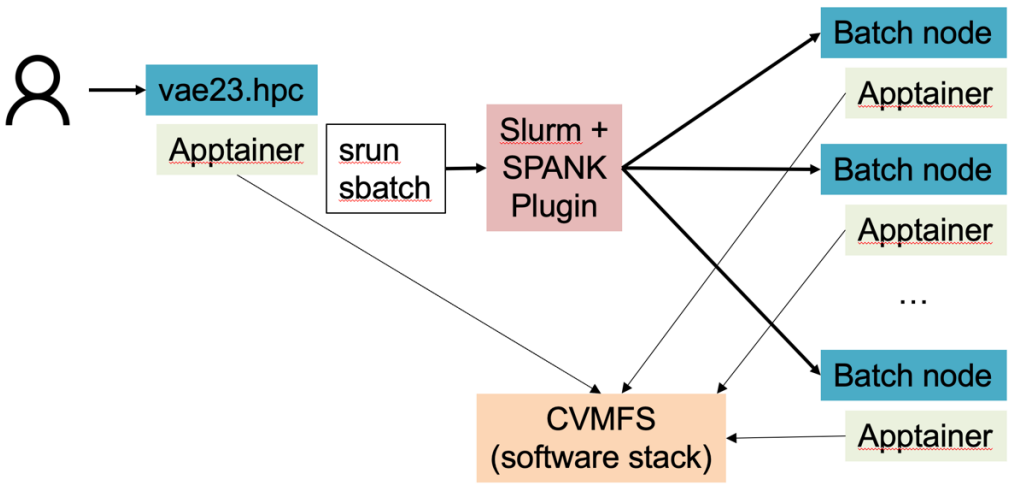
Default submitter nodes offer login to the host OS, where the users can develop their own containers and submit jobs to the cluster.

Dedicated submitter nodes offer a possibility of interactive login into container (VAE). This feature is implemented in a fully transparent way and allows users to have easy access to a large software stack on the cluster.

3.2 Job submission from container [8]

A job submitted to worker nodes from a VAE will be by default executed in the same virtual environment. This is handled internally in a dedicated plugin of the Slurm resource manager. An illustration of such a default workflow is shown in figure 1. By overwriting specific environment variable in the shell of the submitter node, users can execute a task in a custom virtual environment.

Figure 1. Schematic workflow of a job being submitted from within a VAE.



4. Summary

We conclude, that the HPC cluster at FAIR, designed and built based on fully containerized concept, is fulfilling the requirements for online and offline data analysis of experiments at FAIR accelerator. Support of user defined containers, usage of CVMFS shared file system and large software stack, leverage the scalability of FAIR compute cluster.

References

[1] *FAIR Home.* (n.d.). https://fair-center.de/

[2] Messchendorp, J. et. al. (2023). Conceptual Design Report for FAIR Computing. [Manuscript submitted for publication.]

[3] Senger, P. et. al. (2020). Physica Scripta 95 074003. https://doi.org/10.1088/1402-4896/ab8c14/

[4] Kurtzer, G. M. et. al. (2018). *Zenodo*. https://doi.org/10.5281/zenodo.1308868/

[5] Yoo, A. et. al. (2003). *Job Scheduling Strategies for Parallel Processing*, volume 2862 of *Lecture Notes in Computer Science*, pages 44-60, Springer-Verlag

[6] *CernVM-FS 2.11.3 documentation.* (n.d.). https://cvmfs.readthedocs.io/en/stable/

[7] *Containerize OpenSSH Client Logins.* (n.d.). https://github.com/vpenso/openssh-container-login/

[8] Kretz, M. et. al. (2024). Zenodo. https://doi.org/10.5281/zenodo.11189831/