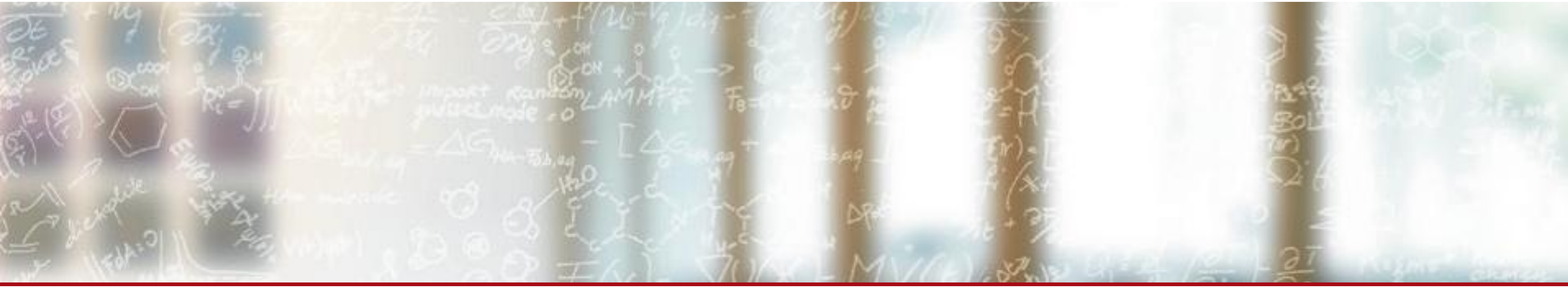
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Sarus: towards HPC performance for portable containers

EuroCC/CASTIEL webinar

Alberto Madonna, ETH Zurich / CSCS
Theofilos Manitaras, ETH Zurich / CSCS

November 12, 2021

EURO CASTIEL

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.

Table of Contents

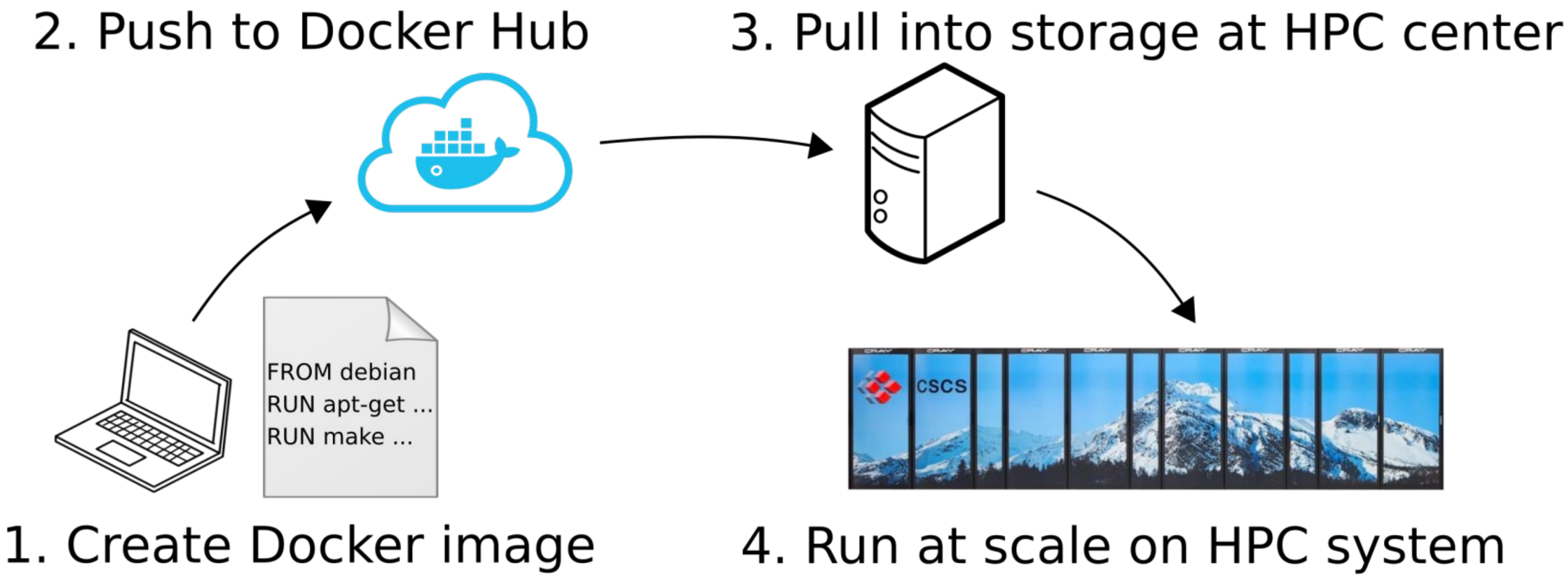
1. Sarus overview
2. HPC features highlights
3. OCI Hooks
4. Demos
5. Performance tests
6. Concluding remarks

Sarus container engine

- Designed for the requirements of 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

SARUS

Typical user workflow at CSCS



2. Push to Docker Hub

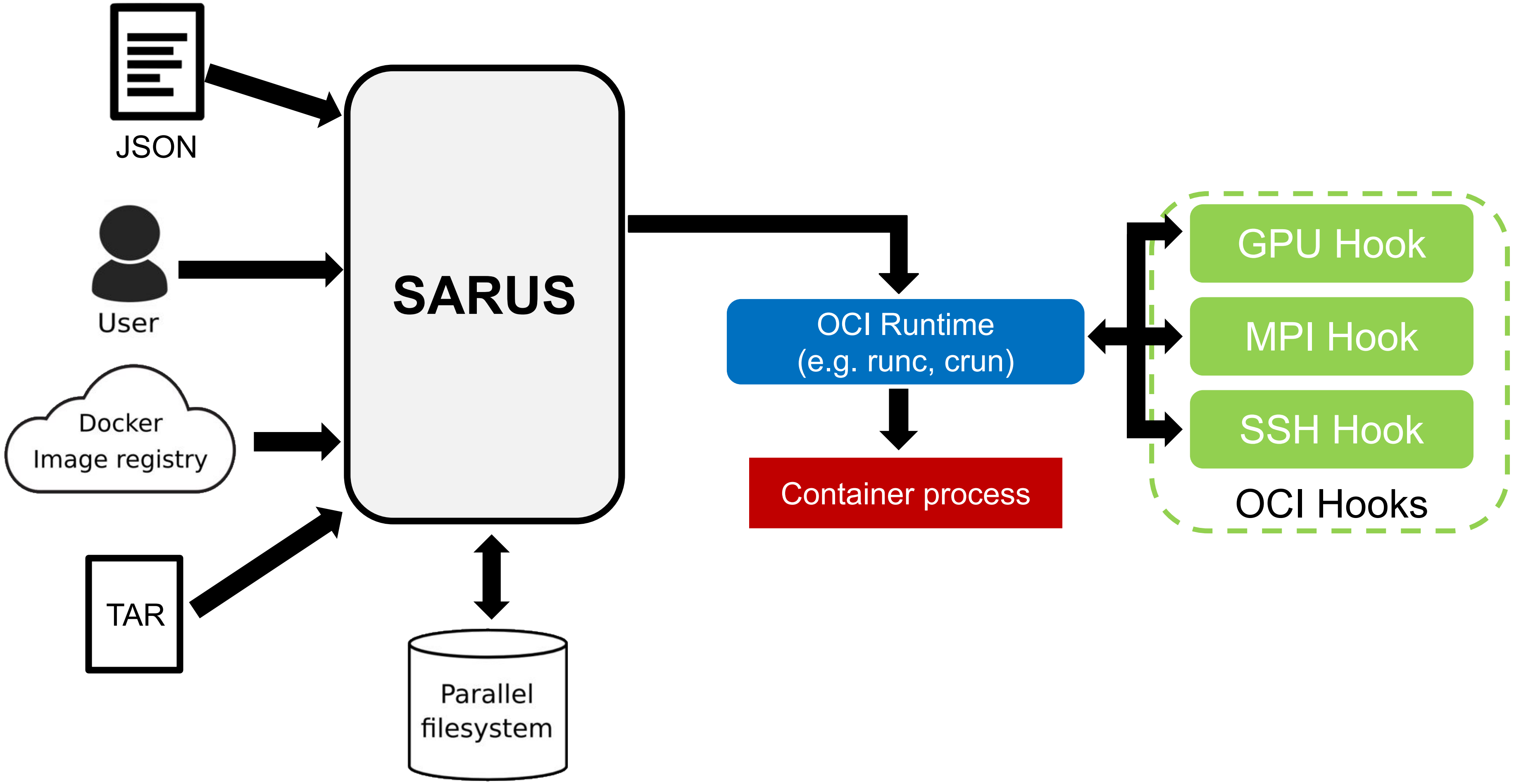
3. Pull into storage at HPC center

```
FROM debian
RUN apt-get ...
RUN make ...
```

1. Create Docker image

4. Run at scale on HPC system

Architecture overview



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

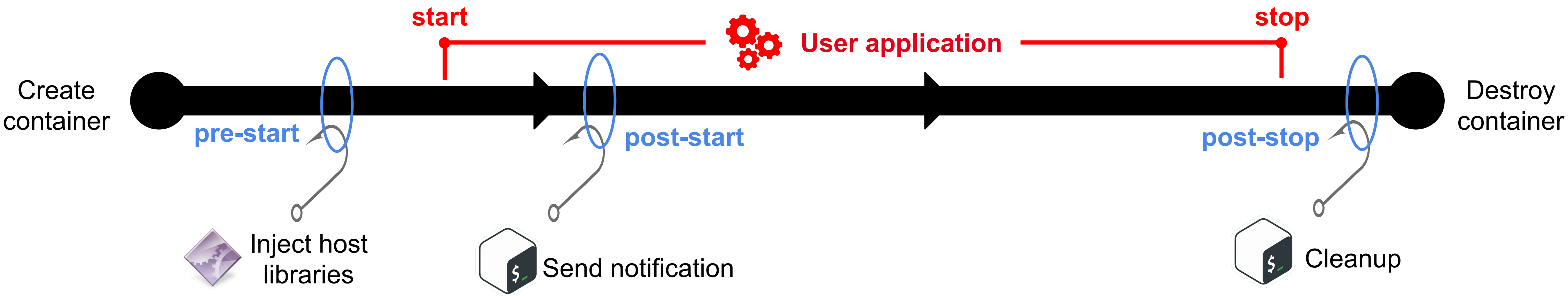
OPEN CONTAINER INITIATIVE

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

The Open Container Initiative (OCI) Hooks

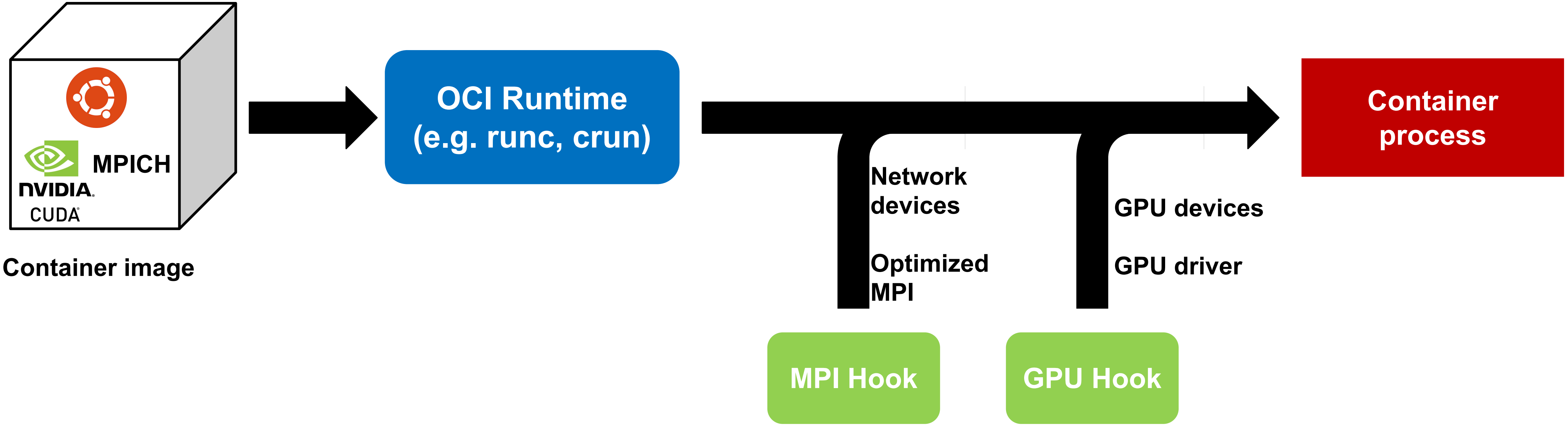
- “An open governance structure for creating open industry standards around container formats and runtime”

- 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.



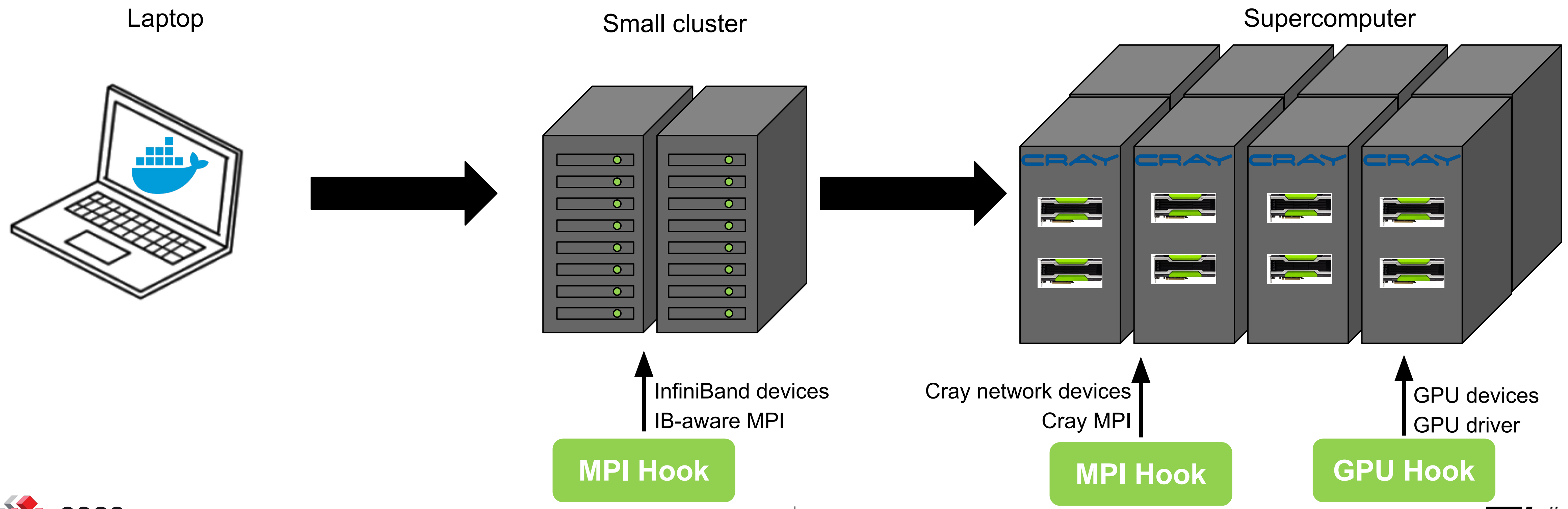
OCI Hooks: runtime customization of portable images

- 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

NVIDIA®

CRAY

- 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

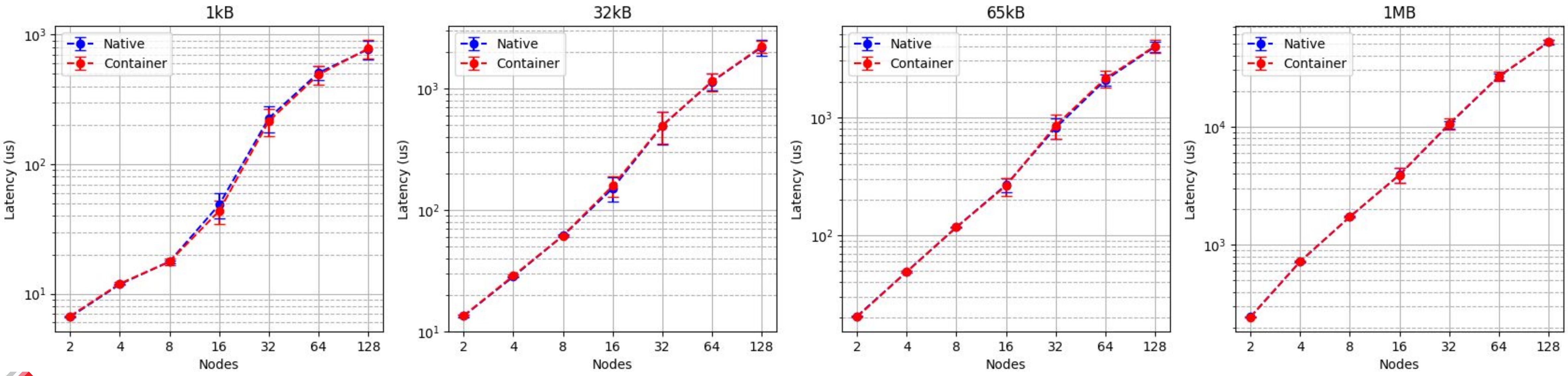
- NVIDIA Container Toolkit for GPU support
- MPI hook (MPICH-based)
  - Native performance from host MPICH-based libraries
  - Developed by CSCS, bundled with Sarus
- Glibc hook
  - Replaces container’s glibc if older than host’s glibc
  - Ensures that mounted host resources (e.g. MPI) work inside the 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

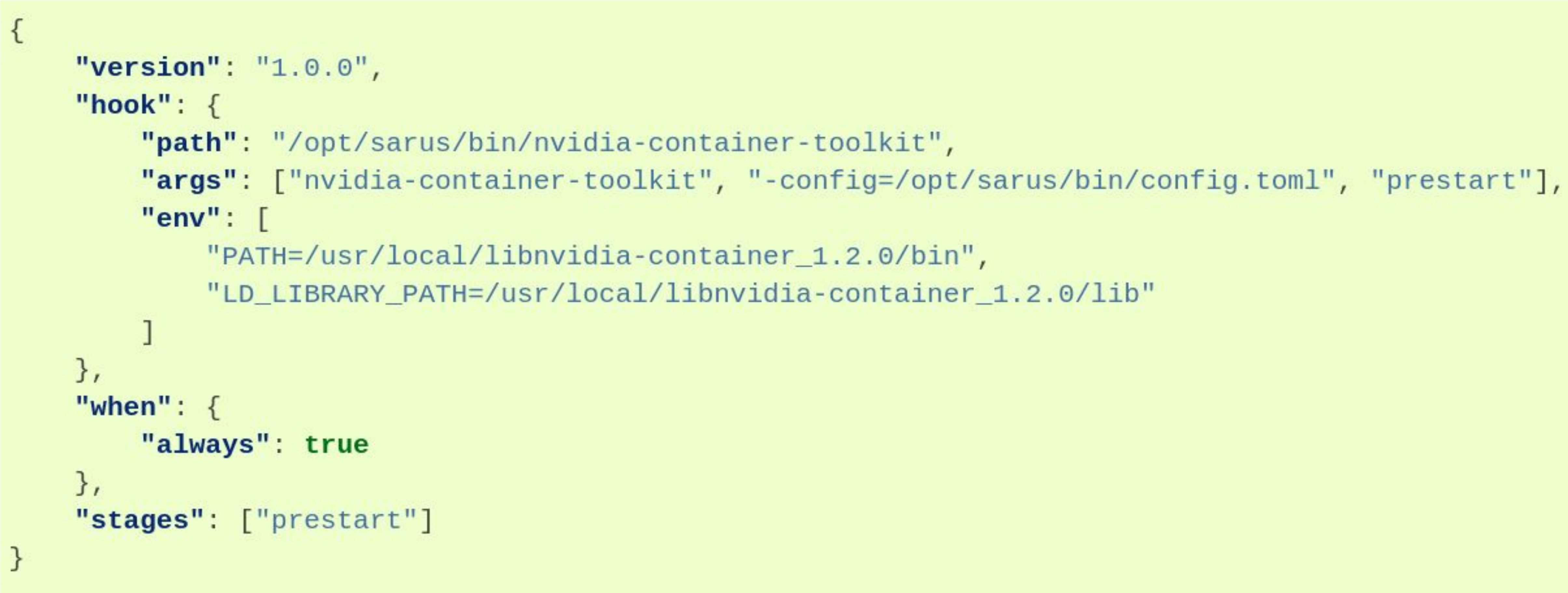


NVIDIA Container Toolkit

- 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

Hook configuration example: NVIDIA Container Toolkit



```
{
    "version": "1.0.0",
    "hook": {
        "path": "/opt/sarus/bin/nvidia-container-toolkit",
        "args": ["nvidia-container-toolkit", "-config=/opt/sarus/bin/config.toml", "prestart"],
        "env": [
            "PATH=/usr/local/libnvidia-container_1.2.0/bin",
            "LD_LIBRARY_PATH=/usr/local/libnvidia-container_1.2.0/lib"
        ]
    },
    "when": {
        "always": true
    },
    "stages": ["prestart"]
}
```

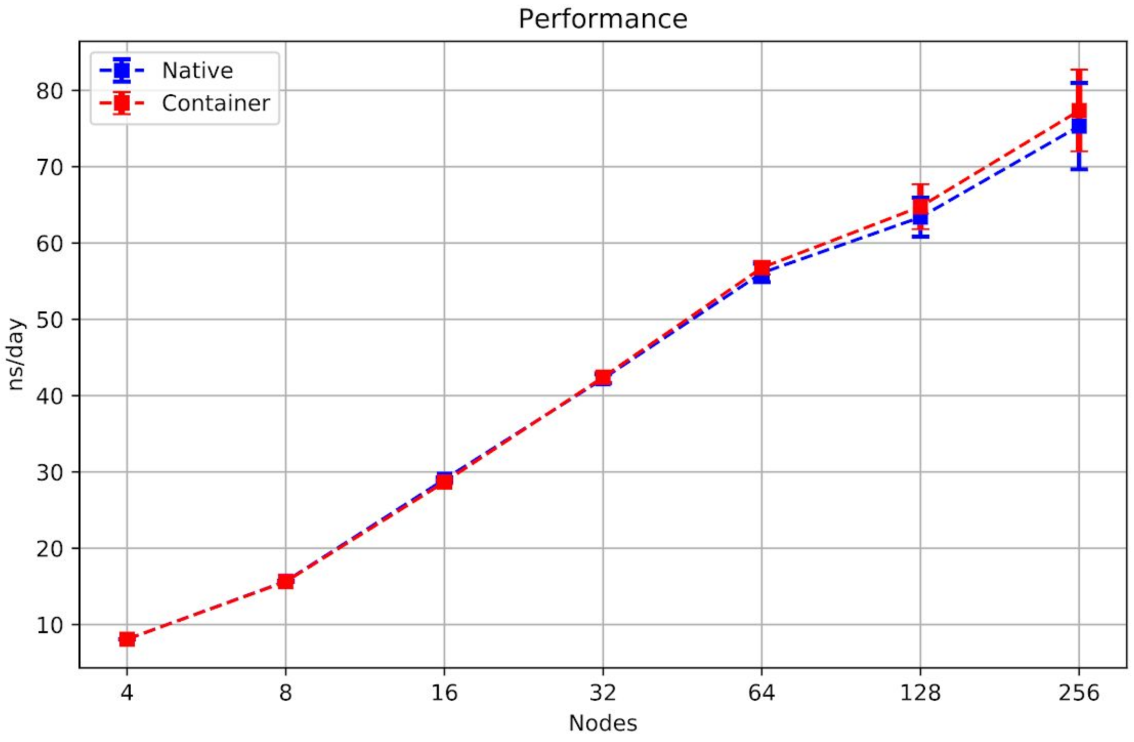
More documentation and examples at https://sarus.readthedocs.io/en/stable/config/configure_hooks.html

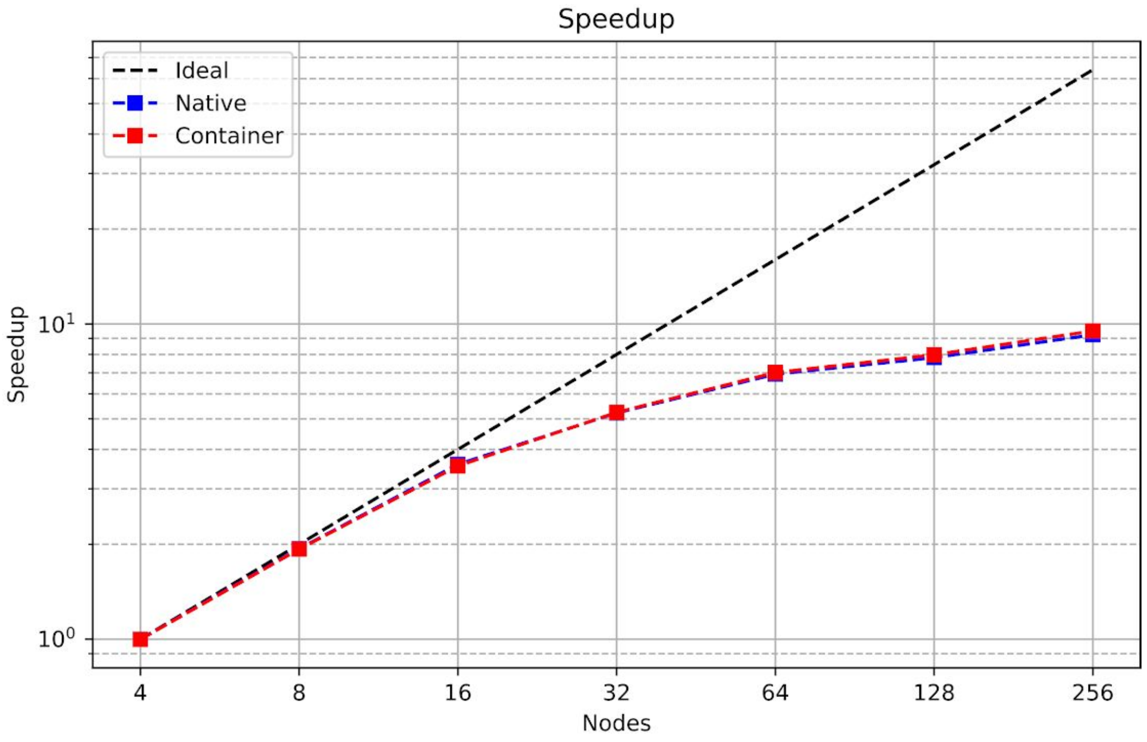
Demo: standalone installation

Demo: container customization (MPI/GPU)

Performance tests

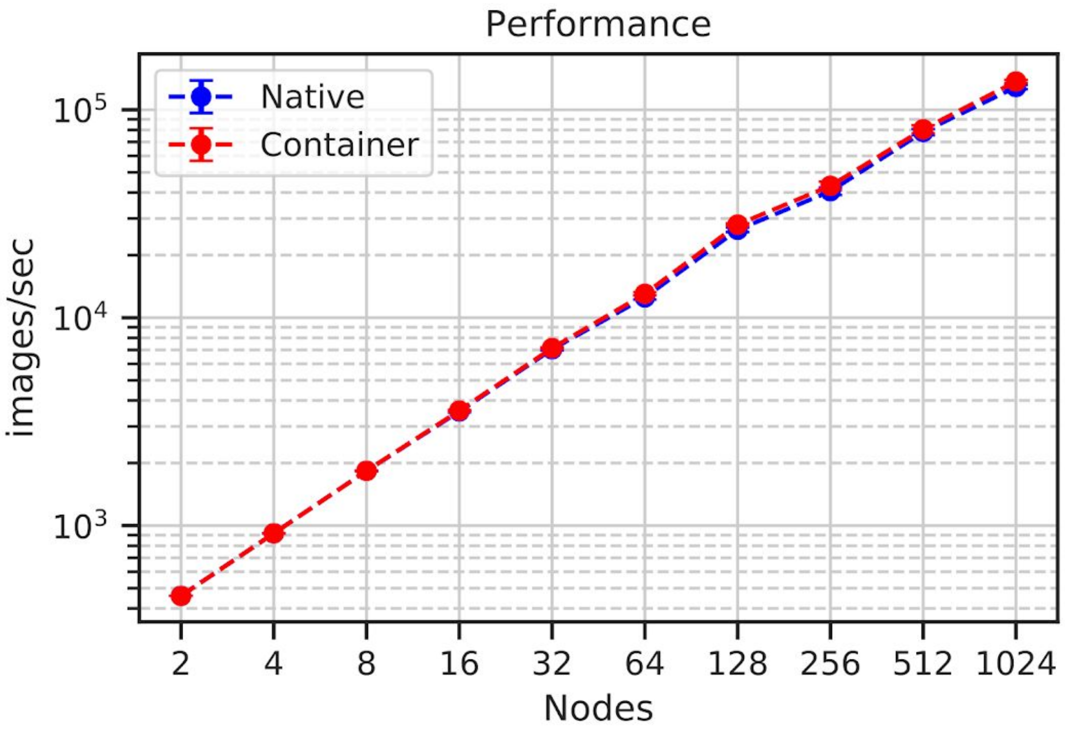
GROMACS (Classical Molecular Dynamics)

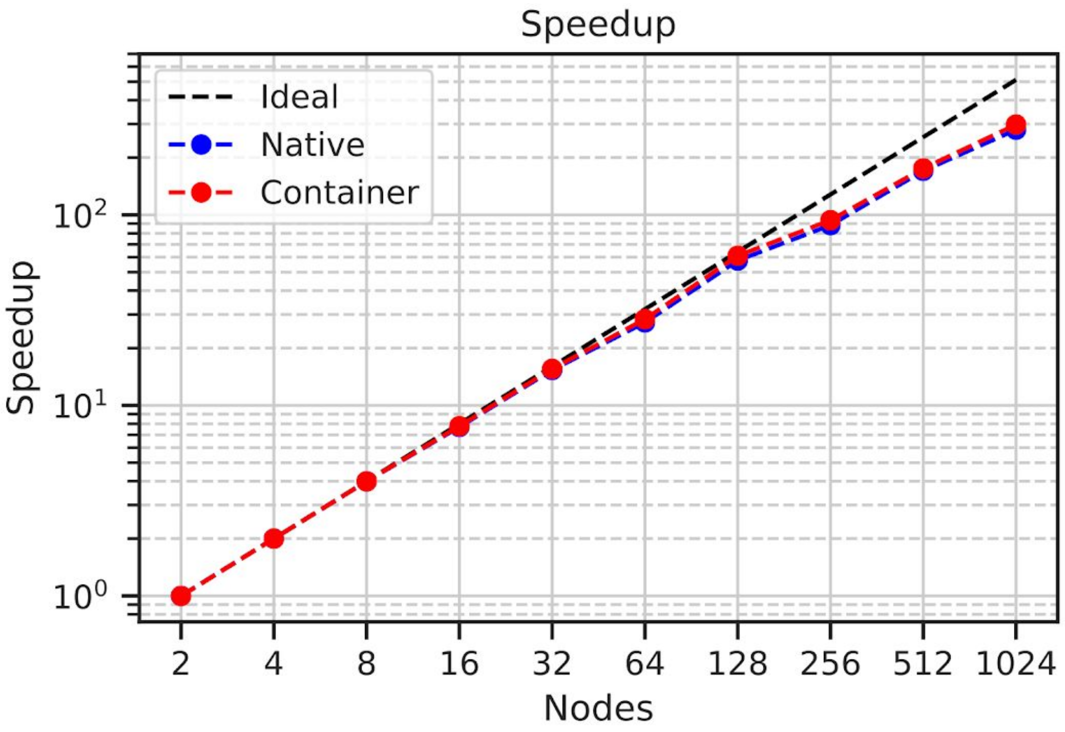




Software: GROMACS 2018.3, CUDA 9.1
Test case: PRACE Unified European Applications Benchmark Suite, GROMACS Test Case B
System: Piz Daint hybrid partition (Intel Xeon E5-2690 v3, NVIDIA Tesla P100, Cray Aries Interconnect)

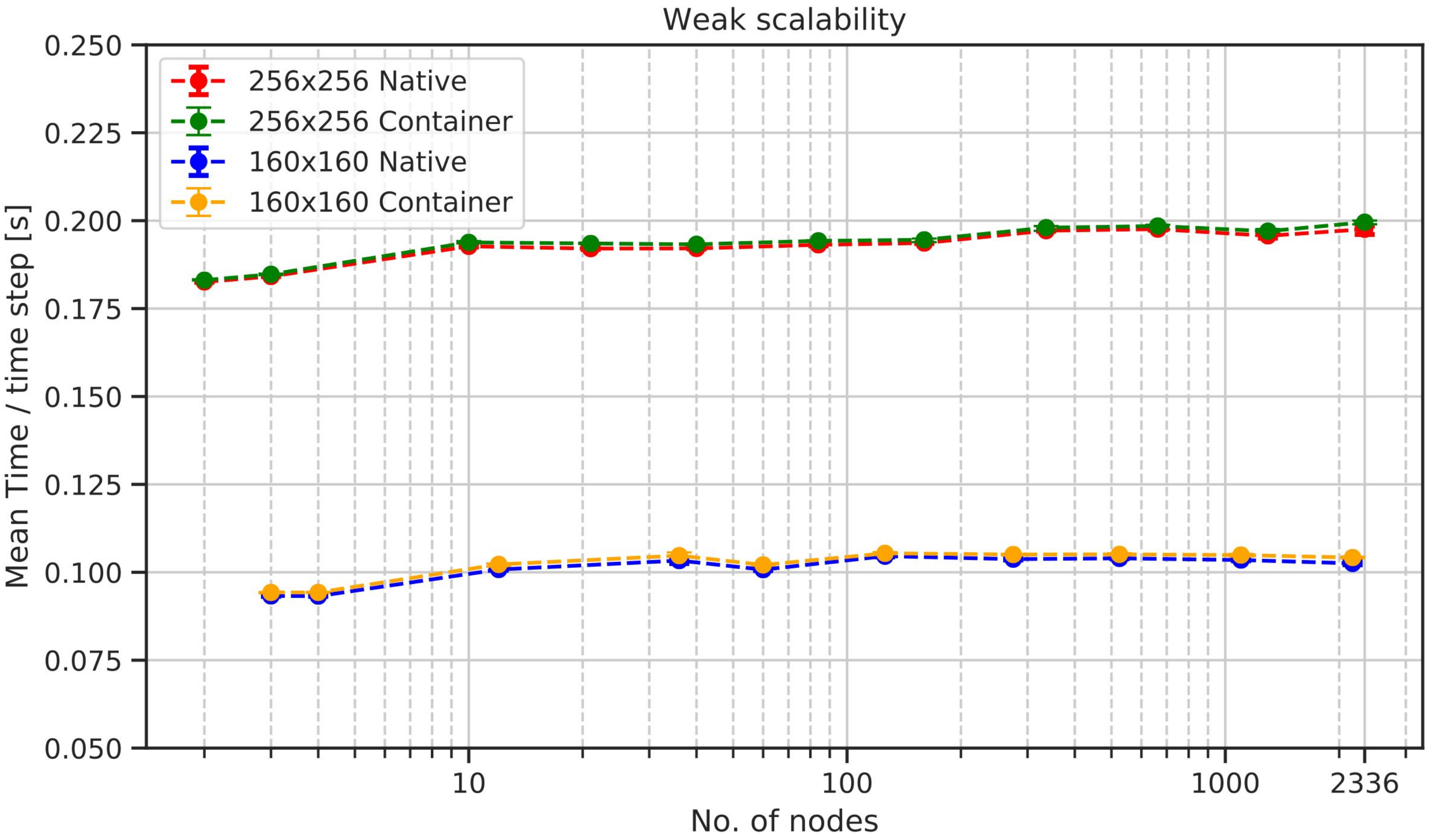
TensorFlow + Horovod (Deep Learning training)





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

- Combines container portability with native HPC performance
- Integrates with HPC infrastructure and software
- Customizes containers at runtime with standard plugins
- Provides a Docker-like command line interface

Further reading

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

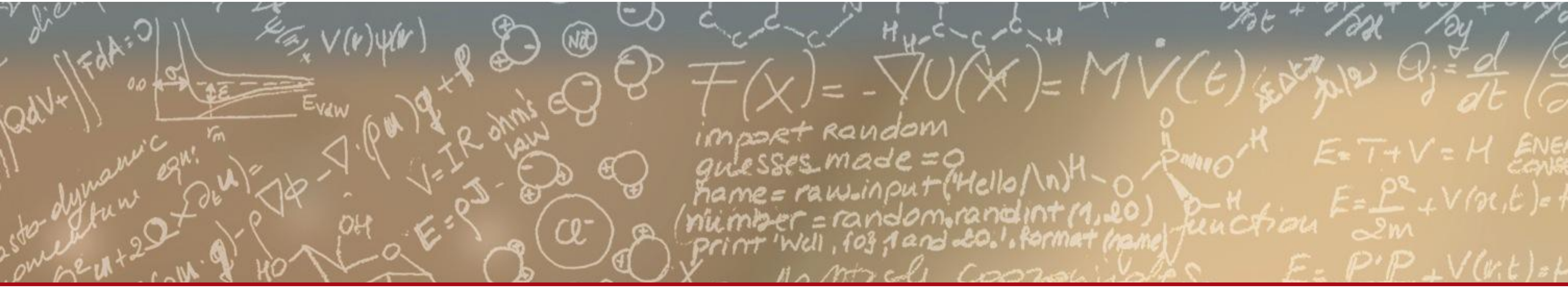
- Full documentation:
  https://sarus.readthedocs.io

- Contact:
  sarus@cscs.ch

- 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



Thank you for your attention.

EURO CASTIEL

EuroHPC
Joint Undertaking

Backup slides

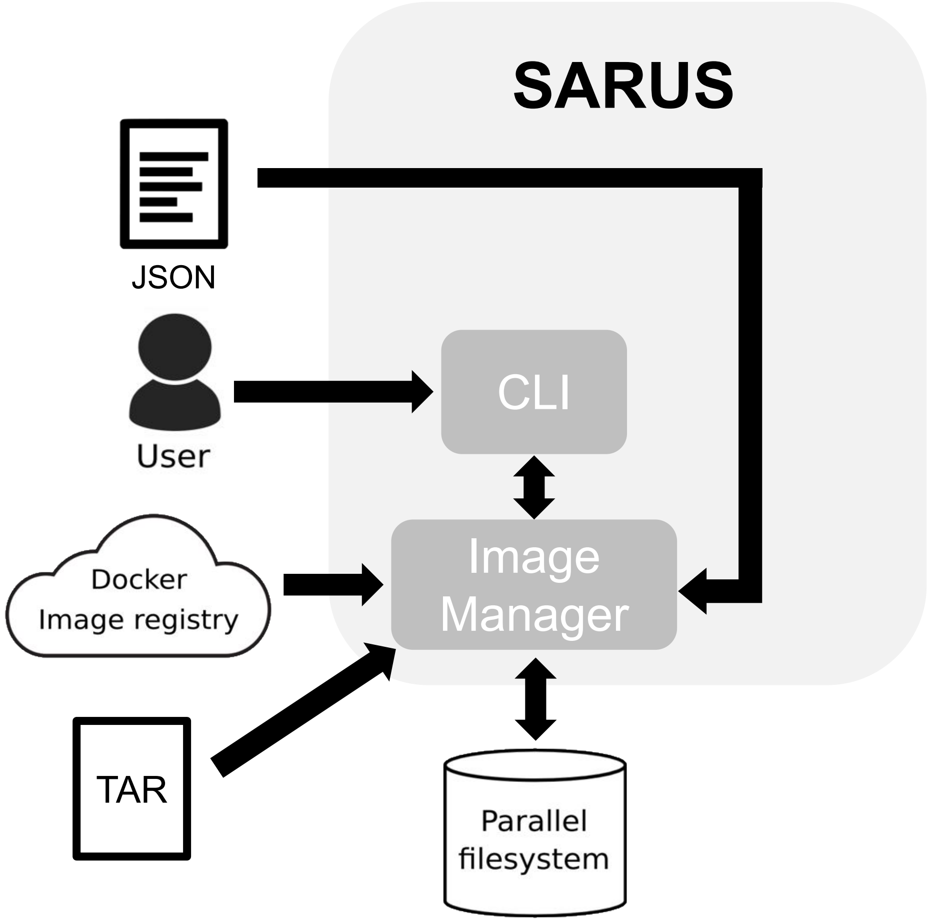
Sarus: image input

- Pull from OCI registry with multiple download threads

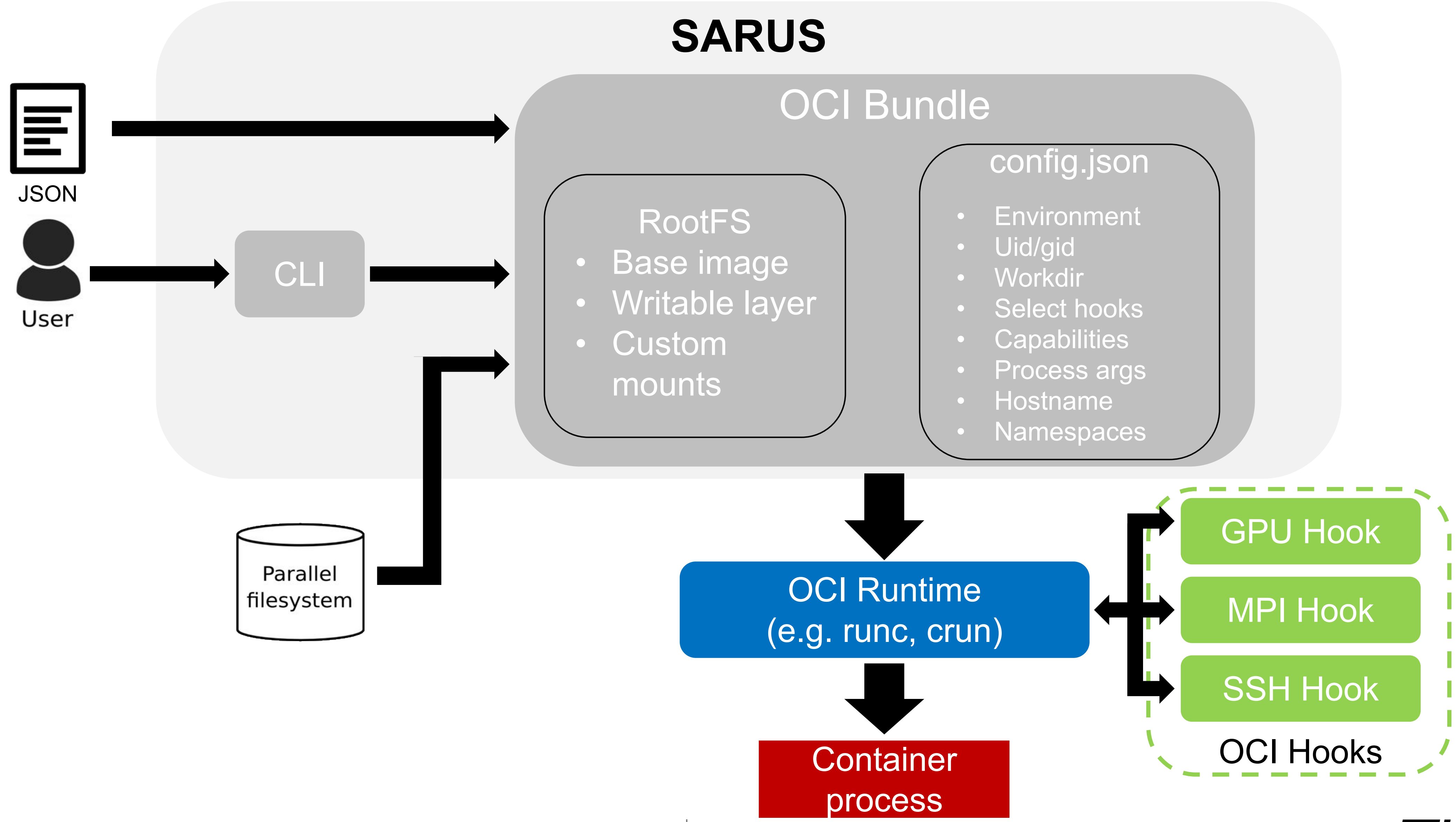
- Alternatively, load layers from local tar file

- Extract all layers and convert them into a *single* squashfs file

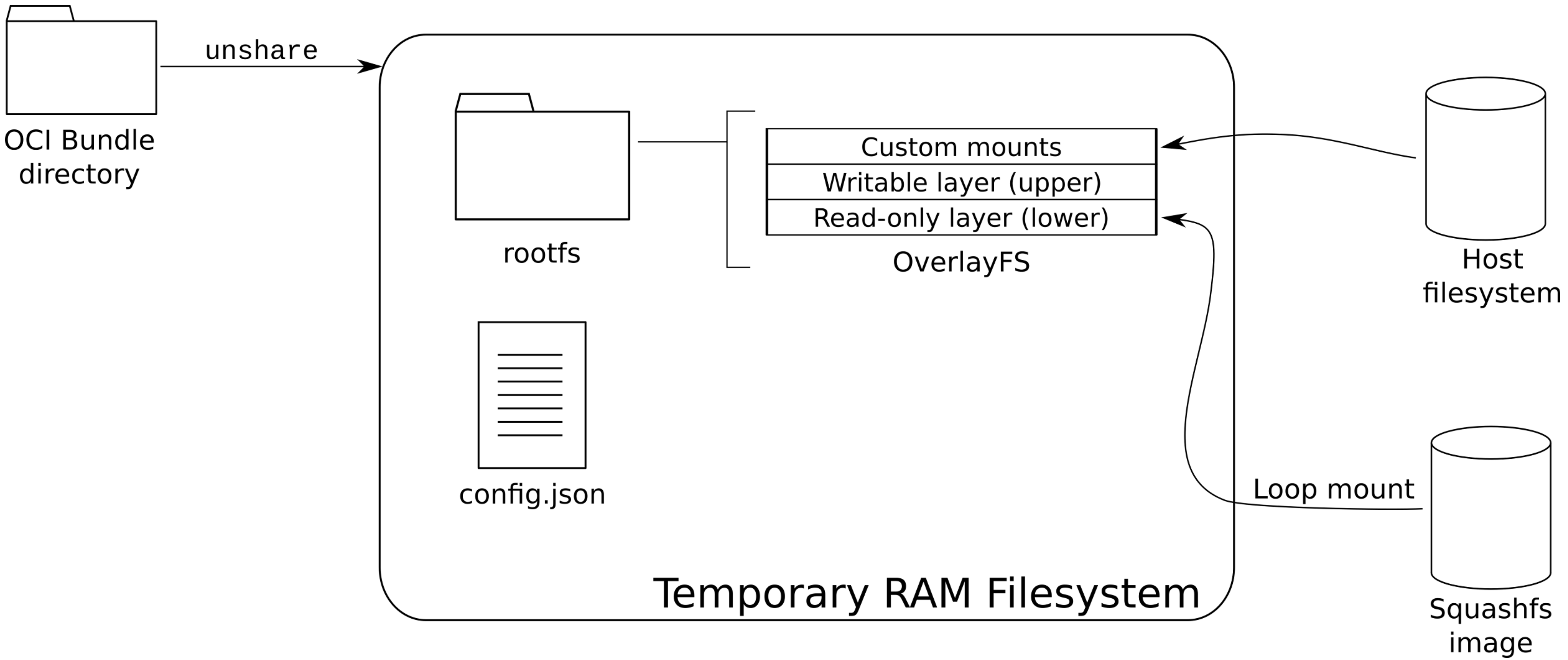
- Accompanying metadata file is generated from image metadata



Sarus: container execution



Sarus: container rootfs creation



Sarus: config.json creation

- Set container process to have the same uid/gid of host user
- Support OCI entrypoint, default arguments, workdir
- Create container env variables by uniting host and image environments (image env vars have precedence)
- Disable all Linux capabilities of the container process
- Set `no_new_privs` flag to 1
- Enable mount and PID namespace isolation
- Set CPU affinity to be the same of the host process (!)

MPI containers on Piz Daint

- Generic images can run unmodified by instructing Slurm to use the PMI-2 interface:

```
srun --mpi=pmi2 sarus run <image> <args>
```

- This way, containers will use the MPI libraries from the image and run at sub-optimal performance

- Images using MPICH and derivatives: work out of the box

- Images using OpenMPI: OpenMPI must be built with PMI-2 support
  - Configure example on Ubuntu 18.04:

```
./configure --prefix=/usr --with-pmi=/usr/include/slurm-wlm --with-pmi-libdir=/usr/lib/x86_64-linux-gnu \
 CFLAGS=-I/usr/include/slurm-wlm
```

MPI containers on Piz Daint

- Images using MPICH-based implementations can take advantage of ABI compatibility (https://www.mpich.org/abi/)

- Sarus can replace the image MPI with host libraries at runtime, achieving the full performance of the Cray Aries interconnect:

```
srun sarus run --mpi <image> <args>
```

- Recommended libraries for compatibility with Piz Daint:

MPICH 3.1.4
MVAPICH2 2.2
Intel MPI Library 2017 Update 1

GPU containers on Piz Daint

- When running on Piz Daint’s GPU nodes, GPU devices are automatically added to containers

- Fastest way to get CUDA in a Dockerfile: use NVIDIA official images! https://hub.docker.com/r/nvidia/cuda

```
FROM nvidia/cuda:11.3.0-devel-ubuntu20.04
```

- NVIDIA images are provided for Ubuntu, Red Hat UBI and CentOS
  - Other distributions can still install the CUDA Toolkit through package manager or runfile

- The NVIDIA driver should NOT be installed in the image (it’s bound to the hardware!)