Standardizing HPC-resource adaptation for HEP workflows

Tomas Lindén[1], Gianfranco Sciacca[2], Ievgen Sliusar[3]
On behalf of the ATLAS and CMS Computing Activites

HIP[1], UniBE[2], UiO[3]

25.05.2026

CHEP 2026, Bangkok, Thailand, 25th of May 2026

Acknowledgements

This is largely the work of *Ievgen Sliusar*,
with computing platform and CMS specific integration provided by *Tomas Lindén*,
and additional ATLAS integration support by *Gianfranco Sciacca*.

Figure: The LUMI EuroHPC at CSC

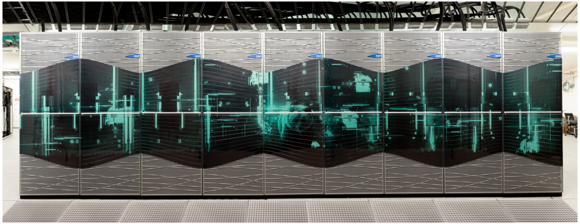


Figure: The CSC Mahti HPC

Contents

Introduction

HPC systems can potentially be a large HEP resource pool if enabled for HEP workflows

- Often HPCs have been integrated case by case with ad hoc solutions for HEP usage

Challenges of HPC Environments

- *Software challenges*
  - Often lack a system installed CVMFS [1]
  - Only unprivileged access
  - Only SSH or web-based access, sometimes no outbound network access at all
  - No ability to change the OS configuration (e.g. to run nested containers out of CVMFS)
  - Cannot host services inside HPC environment
    - needed to interface to the experiment frameworks, based on pilot jobs calling actual payloads
    - remote non-interactive access to the job scheduler
- *Hardware challenges*
  - Unique storage configuration
    - Local disks on compute nodes not always available
    - Filesystem often not performant for HEP workloads
    - Local scratch, shared scratch, shared project software, home directories, ...
  - Often memory constrained nodes

The problem: The three HPCs used in this work lack a native CVMFS installation and have a restrictive container configuration.

Introduction

The goal of this project is the development of a generic solution for the HEP integration of flagship HPC systems worldwide:

- providing a standard high-throughput execution interface
  - Use **ARC CE** [2] as the jobsubmission interface to HPCs
  - It also supports Input/Output Data Staging, job logs and accounting
- ensuring a consistent environment for job execution with a new tool **fapptainer** [3]
  - Run unmodified experiment's pilot jobs in containers with a standard software environment
  - Support running containers from within CVMFS as needed
- Maintain portability by using stock open source software whenever possible
- Outbound SSH is used to access the HPC system and to submit jobs
  - SSHFS used to mount remote shared HPC filesystems - standard component
  - Wrapper scripts to invoke the batch system
- Limitations
  - A fast network connection between the HPC and the ARC CE is required
  - A single account is used for all jobs - preferably a robot account

Computing infrastructure

HPCs hosted by the Finnish IT Center for Science (CSC) in Kajaani

- **LUMI** - EuroHPC Joint Undertaking pre-exascale
  - HPE Cray EX, CPU partition: AMD EPYC 7763 "Milan" Zen 3, SLES15, HPE customised
  - *No local disk on nodes, local disk is preferred for CVMFS cache and local job scratch*
  - Lustre on disk for home, project, scratch areas, also *flash* Lustre, object storage
- **Mahti** - Atos BullSequana XH2000
  - AMD EPYC 7H12 "Rome" Zen 2 CPUs, RHEL9
  - *A few nodes have local NVMe storage*, disk Lustre for home, project, scratch areas
- **Puhti** Atos BullSequana X400
  - Xeon Gold 6230, RHEL9
  - *A few nodes have local NVMe storage*, disk Lustre for home, project, scratch areas

Virtual machines on CSCs cPouta OpenStack

- *ARC CE:s* stock ARC 7, default RTE for starting job in a container
  - **snowarc.hip.fi**, used for ssh submission to **Mahti** (initially to **Puhti**)
  - **arc2lumi.hip.fi**, used for ssh submission to **LUMI**
- Squid proxy VM on cPouta CSC OpenStack, also T2_FI_HIP Helsinki Squid proxy servers in use

HPC job execution environment, storage configuration

Job execution environment

- Run inside a container based on minimal EL9 worker-node image
  - Decouple the execution from the host OS
  - Contains HEP_OSlibs, basic utilities and grid proxy tools
- CVMFS is provided via singcvmfs[4]
  - Runs unprivileged - only requires Apptainer / SingularityCE on the host
  - Mounted inside the container the job runs in
  - Limitation: cannot share cache with jobs on the same node - but can use shared "alien cache"
  - Requires outbound Internet connection

Storage configuration

LUMI:

- Lustre(flash): CVMFS alien cache, image cache, job SCRATCH, ARC session dir
- Lustre(HDD): ARC cache

Mahti:

- NVMe disks on some nodes: CVMFS 50 GB job local alien cache, job SCRATCH
- Lustre(HDD): ARC session and cache directories, image cache

Container challenges on HPC systems

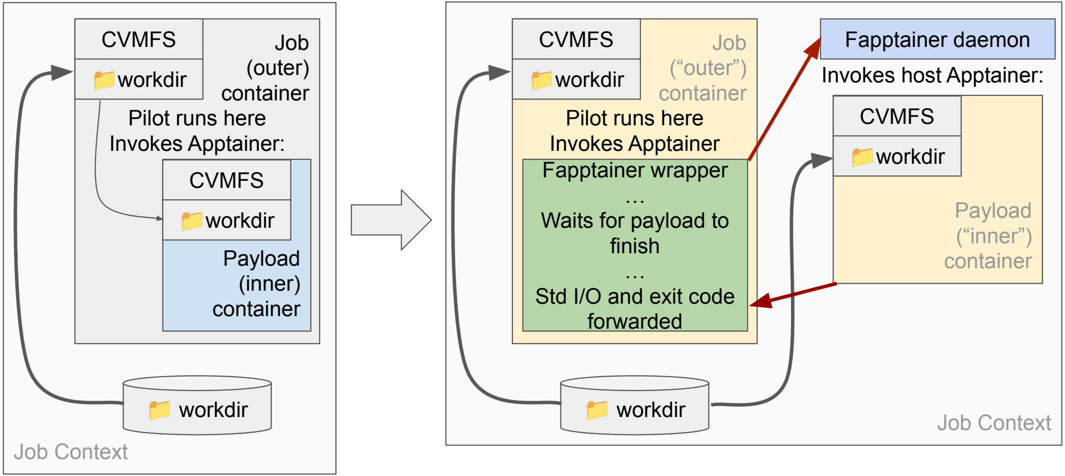
Apptainer/SingularityCE as configured on HPCs does not support *nested* containers

- Relies on unprivileged user namespaces feature of Linux, disabled on HPC systems
- Required to run diverse workloads (without pre-built fat-container)
- Pilot determines the image to run the payload in

This problem is solved by the un-nesting tool **fapptainer** which runs containers sideways instead of nested

- Intercepts container engine invocation inside parent (”outer”) container
- Prepares the needed image locally from CVMFS or public registries by use of wrapper scripts
- Executes new (”inner”) container and forwards the standard I/O streams
- Maintains the same bind-mounts as for the parent and provides CVMFS if requested

Un-nesting containers – Fapptainer tool



Container challenges on HPC systems

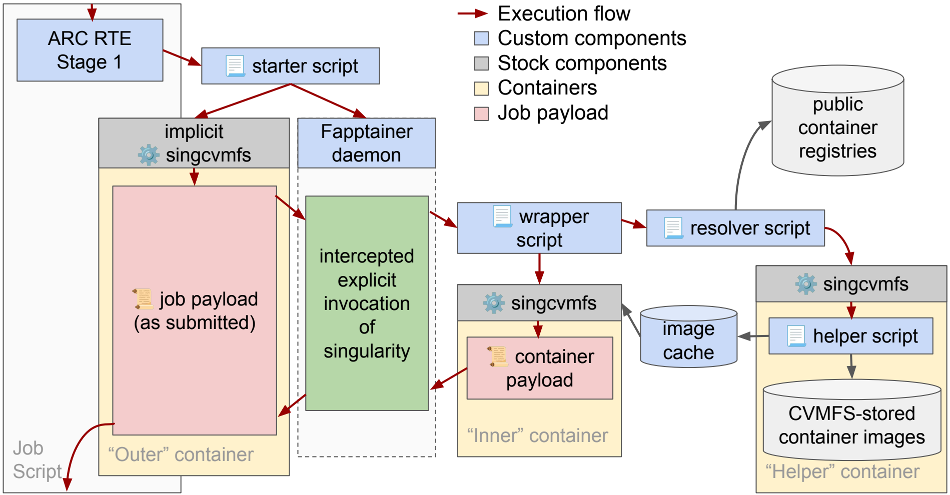
Apptainer/SingularityCE as configured on HPCs do not support running containers from user-mounted filesystems (e.g. CVMFS)

- Depends on enabling user_allow_other in /etc/fuse.conf on host

This problem is solved by a "resolver" script developed to rebuild the required images on the fly from CVMFS or public container registries

- Store and cache them locally on the shared filesystem for re-use by other jobs
  - /cvmfs/atlas.cern.ch/... , /cvmfs/cms.cern.ch/...
  - /cvmfs/unpacked.cern.ch, /cvmfs/singularity.opensciencegrid.org

Execution flow and CVMFS



ATLAS jobs

- Submitted by ARC control tower
  - "pre-loaded" pilots, including the payload and the resources request to LRMS (cores, mem, cputime)
  - Input data pre-staged by ARC CE
- The PanDA Pilot handles the environment setup for the job, selects an appropriate container image and a software release combination from CVMFS
- Resource usage process/device-level statistics from prmon [5] (e.g. RSS, PSS, I/O, CPU rate ...)
- Prmon monitors a specific process tree and now works with fapptainer with visibility into processes of the inner container
- ATLAS has run production workloads on the 3 HPCs Puhti, Mahti and LUMI
- M. Svatos et al., have used fapptainer to run ATLAS jobs on LUMI, see this CHEP 2026 poster [6]

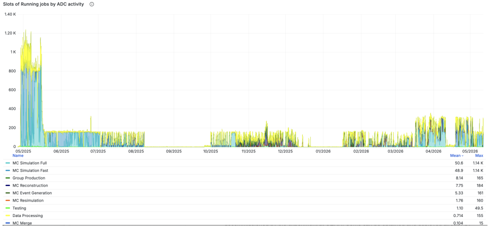


Figure: ATLAS jobs last 12 months on Mahti & Mahti+LUMI

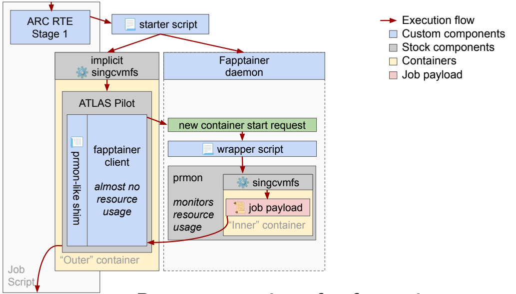


Figure: Prmon container for fapptainer

CMS jobs

- CMS runs ETF and HammerCloud monitoring jobs
- Pilots submitted with GlideinWMS are capable of running multiple payloads simultaneously with *fapptainer handling multiple "inner" containers in parallel*
- Worker nodes stream data directly from/to external data services
- Container images are published in the OpenScienceGrid registry and mirrored to CVMFS
- CMSSW is run from CVMFS in a compatible OS container
- ETF-jobs are green except for IPv6 tests
- Production pilots can run any type of CMS job
- Number of jobs throttled in ARC due to small allocations
- HTCondor resource usage reporting for accounting of jobs with fapptainer is being worked on because HTCondor has no explicit callout for a pluggable resource monitor

Figure: Mahti ETF-jobs 2025.06.01-2026.02.11.

Figure: LUMI CMS pilots job slots 2026.04.29–2026.05.17

Experiences with production jobs

- Memory overhead for running CVMFS in job context $\approx$ 200 MB
  - Some ATLAS jobs memory usage might exceed the request
  - Increase memory request with an ARC CE RTE before submission to SLURM
- Environment variable pass-through and filtering between containers
  - Stock Apptainer forwards all the environment vars to the container it starts
  - Fapptainer captures the environment from outer container but does not automatically forward it to the inner one, was solved by a adding scripts to re-export the vars
- Non-simulation ATLAS jobs require much more input data
  - 1 TB project space filled up in a day initially on Puhti
  - Pilot refuses to call payload if no free space is available (can only be diagnosed from PanDA side, ARC considers such a job as successful)
- ATLAS data staging over SSHFS stressed the small ARC CE VMs
  - The ARC data staging service was implemented on the HPC login node
- Lustre high latency response can be problematic
  - CMS uses lots of small files which is not optimal for Lustre

Summary

- **The three HPCs used in this work have a restrictive container configuration and they lack a native CVMFS installation.**
- **A new approach has been developed to integrate these HPC systems for unmodified HEP experiment workflows.**
- This method builds the standardised environment for HEP computing jobs **using standard open source software** and a newly developed tool, **fapptainer**, requiring the HPC to provide only a functional **container runtime** and **outbound network connectivity**.
- **The presented technique allows running unmodified standard Grid jobs with payloads having access to the CVMFS software repositories and the ability to run nested container images from it.**
- **The approach, designed to be HPC centre agnostic, has been successfully validated by running production workloads on the Puhti (ATLAS), Mahti and LUMI (ATLAS and CMS) supercomputers hosted by CSC in Finland contributing to the experiments' computational resources.**
- *A larger ATLAS production using fapptainer has been done using the Czech national allocation on LUMI [6].*
- In CMS a larger production application is planned for LUMI.
- This work has been done with resources from the LUMI and CSC projects 462000247 (LUMI-as-a-Service), 462000871 (LUMI for HEP) and 2006540 (LUMI pilot).

References I

- Jakob Blomer et al 2011 J. Phys.: Conf. Ser. 331 042003, https://doi.org/10.1088/1742-6596/331/4/042003.
- NorduGrid Collaboration. Advanced Resource Connector (ARC), 2025. Zenodo, https://doi.org/10.5281/zenodo.15080499.
- Fapptainer software - https://source.coderefinery.org/slu/fapptainer.
- Singcvmfs - https://github.com/cvmfs/cvmfsexec.
- Prmon - https://atlas-software.docs.cern.ch/athena/performance/prmon/.
- J. Chudoba, M. Svatos and P. Vokac, *Using HyperQueue to increase utilization of EuroHPC resources by ATLAS Experiment*, https://indico.cern.ch/event/1471803/contributions/6966883/