CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Sarus: highly scalable Docker containers for HPC systems

hpc-ch Forum on Virtualization

Alberto Madonna, CSCS

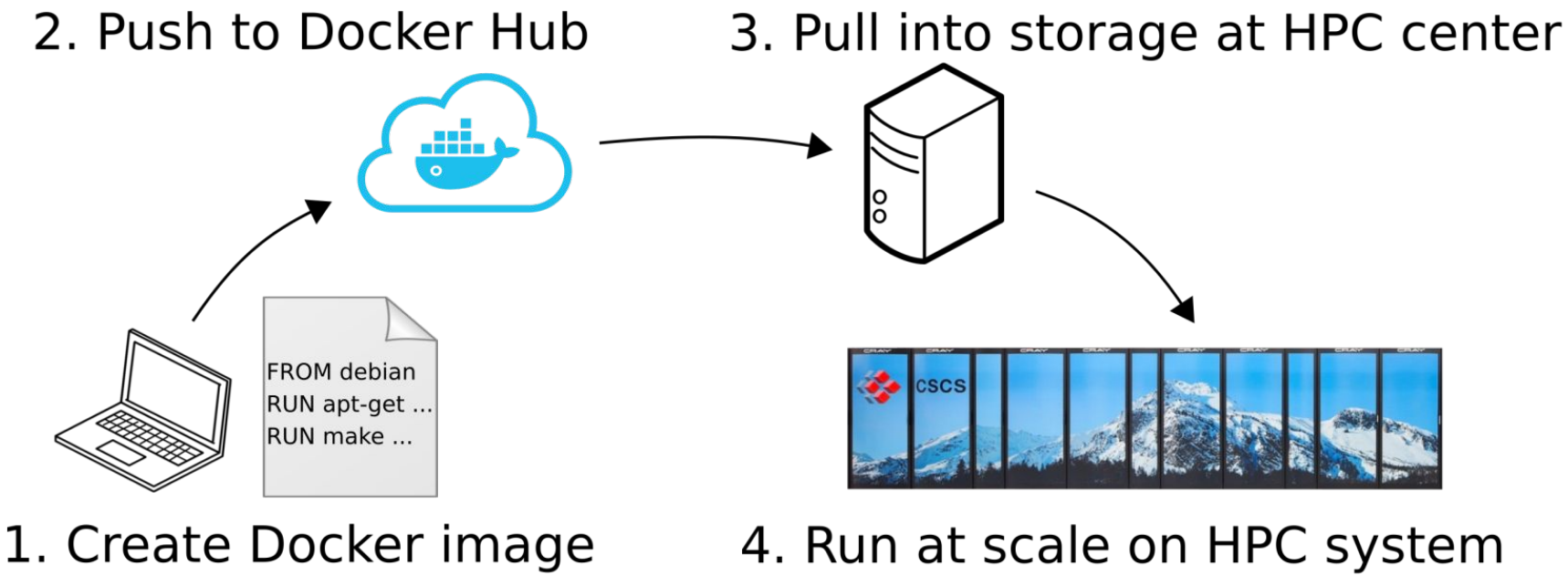
May 20th, 2021

Table of Contents

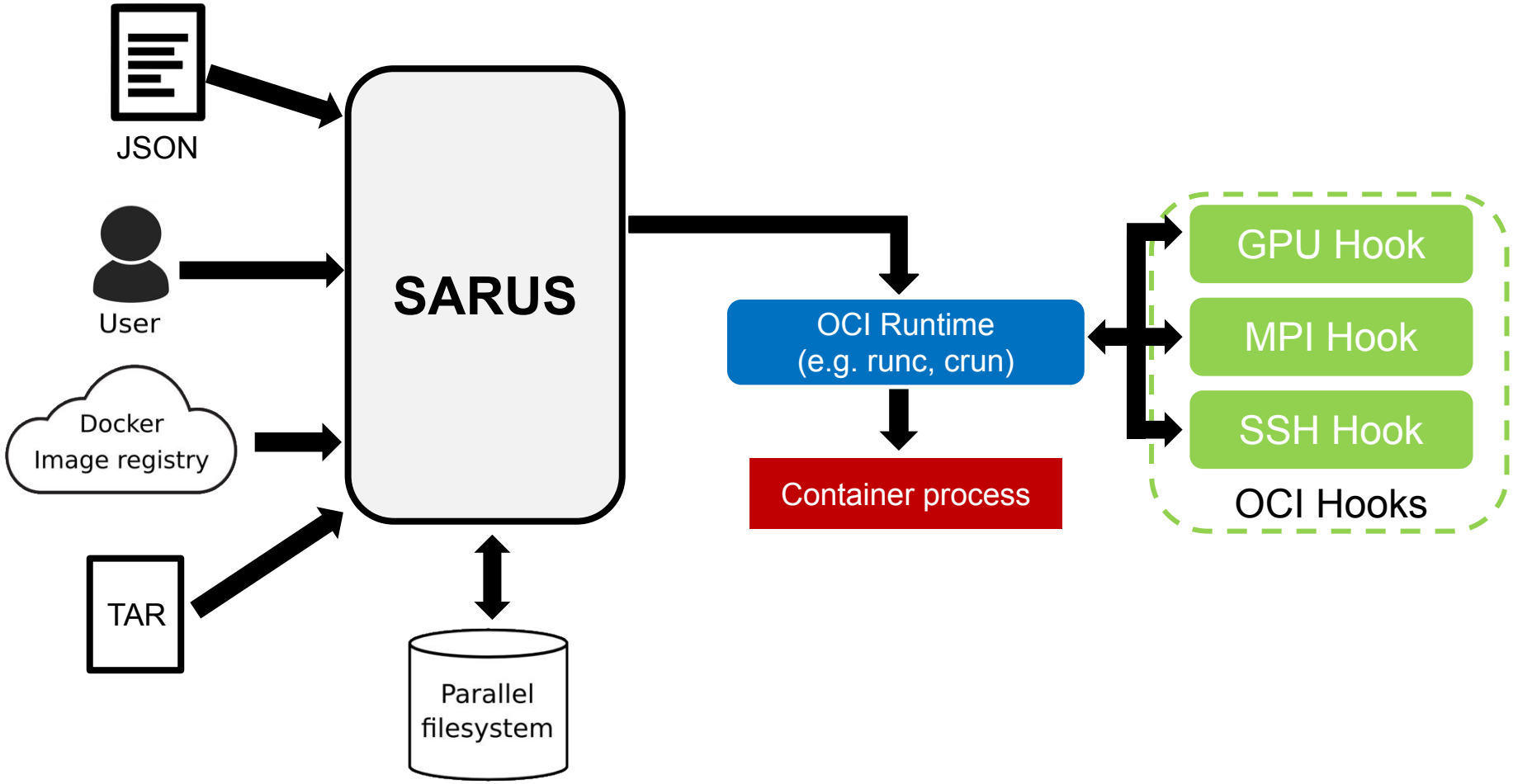
Sarus container engine

- Designed for the requirements HPC
- Consistent UX with Docker: small learning curve
- Transparent native performance through OCI hooks
- Enables use of standard, open, upstream components on HPC systems
- Extensible architecture encourages vendor engagement and improves maintainability

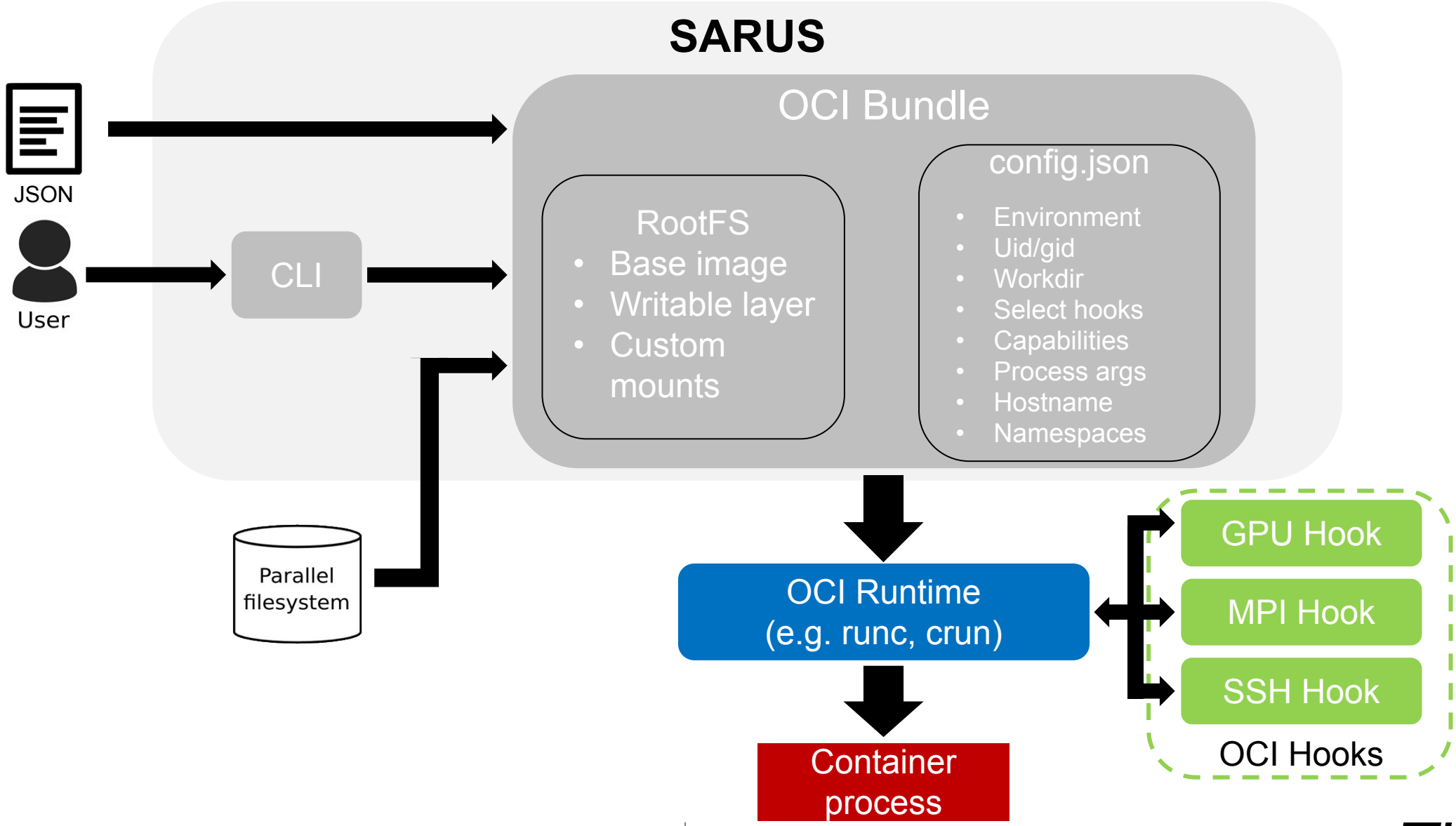
Typical user workflow at CSCS



Architecture overview



Sarus: container execution



HPC features highlights

Features specific to HPC (1): exposing the PMI-2 interface

- MPI libraries (e.g. MPICH) use PMI-2 to communicate between processes
- PMI-2 features specific env variables and UNIX sockets (exposed as file descriptors) to communicate between processes

- **Problems:**
  - PMI-2 processes on the same node communicate through a common /dev/shm
  - runc's file-descriptor preservation mechanism only works with a contiguous set of FDs
  - FD numbers within the container are not guaranteed to have the same value as in the host

- Sarus implements the following:
  - Close/duplicate FDs as needed to create a minimal contiguous set
  - Set PMI-2 env vars in container to use new FD values
  - Mount /dev/shm from the host system

Features specific to HPC (2): integrating CUDA environment

- The NVIDIA Container Toolkit exposes GPUs based on the variable NVIDIA_VISIBLE_DEVICES. This variable is usually set on Dockerfiles to “all”

- **Problems:**
  - Slurm GRES plugin sets CUDA_VISIBLE_DEVICES
  - Inside the container the GPU IDs are reset.
    E.g., GPUs 1,3 on the host become GPUs 0,1

- Sarus implements the following:
  - Set NVIDIA_VISIBLE_DEVICES to honor the WLM allocation
  - Set CUDA_VISIBLE_DEVICES inside the container to ensure correct functionality of GPU apps, even in case of partial or shuffled device allocations on multi-GPU systems
  - Compare this with Docker CLI > 19.03

OCI Hooks

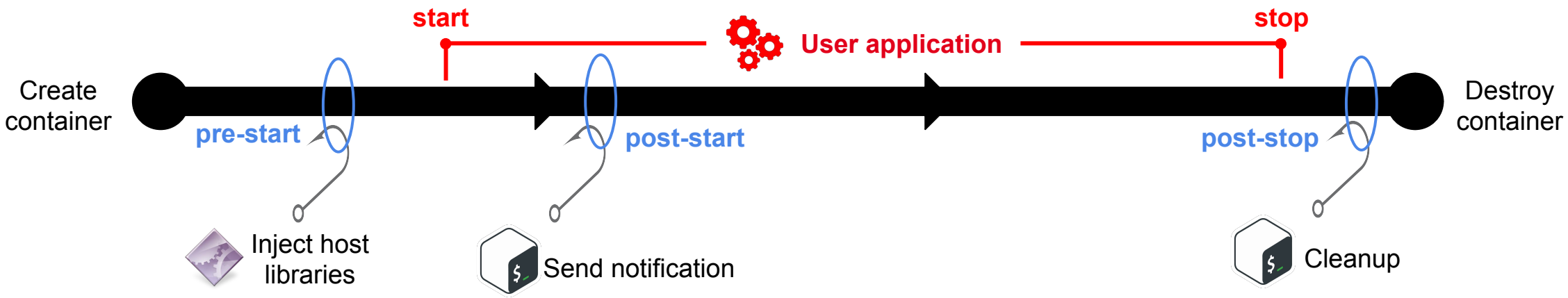
The Open Container Initiative (OCI) Hooks

- “An open governance structure for creating open industry standards around container formats and runtime”

The Open Container Initiative (OCI) Hooks

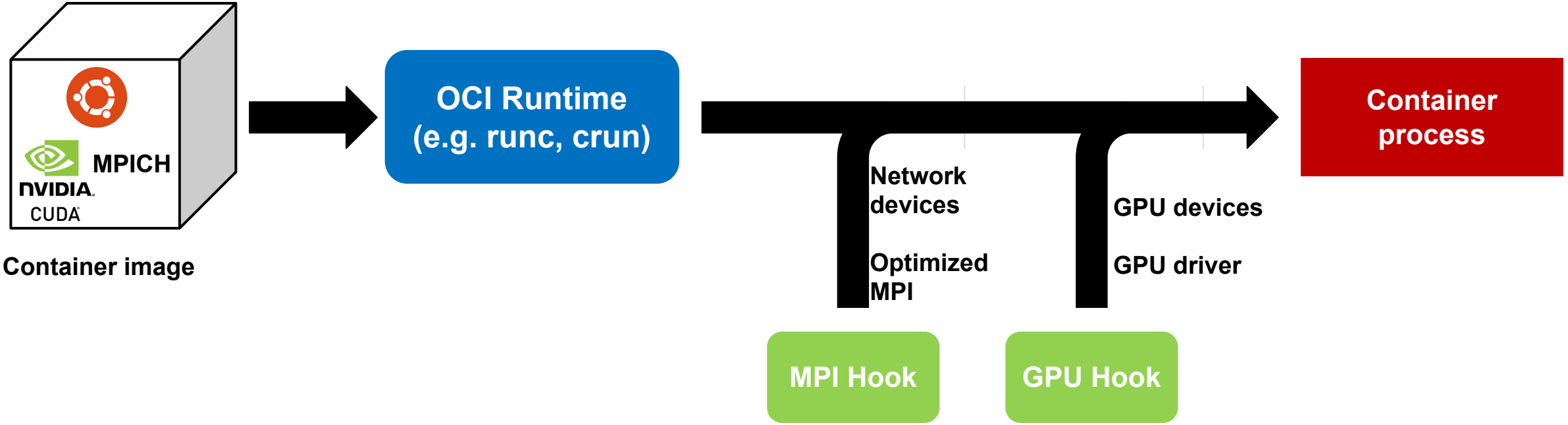
- "An open governance structure for creating open industry standards around container formats and runtime"

- The OCI Runtime Specification defines an interface to plug-in, or **hook**, external programs at certain points in the lifecycle of the container. Such programs can customize the container.



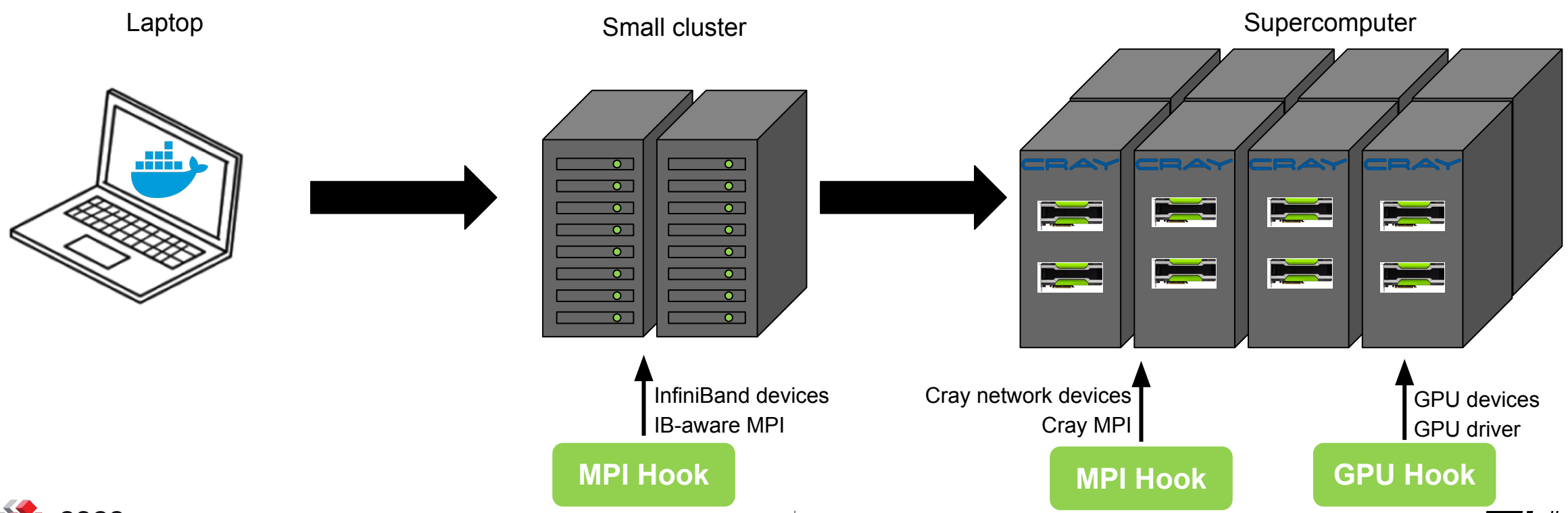
OCI Hooks: runtime customization of portable images

- While the ***image*** remains portable and self-sufficient, hooks can act at launch-time to create machine-specific, high-performance ***containers***



OCI Hooks: tailoring installations to systems features

- By configuring hooks matching the features available on specific machines, admins can maintain leaner installations
- Containers leverage the advantages of each system as users move through the application/research lifecycle



OCI Hooks: enabling separations of concerns

OCI Hooks interface
(part of the Runtime spec)

Container engine / runtime developers

Sarus, Podman

`runc, crun`

`...`

- Do not need to integrate or reverse-engineer the specifics of high-performance technologies

Vendors, technology specialists

- Can develop feature-specific extensions without having to know how containers are created

Results in sustainable, timely, higher-quality support of specific technologies in containers

OCI Hooks used at CSCS

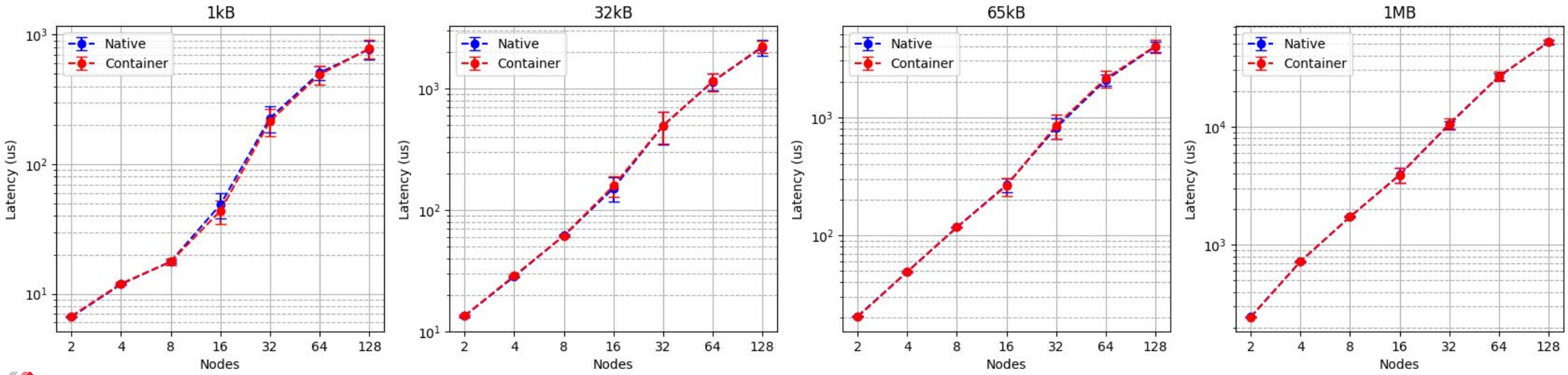
- NVIDIA Container Toolkit for GPU support
- MPI hook (MPICH-based)
  - Native performance from host MPICH-based libraries
  - Developed by CSCS, bundled with Sarus
- Glibc hook
  - Replaces container's glibc if older than host's glibc
  - Ensures that mounted host resources (e.g. MPI) work inside the container
  - Developed by CSCS, bundled with Sarus
- SSH hook
  - Setup ssh connections inside containers
  - Developed by CSCS, bundled with Sarus
- SLURM sync hook
  - Waits for all processes in a SLURM job to start before executing containerized applications
  - Developed by CSCS, bundled with Sarus
- Timestamp hook
  - Writes a timestamp. Useful for developers to time/profile hooks.
  - Developed by CSCS, bundled with Sarus.

MPI Hook

- Replace the container MPI with host libraries at runtime, achieving native performance
- Relies on MPICH ABI compatibility (https://www.mpich.org/abi/)
- Completely transparent to the user:

```
sarus run --mpi ethcscs/osu-mb:5.3.2-mpich3.1.4 ../collective/osu_alltoall
```

OSU all-to-all latency test



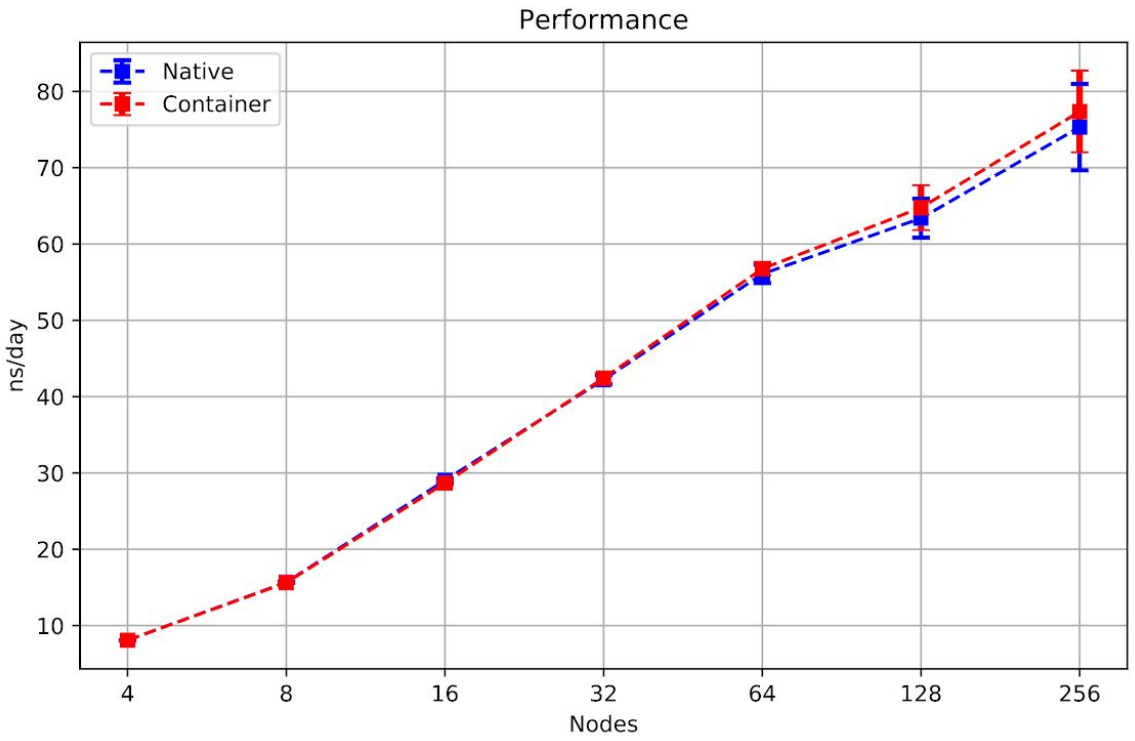
NVIDIA Container Toolkit

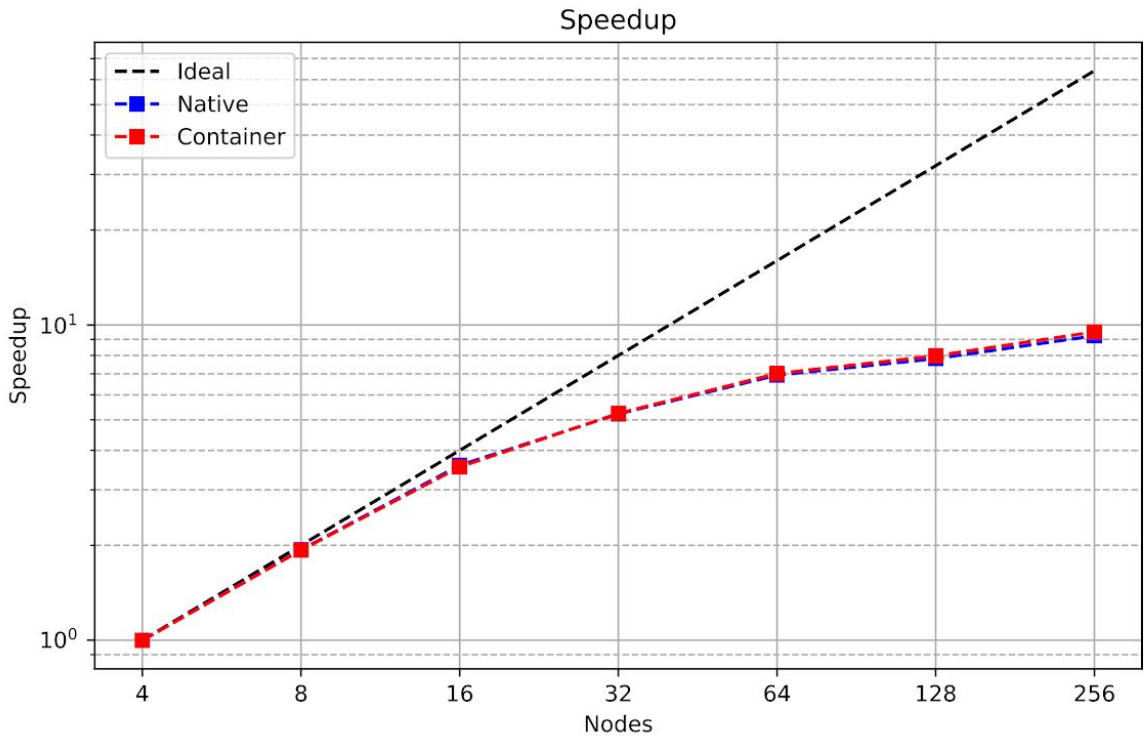
- Open source software by NVIDIA (https://github.com/nvidia/container-toolkit)
- Imports the NVIDIA driver and GPU device files into the container
- Native performance, no input required from the user
- First example of **vendor** hook to be successfully tested on Piz Daint

CUDA SDK N-body sample: FP64 GFLOPS		
	Average	Std. deviation
Native	3059.34	5.30
Container	3058.91	6.29

Performance tests

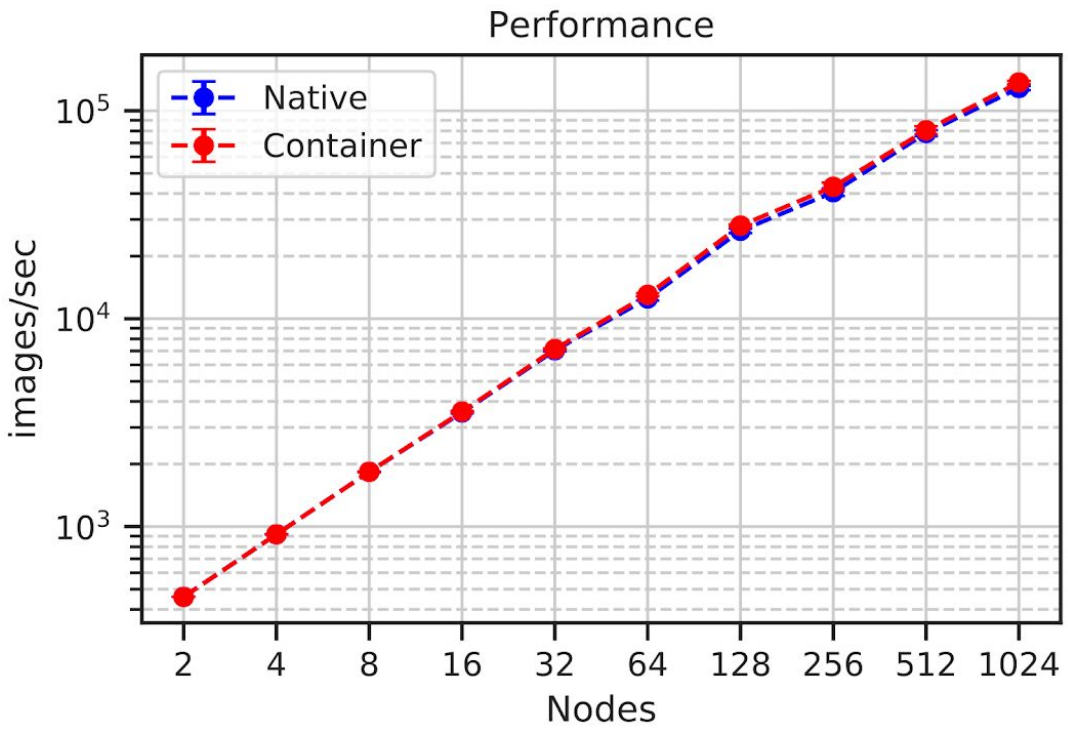
GROMACS (Classical Molecular Dynamics)

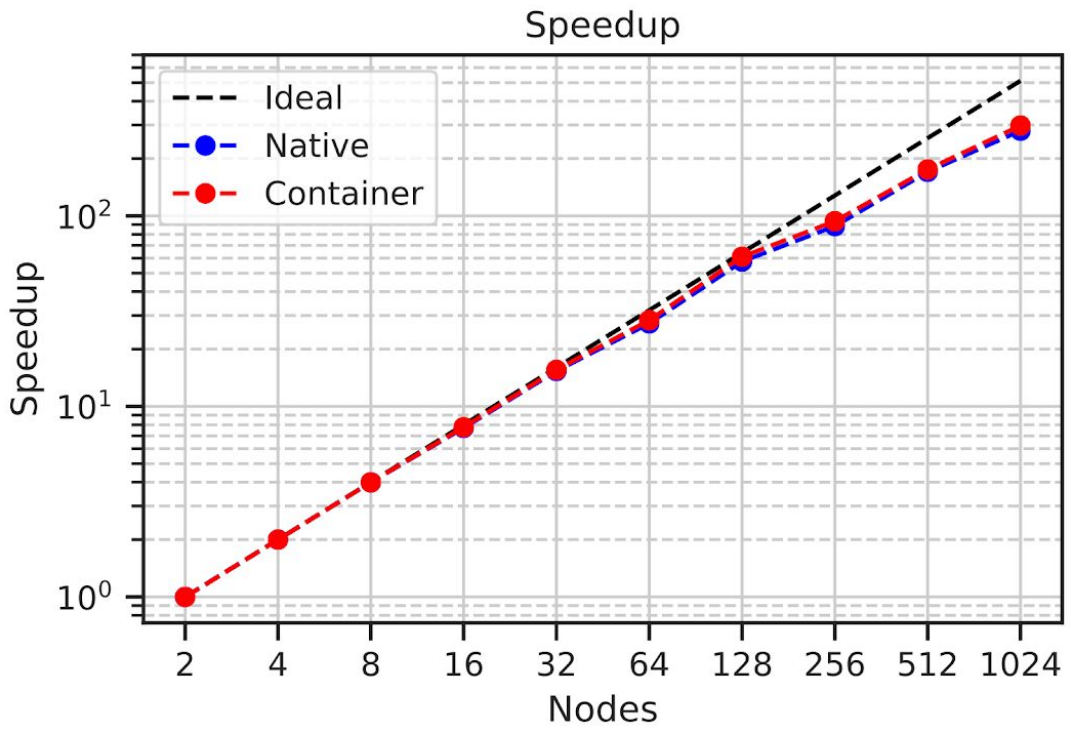




Software: GROMACS 2018.3, CUDA 9.1
Test case: PRACE Unified European Applications Benchmark Suite, GROMACS Test Case B
System: Piz Daint hybrid partition (Intel Xeon E5-2690 v3, NVIDIA Tesla P100, Cray Aries Interconnect)

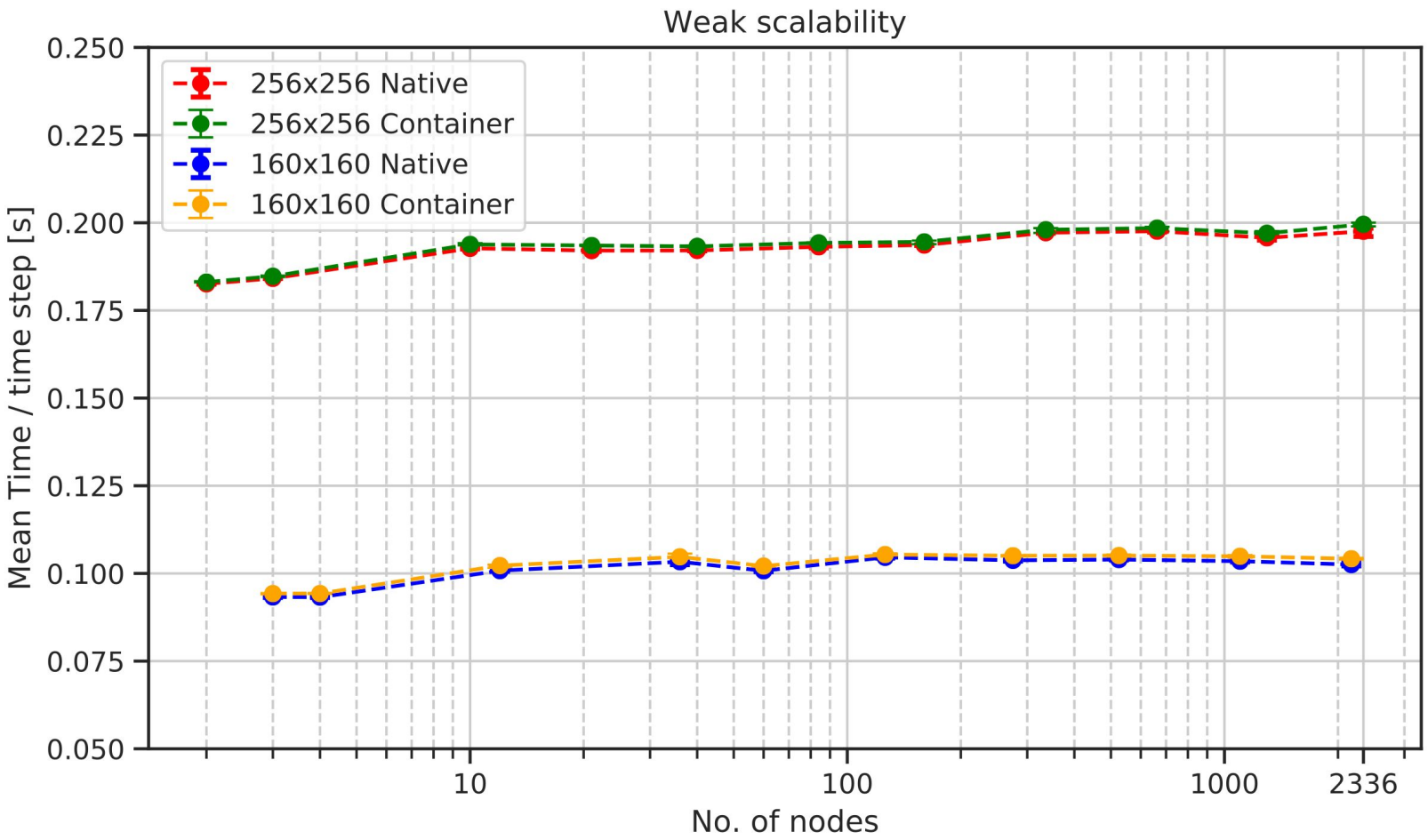
TensorFlow + Horovod (Deep Learning training)





Software: TensorFlow 1.7.0, Horovod 0.15.1, CUDA 9.0
Test case: TF CNN Benchmark scripts, ResNet-50, synthetic ImageNet data
System: Piz Daint hybrid partition (Intel Xeon E5-2690 v3, NVIDIA Tesla P100, Cray Aries Interconnect)

COSMO (Numerical Weather Prediction)



Software: COSMO 5.0, CUDA 9.1
Test case: Near-global idealized baroclinic wave
System: Piz Daint hybrid partition (Intel Xeon E5-2690 v3, NVIDIA Tesla P100, Cray Aries Interconnect)

Conclusion

Sarus is a container engine for HPC, compliant with open standards

- Consistent UX with Docker: small learning curve
- Transparent native performance through hooks
- Enables use of standard, open, upstream components on HPC systems
- Extensible architecture encourages vendor engagement and improves maintainability

Further reading

- Code on GitHub:
  https://github.com/eth-cscs/sarus

- Full documentation:
  https://sarus.readthedocs.io

- Quickstart install:
  https://sarus.readthedocs.io/en/stable/quickstart/quickstart.html

- Benedicic, L., Cruz, F.A., Madonna, A. and Mariotti, K., 2019, June. Sarus: Highly Scalable Docker Containers for HPC Systems. In *International Conference on High Performance Computing* (pp. 46-60). Springer, Cham.
  https://doi.org/10.1007/978-3-030-34356-9_5

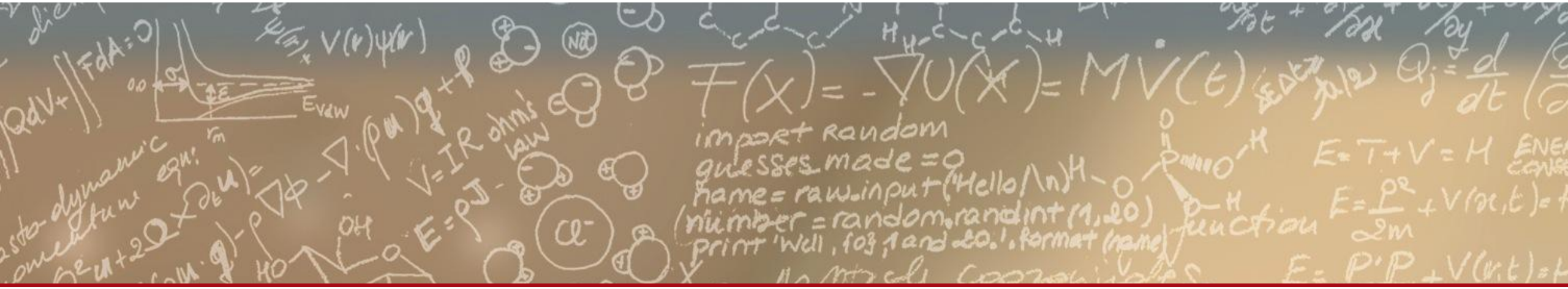
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

EURO

EuroHPC
Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951732. The JU received support from the European Union's Horizon 2020 research and innovation.



Thank you for your attention.