Containers on Frontier

Subil Abraham

HPC Engineer

OLCF User Assistance

What are Containers?

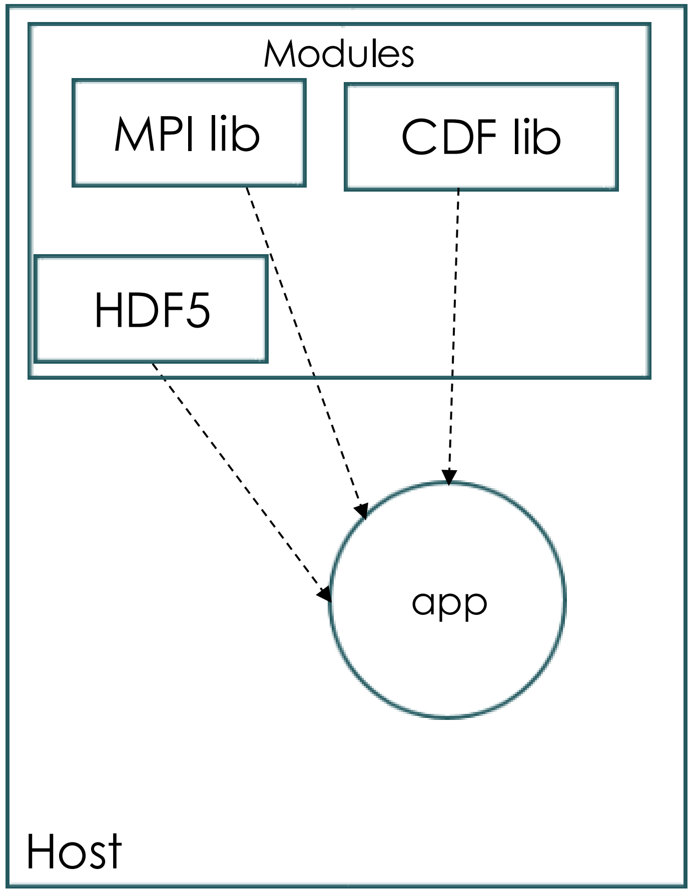
- From Docker's website: A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

Containers in HPC: Why even bother?

- Containers package your application along with its software environment
  - Removes dependence on host system's stack
  - Finer control over system level libraries also
- Byzantine build systems are easier to manage
- Potentially easier to port to cloud or other systems*
- Just hand a container image to a new user on your project
  - Save time setting up their environment from scratch
- Not a VM, runs like an application. So no real performance impact.

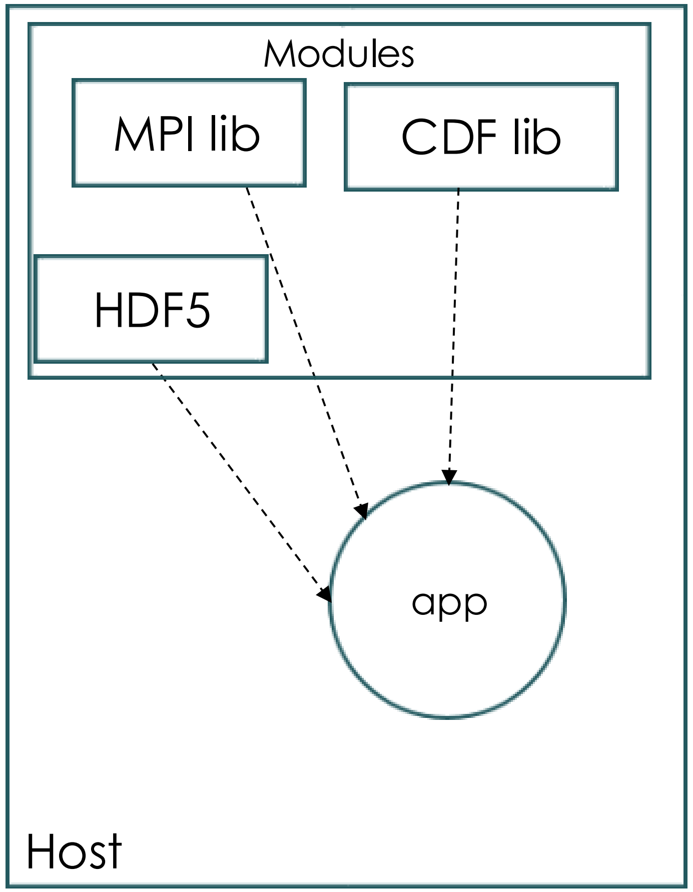
*Provided they're the same CPU and GPU architectures (e.g. Nvidia won't work on AMD GPUs)

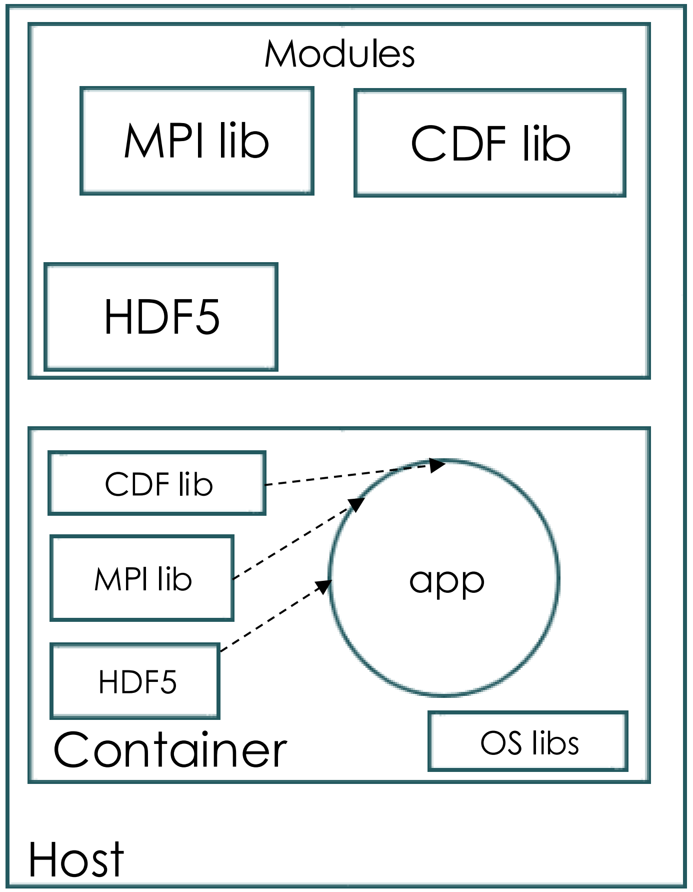
The use of Containers (In the context of HPC systems)



```
module load mpich
module load netcdf
module load hdf5
```

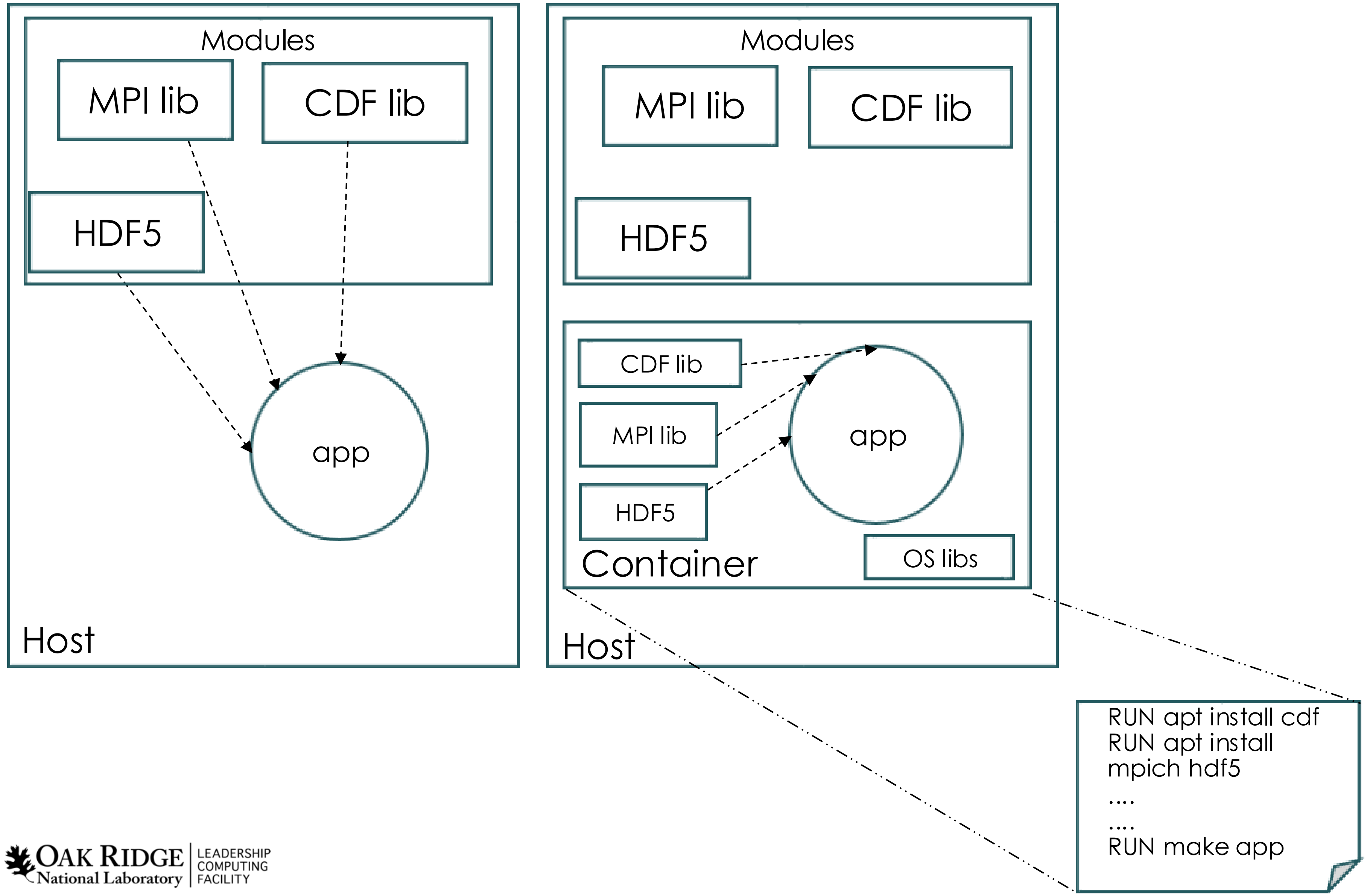
The use of Containers (In the context of HPC systems)





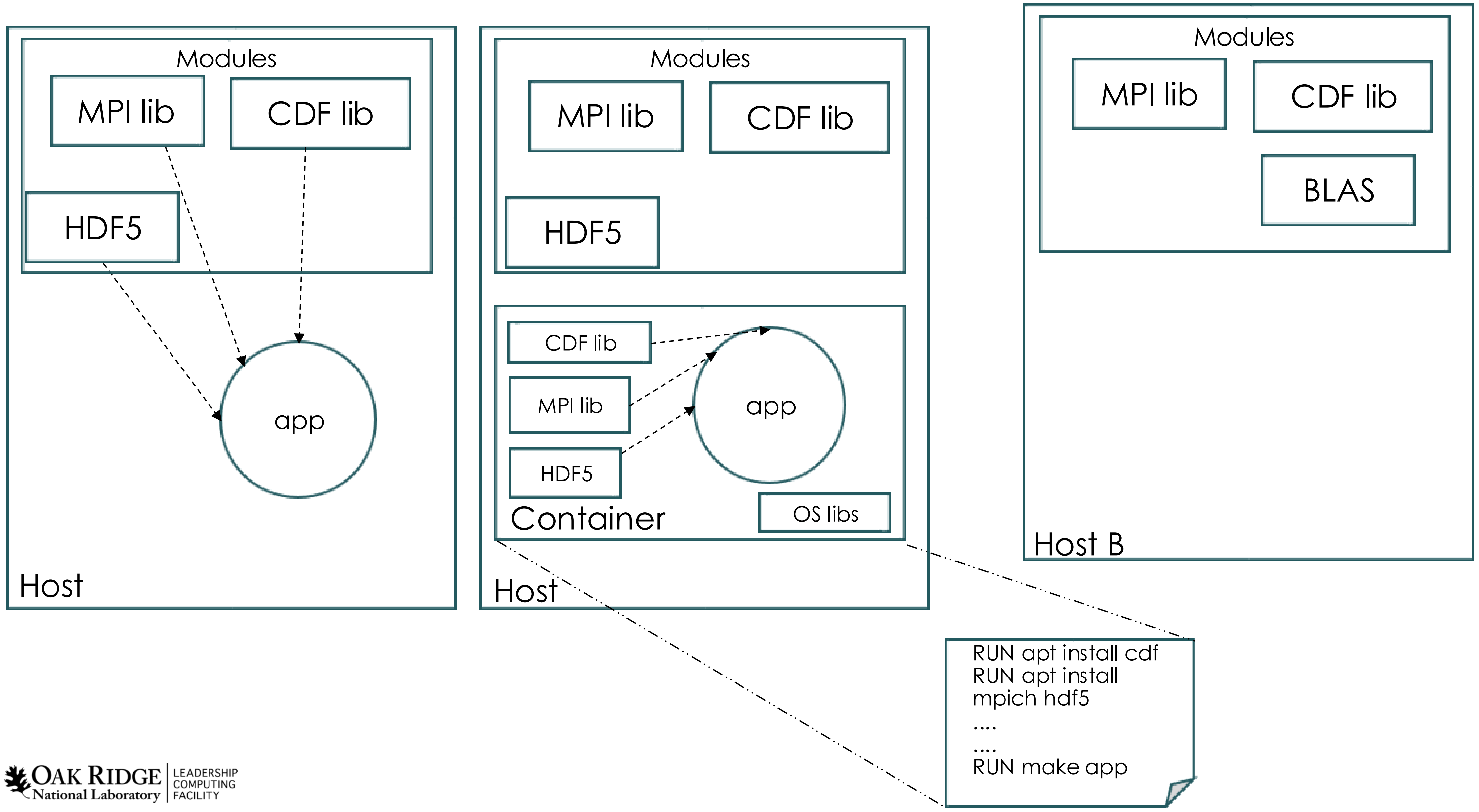
OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

The use of Containers (In the context of HPC systems)



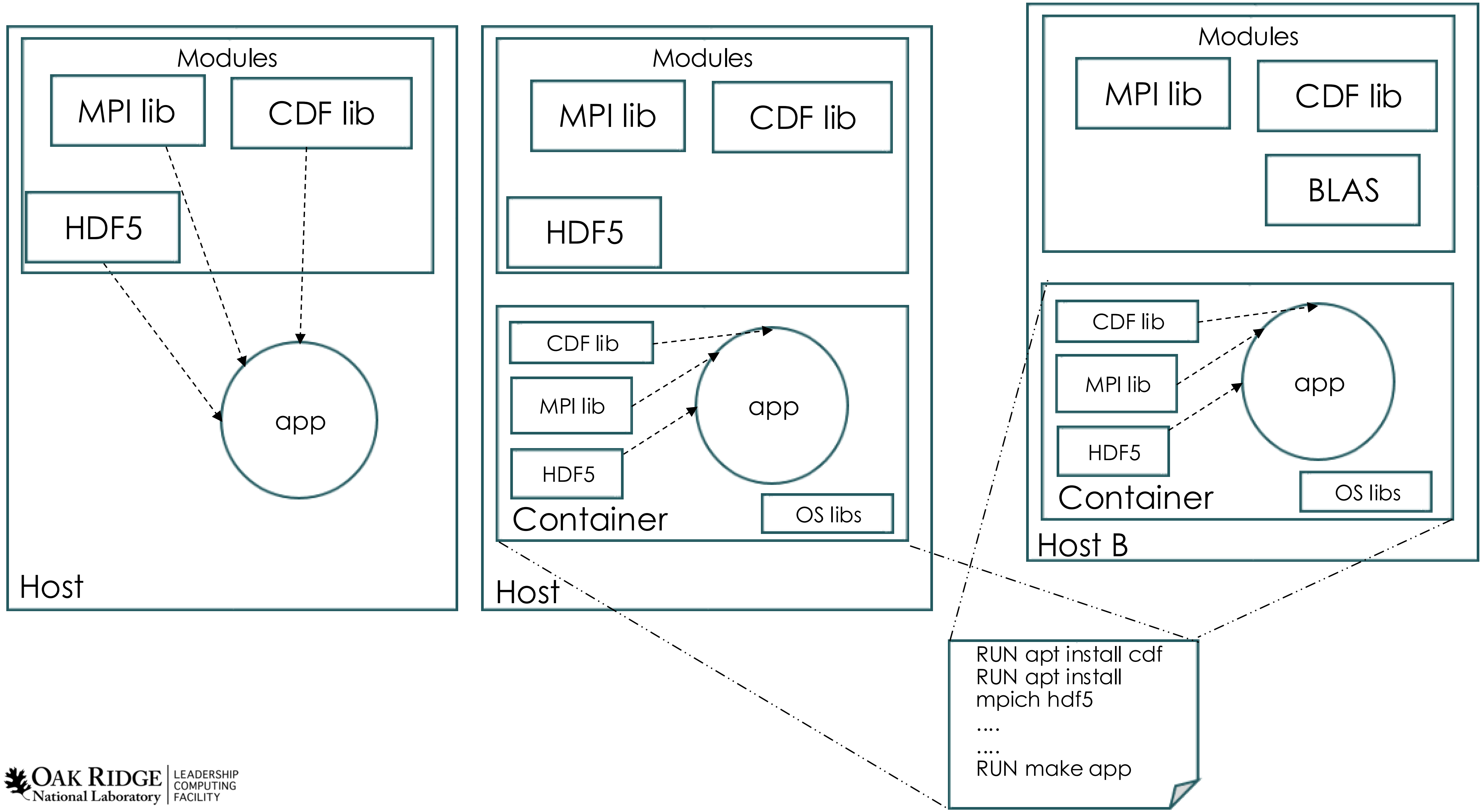
```
RUN apt install cdf
RUN apt install
mpich hdf5
....
....
RUN make app
```

The use of Containers (In the context of HPC systems)



```
RUN apt install cdf
RUN apt install
mpich hdf5
....
....
RUN make app
```

The use of Containers (In the context of HPC systems)



```
RUN apt install cdf
RUN apt install
mpich hdf5
....
....
RUN make app
```

Goal for OLCF

Empower user to build their own containers to run on Frontier. Each user's needs are different, so provide the building blocks to let users get themselves up and going and reach good performance.

Apptainer on Frontier

- Apptainer is a container image builder and the runtime
- Different from Docker and Podman, doesn't support the Containerfile format
  - But it can pull and convert Docker images to its format
- Supports MPI (through mounting host libraries) and AMD GPUs (with --rocm flag)
- Container image is a single file, can be stored anywhere on filesystem. Called SIF files (.sif extension)
- Frontier has Apptainer v1.3.2

Building a container with Apptainer

opensusebasic.def

```
Bootstrap: docker

From: opensuse/leap:15.4


%files

./hello.c /


%post

zypper --non-interactive --gpg-auto-import-keys refresh

zypper install -y  gzip gcc-c++ gcc-fortran

gcc -o /hello /hello.c
```

```
$ apptainer build opensusebasic.sif opensusebasic.def

$ apptainer exec opensusebasic.sif /hello

hello, world!

$ apptainer shell opensusebasic.sif

Apptainer> ls /

autofs  bin  boot  ccs  dev  environment  etc  hello
hello.c  home  host_lib64  lib  lib64  lustre  mnt  opt  proc
root  run  sbin  selinux  singularity  srv  sw  sys  tmp  usr
var
```

Building a container with Apptainer (with a local image)

```
Bootstrap: localimage

From: opensuseleap.sif


%files

./hello.c /


%post

zypper --non-interactive --gpg-auto-import-keys refresh

zypper install -y  gzip gcc-c++ gcc-fortran

gcc -o /hello /hello.c
```

```
$ apptainer pull opensuseleap.sif docker://docker.io/opensuse/leap:15.4

$ apptainer build opensusebasic.sif opensusebasic.def

$ apptainer exec opensusebasic.sif /hello

hello, world!

$ apptainer shell opensusebasic.sif

Apptainer> ls /

autofs  bin  boot  ccs  dev  environment  etc  hello
hello.c  home  host_lib64  lib  lib64  lustre  mnt  opt  proc
root  run  sbin  selinux  singularity  srv  sw  sys  tmp  usr
var
```

Pushing a container to Dockerhub with ORAS

```
$ apptainer registry login --username <your dockerhub username> oras://registry-1.docker.io

Password/Token: <enter your dockerhub password>

$ apptainer push opensusebasic.sif oras://registry-1.docker.io/<your docker username>/opensusebasic:latest

# NOTE: the above image will not work if you try to pull the image with Docker or Podman on another system.
Apptainer images are not compatible with Docker and Podman. It will only work with Apptainer.

$ rm opensusebasic.sif

$ apptainer pull opensusebasic.sif oras://docker.io/<your docker username>/opensusebasic:latest

# NOTE: If you get an error like:

FATAL:   While performing build: conveyor failed to get: while fetching library image: cached file hash(sha256:247d71…) and
expected hash(sha256:d0c012…) does not match
when pulling an Apptainer image from an ORAS registry, try building with apptainer pull --disable-cache …
```

Building a container with base image in ORAS

```
Bootstrap: oras

From: docker.io/<your docker username>/opensusebasic:latest


%post

zypper install -y hostname
```

```
$ apptainer build opensusehostname.sif opensusehostname.def

$ apptainer exec opensusehostname.sif hostname

login05
```

Apptainer definition files vs Docker/Podman containerfiles

```
Bootstrap: docker

From: opensuse/leap:15.4

%files

./hello.c /

%post

zypper --non-interactive --gpg-auto-import-keys refresh

zypper install -y  gzip gcc-c++ gcc-fortran

gcc -o /hello /hello.c
```

```
FROM opensuse/leap:15.4

COPY ./hello.c /hello.c

RUN zypper --non-interactive --gpg-auto-import-keys
refresh

RUN zypper install -y  gzip gcc-c++ gcc-fortran

RUN gcc -o /hello /hello.c
```

Filesystem access in container

- Several paths are automatically mounted into your container by default, including /lustre/orion and /ccs/home
- See /etc/apptainer/apptainer.conf on Frontier for the full list

```
$ apptainer exec \
            opensusehostname.sif ls /lustre/orion/stf007/

proj-shared
scratch
world-shared
```

Running your container with Slurm

```
#!/bin/bash

#SBATCH -A stf007uanofn
#SBATCH -J test
#SBATCH -N 4
#SBATCH -o subil_%j.out
#SBATCH -t 00:10:00

srun -N4 -n4 --tasks-per-node=1 apptainer exec \
            opensusehostname.sif hostname
```

```
Output:
frontier09298
frontier10202
frontier10196
frontier10233
```

Building container images with MPI+GPU applications

- Build your container with MPICH 3.4.2 or 3.4.3 and ROCm installed
  - MPICH 3.4.[2 | 3] is ABI compatible with Cray MPICH. Will let us use Cray MPICH during runtime.
- See https://github.com/olcf/olcf_containers_examples for example (opensusempich342rocm571.def under frontier->containers_on_frontier_docs->gpu_aware_mpi_example)

Running container with MPI+GPU application

- See https://github.com/olcf/olcf_containers_examples for example (submit.sbatch under frontier->containers_on_frontier_docs->gpu_aware_mpi_example)
- Let's walk through the example

Container modules

```
<load other modules>
module load olcf-container-tools
module load apptainer-enable-mpi
module load apptainer-enable-gpu
```

- The above modules sets the environment variables necessary for MPI and GPU support
- More convenient than explicitly setting the environment variables
- You can still set stuff like APPTAINERENV_LD_LIBRARY_PATH=blahblah, apptainer wrapper will prepend 'blahblah' to the rest of the mandatory entries in APPTAINERENV_LD_LIBRARY_PATH before running container
- Experimental. If you run into bugs, please email help@olcf.ornl.gov

Container modules

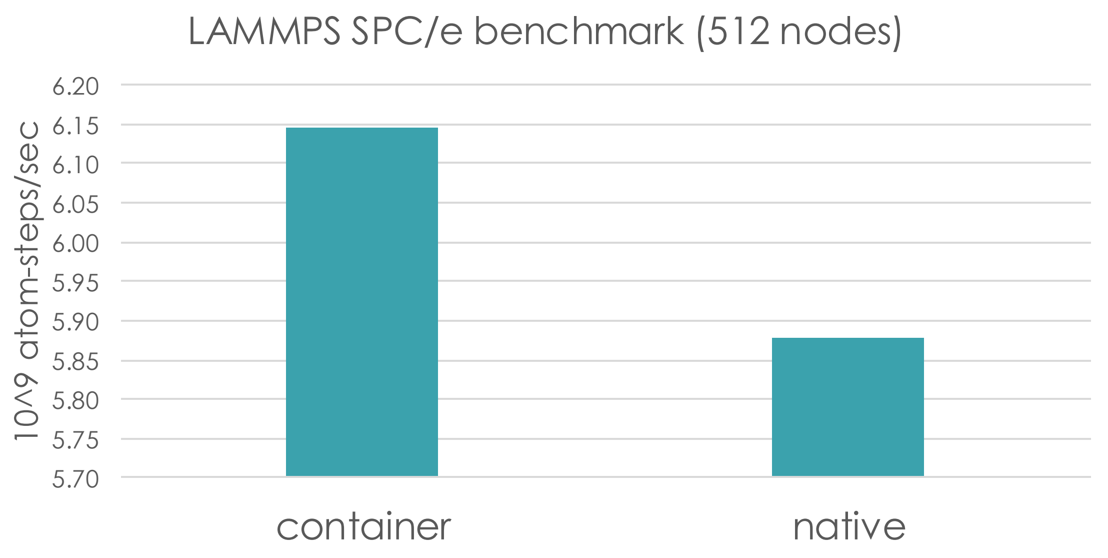
- You can still modify those APPTAINER* environment variables if you want to in your job script
  - the apptainer wrapper script set up by the module will automatically prepend your custom values to the APPTAINER* environment variables it sets.
  - So container will have values set by you and the module
- See submit.slurm in https://github.com/olcf/olcf_containers_examples (under frontier->containers_on_frontier_docs->apptainer_wrappers_lammps)

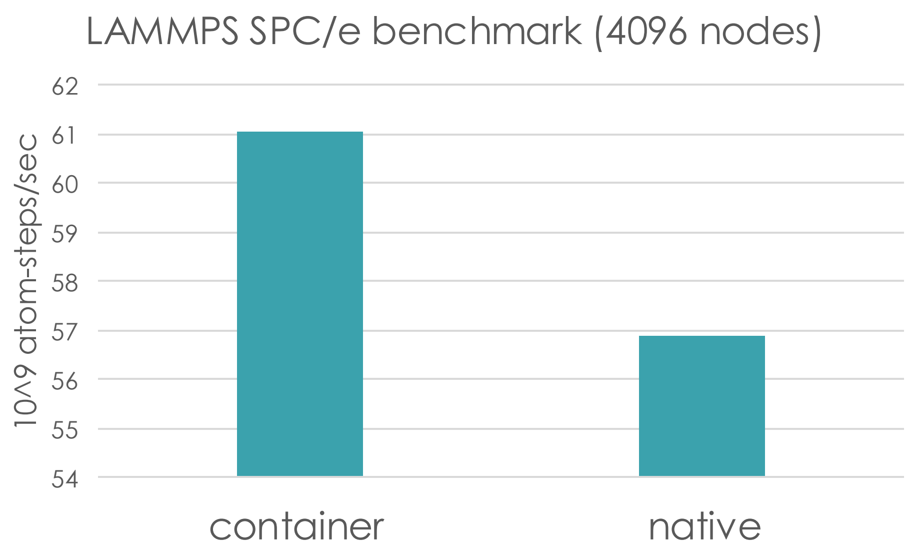
OLCF base images

- OLCF provides base images that matches software versions from CPE/23.12
  - These are **NOT** official Cray PE containers with Cray software. We try to match the software versions to match the indicated CPE/23.12.
  - They are SIF files, so use \`oras://\` or \`Bootstrap: oras\` to retrieve the base image
- Provides a suitable starting point to build your own containers
- See docs
- See example def files in https://github.com/olcf/olcf_containers_examples (Under frontier->containers_on_frontier_docs->apptainer_wrappers_lammps)

No performance impact

- Running LAMMPS OLCF-6 benchmark





*higher is better

If you find consistent performance degradations for your particular code, please let us know. We would be interested in documenting.

Apptainer on Andes

- Apptainer is available and installed on Andes.
- We don't have Andes specific docs yet for Apptainer, but feel free to experiment.

Links and Resources

- Apptainer documentation: https://apptainer.org/docs/user/main/index.html
- Containers on Frontier documentation: https://docs.olcf.ornl.gov/software/containers_on_frontier.html
- Container examples (which the docs reference): https://github.com/olcf/olcf_containers_examples/

Greatest Acknowledgments

- Asa Rentschler
- Elijah MacCarthy