Karlsruher Institut für Technologie

18th HPC Café

03.09.2026

Agenda HPC Café – 03.09.2026

Topics for today:

- **Focus Topic: Using Containers with Apptainer as Software Environments on HoreKa 2, HoreKa and bwUniCluster 3.0**
- Transition to HoreKa 2: Status
- Discussion and open floor

As usual this format is meant to be interactive.
Don’t hesitate to ask questions!

Karlsruher Institut für Technologie

Using Containers with Apptainer as Software Environments on HoreKa 2, HoreKa and bwUniCluster 3.0

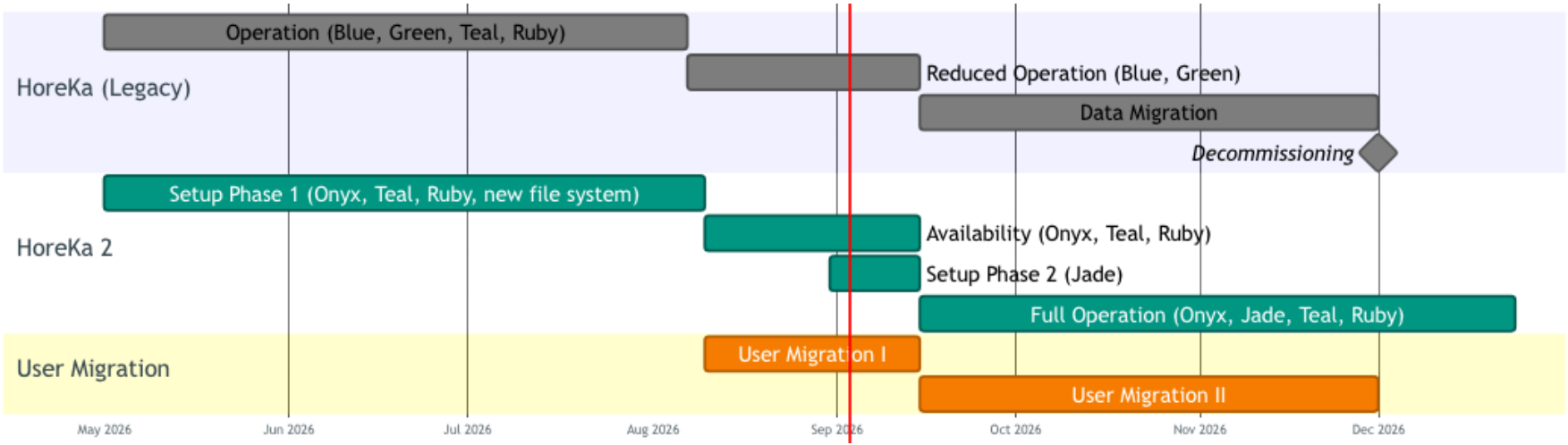
Holger Obermaier (SCC)

Karlsruher Institut für Technologie

Transition to HoreKa 2:

Status

Transition to HoreKa 2: Status



- New documentation under https://docs.nhr.kit.edu
- Transition of CPU projects currently ongoing

Karlsruher Institut für Technologie

Time for Discussion

Next HPC Café

- Next HPC Café planned for **1 October 2026, 10 am**
- Reminder: regular timeslot for the HPC Café
  - Every **first Thursday of the month 10:00 am**

- As always, open to hear ideas for topics and talks, from …
  - Yourself
  - Colleagues and collaborators
  - Also just expressions for “topics of interest”

Apptainer on HoreKa 2

Holger Obermaier | 3. Sept. 2026

Scientific Computing Center

Table of Contents

Motivation: Containerization

Containerization

- Application runs in an isolated user space
- Container provides a full and independent computing environment
  - Configurations, libraries, applications, dependencies
- OS kernel is shared by other containers and native applications

Motivation: Apptainer

Apptainer

- Tool for creating and running *containers*
- Tailored especially for *HPC* usage
- Allows unprivileged users to use containers
  $\Rightarrow$ *Fake root* can be used to build containers
- No *sub-UIDs* or *sub-GIDs* support required
  $\Rightarrow$ Only one user available in container environment
- Isolation with Linux User Namespaces
  $\Rightarrow$ Negligible performance impact
- Host directories can be mapped into the container
- CPU and GPU resources can be accessed from within the container

Video: Interactive container usage

Usage Pros

Reproducibility

- Identical environments across runs
- Consistent results
- Predictable performance

Portability

- Prepare container anywhere (e.g. on your personal laptop)
- Very complex software setups only need to be done once
- Deploy prebuilt container on the cluster

Usage Pros (continued)

Performance

- Build container optimized for the hardware
- **Note: This contradicts portability**

Convenience

- Use the same environment everywhere
- Share the environment within your scientific group
- Use containers prepared by the community
- Predefine your environment variables
- Have the necessary compiler and toolchain ready

Usage Cons

Drawbacks

- The *Easybuild modules* provided and optimized by the operational team are not used
  - Software updates are left to the user
  - Support for software issues is limited
  - Optimal performance requires hardware-matched container libraries
- One more layer of abstraction
  $\Rightarrow$ Debugging gets even more complex
- MPI setup is more complex

Apptainer - Building a Container

Definition Files

- Describe how to build a container
- Configure:
  - Base container image
  - Additional software (e.g. via package manager, or from source)
  - Runtime environment variables

Container Image Sources

- Docker Hub
- AMD ROCm Platform
- GNU Compiler Collection
- Intel oneAPI HPC Toolkit
- NVIDIA GPU Cloud (NGC)

Example: Building a GCC Container

Files and Resources

- Definition file: gcc.def
- GCC environment settings: env_GCC.sh
- OpenBLAS build script: build_OpenBLAS.sh

Video: Building a GCC container

Apptainer - Running an MPI Application

Problem: SLURM Software Package

Problem: `mpirun` invokes `srun` (SLURM) to launche processes
$\Rightarrow$ SLURM normally unavailable inside the container

Solution: Launch processes with `srun` outside the container

Problem: Resource Information

Problem: MPI needs resource allocation details

Solution: Use `srun -mpi=pmix` to pass resource info into container

Example: Running an MPI Application

Files and resources

- NVIDIA HPC SDK (includes NVIDIA HPC-X with latest MPI software stack)
  `apptainer pull docker://nvcr.io/nvidia/nvhpc:26.5-devel-cuda_multi-ubuntu24.04`
- MPI source code: hello_world.c
- Build script: build.sh
- Batch script: hello_world.sbatch

Output

```
Rank 1 of 2 (hk2n1004.localdomain)
Rank 0 of 2 (hk2n1003.localdomain)
```

Video: Running an MPI application