JEDI Container and Cloud Platforms
Current Status

JCSDA: Mark Miesch, David Hahn, Dan Holdaway, Steve Herbener, Francois Vandenberghe, Xin Zhang, Tom Auligne, Yannick Tremolet + JEDI core team

AWS: Kevin Jorissen, Karthik Raman

S4: Scott Nolin, Jesse Stroik

NCCS: Kenny Peck, Nick Acks

Thanks also to the Singularity community (particularly David Trugdian, Vanessa Sochat, Bennet Fauber) for great support

Outline

I) JEDI Portability Overview

- ✦ Types of containers
- ✦ Container usage
  - CI/CD, development, HPC…

II) HPC SuperContainers

- ✦ Construction
- ✦ Usage
- ✦ Benchmarking

III) JEDI on AWS

- ✦ single-node
- ✦ cluster

IV) Summary & Outlook

- ✦ Status of Singularity

Software container (working definition)

A packaged user environment that can be “unpacked” and used across different systems, from laptops to cloud to HPC

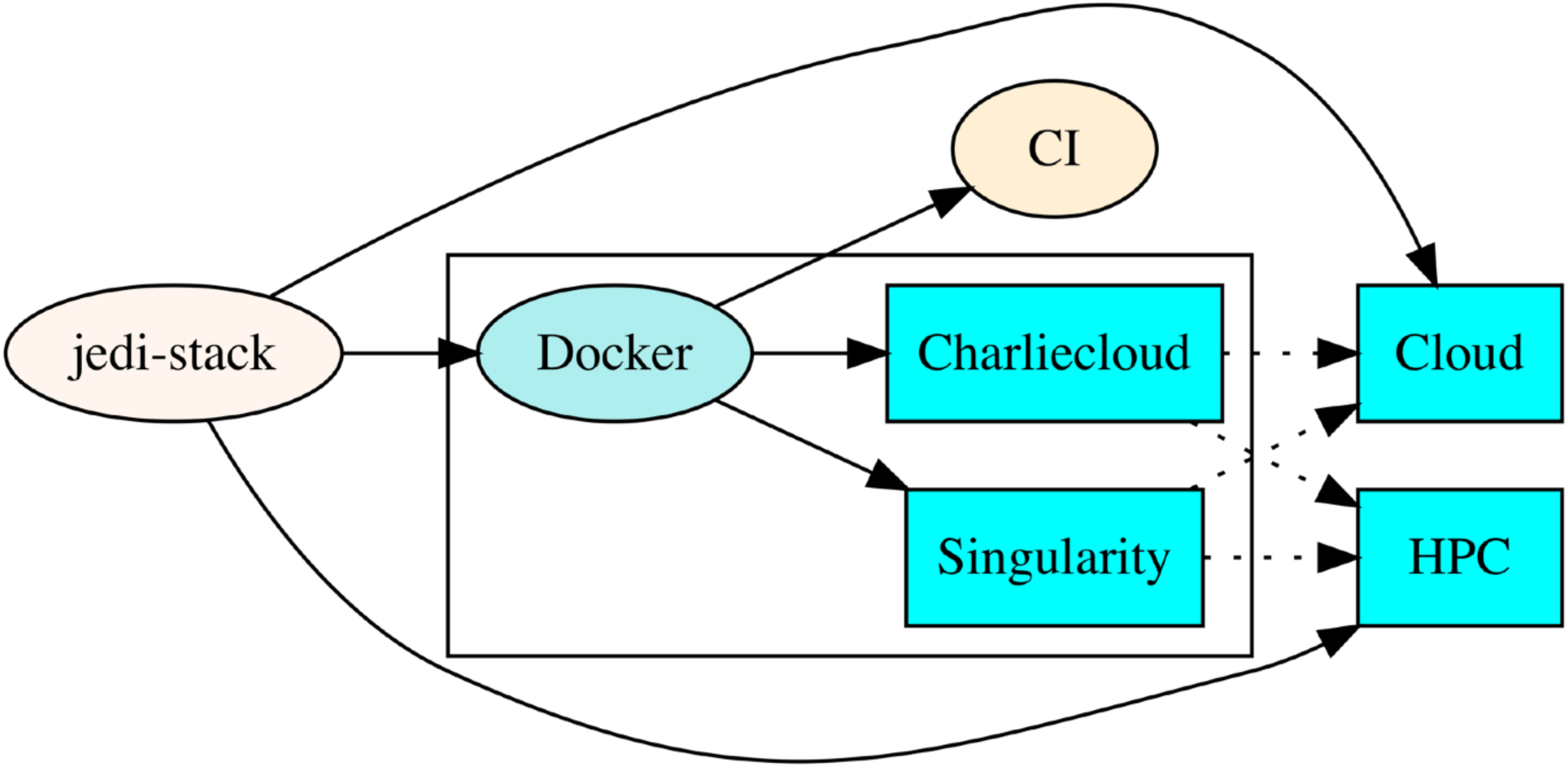
Container Providers

- **Docker**
  - ✦ Main Advantages: industry standard, widely supported, runs on native Mac/Windows OS
  - ✦ Main Disadvantange: Security (root privileges)

- **Singularity**
  - ✦ Main Advantages: Reproducibility, HPC support
  - ✦ Main Disadvantage: Not available on all HPC systems

- **Charliecloud**
  - ✦ Main Advantages: Simplicity, no need for root privileges
  - ✦ Main Disadvantages: Fewer features than Singularity, Relies on Docker (to build, not to run)

Unified Build System



Tagged jedi-stack releases -> tagged containers, AMIs, and HPC environment modules, ensuring common software environments across platforms

Container Types

- **Development**
  - **Contains**: Compilers, dependencies
  - **Omits:** JEDI code
  - **Used for:** CI/CD, Development, Optimization
- **Application**
  - **Contains:** Compiled JEDI bundles, runtime dependencies
  - **Omits:** Compilers, compile-time dependencies
  - **Used for:** Run JEDI releases across platforms
- **Tutorial**
  - **Contains:** Compilers, dependencies, JEDI source code, compiled JEDI bundles, run scripts, input files
  - Writable
  - **Used for:** Online tutorials, JEDI Academies

Current containers

- Development
  - gnu-openmpi-dev (D, S, C)
  - clang-mpich-dev (D, S, C)
  - intel19-impi-dev (D, S, C)
- Application
  - intel19-impi-app (S ⇒ S)
- Tutorial
  - gnu-openmpi-tut (⇒ D, S ⇒ S)

Distribution

Docker Hub

Sylabs cloud

AWS S3 (public)

AWS S3 (private)

II: SuperContainers

HPC Supercontainers are application containers that are designed to be used across multiple nodes on HPC systems, including cloud-based clusters

- Singularity
- Intel runtime libraries (multi-stage build)
- fv3-bundle (currently)
- Enhanced components
  - Infiniband drivers (Mellanox or linux inbox OFED)
  - PMI (PMI0 and/or slurm PMI2)
  - UCX and components
    - KNEM, XPMEM
- Built with NVIDIA's HPC-container-maker (hpccm)
  - https://github.com/jcsda/containers

SuperContainers Usage

Executed in multi-container (hybrid) mode to exploit system MPI configuration

mpiexec -np 864 <...> singularity exec <container> <application>

- **Each MPI task launches its own container**
- **MPI inside & outside container must be compatible**
- **Must take measures to avoid conflicts between host & container environment**

Contrast with the solo-container/single-node mode typically used for development containers

SuperContainers Usage

Portion of a modulefile used to eliminate host/container environment conflicts on S4 & AWS

```
setenv("SINGULARITYENV_PATH","/opt/jedi/bin:/opt/intel/psxe_runtime_2020.0.8/linux/mpi/intel64/libfabric/bin:/opt/
setenv("SINGULARITYENV_CPATH","/opt/jedi/include:/opt/intel/psxe_runtime_2020.0.8/linux/daal/include:/opt/intel/ps
setenv("SINGULARITYENV_LD_LIBRARY_PATH","/opt/jedi/lib:/opt/intel/psxe_runtime_2020.0.8/linux/daal/lib/intel64_lin
setenv("SINGULARITYENV_LIBRARY_PATH","/opt/jedi/lib:/opt/intel/psxe_runtime_2020.0.8/linux/daal/lib/intel64_lin:/
setenv("SINGULARITYENV_CLASSPATH","/opt/intel/psxe_runtime_2020.0.8/linux/daal/lib/daal.jar:/opt/intel/psxe_runtim
setenv("SINGULARITYENV_DAALROOT","/opt/intel/psxe_runtime_2020.0.8/linux/daal")
setenv("SINGULARITYENV_FI_PROVIDER_PATH","/opt/intel/psxe_runtime_2020.0.8/linux/mpi/intel64/libfabric/lib/prov")
setenv("SINGULARITYENV_IPPROOT","/opt/intel/psxe_runtime_2020.0.8/linux/ipp")
setenv("SINGULARITYENV_I_MPI_ROOT","/opt/intel/psxe_runtime_2020.0.8/linux/mpi")
setenv("SINGULARITYENV_MANPATH","/opt/intel/psxe_runtime_2020.0.8/linux/mpi/man:/usr/local/man:/usr/local/share/ma
setenv("SINGULARITYENV_MIC_LD_LIBRARY_PATH","/opt/intel/psxe_runtime_2020.0.8/linux/compiler/lib/intel64_lin_mic")
setenv("SINGULARITYENV_MKLROOT","/opt/intel/psxe_runtime_2020.0.8/linux/mkl")
setenv("SINGULARITYENV_PKG_CONFIG_PATH","/opt/intel/psxe_runtime_2020.0.8/linux/mkl/bin/pkgconfig")
setenv("SINGULARITYENV_PYTHONPATH","/usr/local/lib:")
setenv("SINGULARITYENV_TBBROOT","/opt/intel/psxe_runtime_2020.0.8/linux/tbb")

unsetenv("I_MPI_TMPDIR")
unsetenv("I_MPI_DIR")
unsetenv("I_MPI_LIB")
unsetenv("I_MPI_LIBRARY_KIND")
unsetenv("I_MPI_LINK")
unsetenv("I_MPI_DAPL_UD")
unsetenv("I_MPI_CC")
unsetenv("I_MPI_CXX")
unsetenv("I_MPI_F90")
unsetenv("I_MPI_F77")
unsetenv("I_MPI_INC")
unsetenv("I_MPI_ROOT")
unsetenv("I_MPI_PMI_LIBRARY")

setenv("SLURM_MPI_TYPE","pmi2")

whatis("Name: ".. pkgName)
whatis("Category: Application")
whatis("Environment variables for multinode Singularity applications")
```

SuperContainers Usage

slurm batch script for AWS (with container)

```
#!/bin/bash
#SBATCH --job-name=con1
#SBATCH --ntasks=864
#SBATCH --cpus-per-task=1
#SBATCH --time=1:00:00
#SBATCH --mail-user=miesch@ucar.edu

source /usr/share/modules/init/bash
module purge
module use /home/ubuntu/runs/Hofx_benchmark/modulefiles
module load intelmpi/2019.6.166
module load singularityvars
module list

ulimit -s unlimited
ulimit -v unlimited

export I_MPI_DEBUG=5
export I_MPI_FABRICS=shm:ofi
export I_MPI_OFI_PROVIDER=efa

export SLURM_EXPORT_ENV=ALL
export OMP_NUM_THREADS=1

JEDICON=/home/ubuntu
JEDIBIN=/opt/jedi/fv3-bundle/build/bin

cd /home/ubuntu/runs/Hofx_benchmark/conpc

mpiexec -np 864 singularity exec --home=$PWD $JEDICON/jedi-intel19-impi-hpc-app.sif ${JEDIBIN}/fv3jedi_var.x Config/3dvar_new.yaml

exit 0
```

JEDI Benchmarking

Benchmark FV3-GFS JEDI 3DVar Application

- Resolution c192
- ~9 of 12 million obs pass QC
- Inner loop: 30 iterations
- Outer loop: 2 iterations
- 864 MPI tasks (12x12x6)

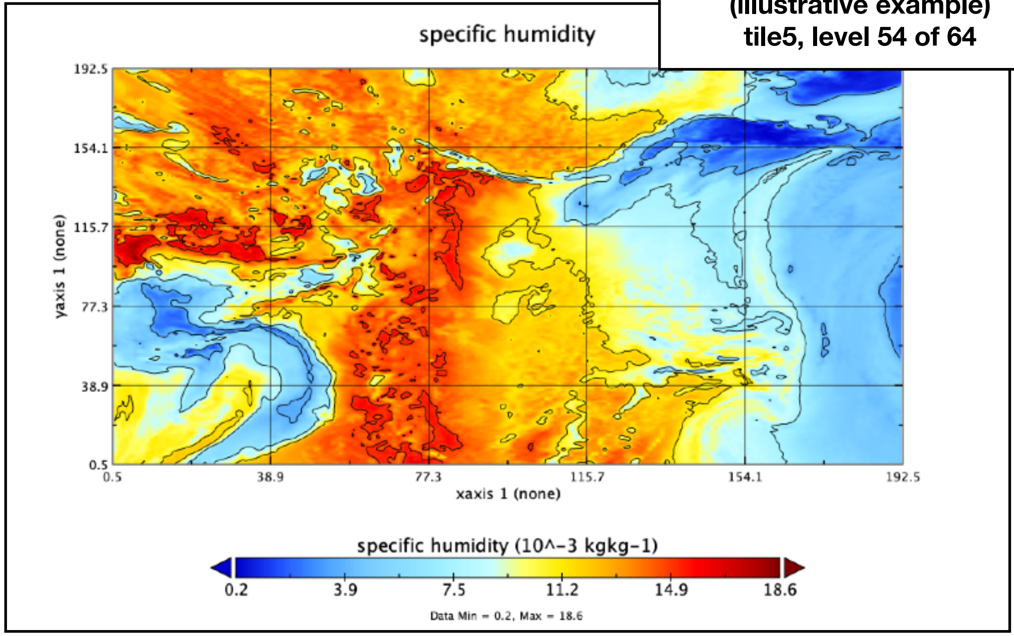
~12 million obs

Aircraft, Radiosonde, Rass, Satwind, Scatwind, Vadwind, AMSUA-NOAA19, AIRS-AQUA, IASI-METOPA, CRISFSR-NPP

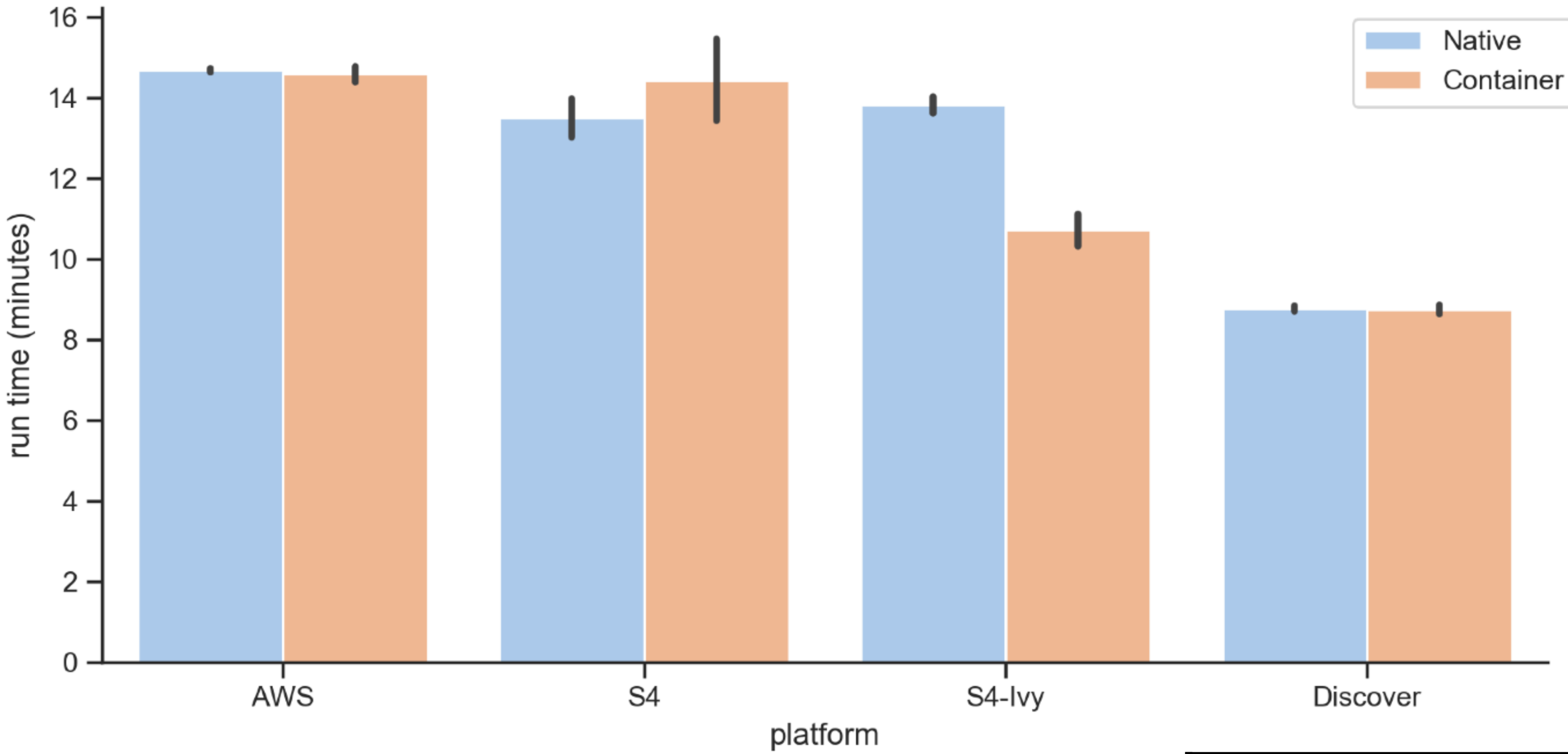
Platforms

- Discover: NASA NCCS
- S4: SSEC/Univ. Wisconsin
- AWS
    - 24 c5n.18xlarge nodes
    - 36 cores/node
    - Elastic Fabric Adapter (EFA)

Analysis
(illustrative example)
tile5, level 54 of 64



Container Benchmarking



Estimated AWS cost	
On demand	$23
Spot	$7

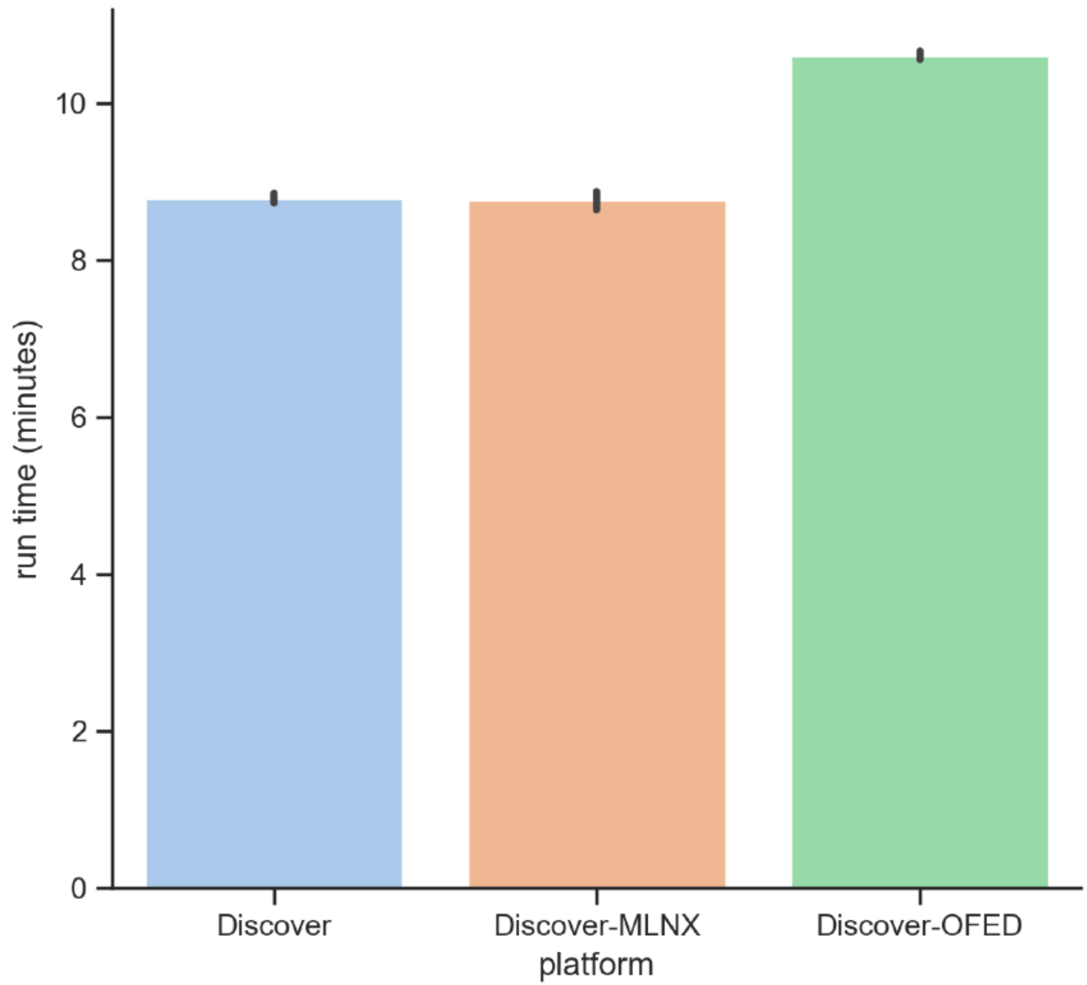
No overhead for running in the container

Container Benchmarking

Full disclosure - that wasn't actually the same container running on Discover

Replacing generic OFED infiniband drivers with Mellanox drivers needed to achieve native performance

It should be possible to include both but not yet tested



Container Benchmarking

More tips/tricks for Discover

- Since singularity was configured with an unprivileged (non-setuid) installation mode, the container image must be converted first to a sandbox directory and then the container must be run from the sandbox

singularity build --sandbox jedi-intel19-impi-hpc-app-sandbox/ jedi-intel19-impi-hpc-app.sif

- It is necessary to use mpiexec instead of mpirun

III: JEDI on AWS

- **Single Development node**
  - For development, optimization…
  - jedinode.py

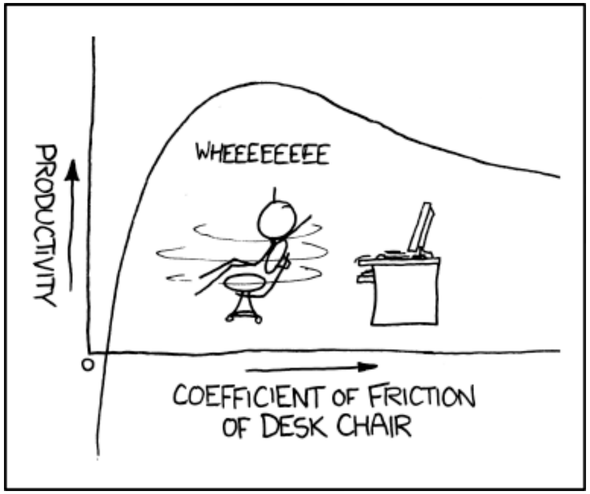
- **Cluster**
  - For applications, optimization, testing…
  - AWS ParallelCluster

See current PRs in jedi-tools, jedi-docs

Unified approach to facilitate maintenance: Intel compilers and environment modules (gnu-openmpi, intel-impi) provided by means of an external volume that is auto-mounted at boot time

III: JEDI on AWS

- **Single Development node**
  - ✦ Easy to use
  - ✦ Can terminate/stop from EC2 console
  - ✦ Custom AMI
  - ✦ Intel 19 compilers/mpi
  - ✦ gnu-openmpi, intel-impi stacks
  - ✦ Docker, Singularity, Charliecloud




```
18:34 $ jedinode.py --help
Usage: jedinode.py [OPTIONS]

Options:
  --key TEXT                ssh key   [required]
  --type TEXT               Instance type (default c5.4xlarge)
  --ncores INTEGER          Number of cores (you can omit this for most instance
                            types)
  --securitygroup TEXT      Security group id (default is virginia-default)
  --region TEXT             Region (default is us-east-1)
  --spot                    spot market (default is False)
  --maxprice TEXT           Max Price (defaults to on-demand price; only used if
                            spot is set)
  -h, --help                Show this message and exit.
```

III: JEDI on AWS

- **ParallelCluster**
  - Autoscaling: cluster size adjusts on demand
  - EFS, FSx for lustre
  - Intel 19 compilers/mpi
  - gnu-openmpi, intel-impi stacks
  - AWS-provided AMI; security patches, latest hardware support
  - Post-install script: Singularity, git-lfs…
  - Spot pricing or on demand
  - VPC (public master, private compute nodes) with subnets in us-east-1c (best availablity)
  - Dynamic placement group for collocated resources

IV: Summary & Outlook

- **Containers**
  - Development, Application, Tutorial
  - Great for getting up and running fast with JEDI without sacrificing performance
  - Key for our public releases

- **Supercontainers**
  - Not plug and play - takes a little fiddling to get good performance
  - Can run multi-node HPC applications with no overhead

- **AWS**
  - Use jedinode.py for single devel nodes
  - Use ParallelCluster for multi-node clusters

Singularity outlook

Singularity was developed by Greg Kurtzer and colleagues at Lawrence Berkeley Lab (2015). In 2017, Kurtzer formed a new company called Sylabs to provide Singularity and related services, such as the Sylabs cloud container repository.

May, 2020: Kurtzer (Sylabs CEO) announced that he will be "starting a new company [hpcng], which will leverage Singularity as a foundational building block". Greg will remain as the Singularity Open Source project lead and the new company with be a "non-commercialized", open source, "community-focused GitHub organization"

https://github.com/hpcng

HPCNG = "The Next Generation of High Performance Computing"

Singularity outlook: MacOS

Good news! Singularity Desktop for Mac exists in a beta version!

https://sylabs.io/singularity-desktop-macos/

Bad news: I haven't been able to get it to work yet with the jedi containers

More bad news: Unclear if users will have sufficient numbers/experience/inclination to maintain Singularity for Mac as a community project

More bad news: Mac as a platform might become more difficult in the future with Apple's recent announcement to move away from x86 architectures to ARM. Maintaining containers for two platforms might become a challenge

Extra slides

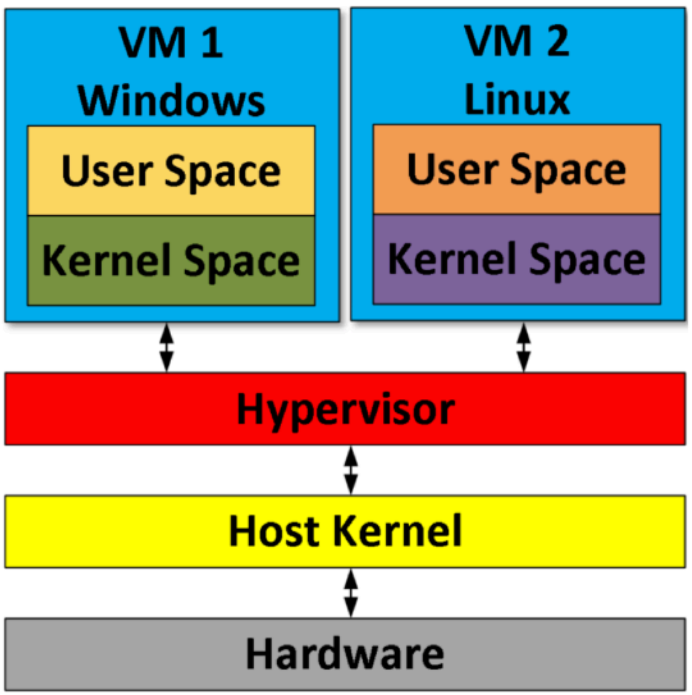
JEDI Software Dependencies

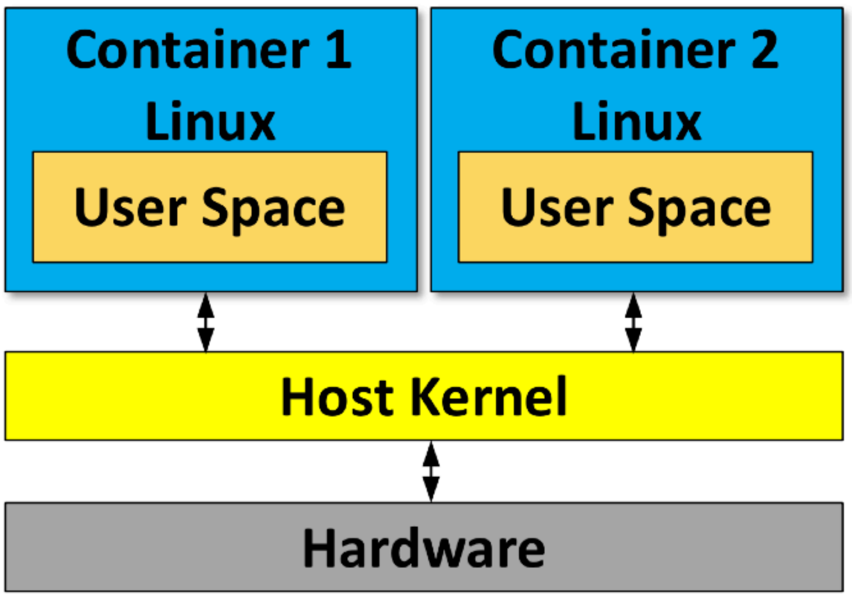
- Essential
  - Compilers, MPI
  - CMake
  - SZIP, ZLIB
  - LAPACK / MKL, Eigen 3
  - NetCDF4, HDF5
  - udunits
  - Boost (headers only)
  - ecbuild, eckit, fckit

Common versions among users and developers minimize stack-related debugging

- Useful
  - ODB-API, eccodes
  - PNETCDF
  - Parallel IO
  - nccmp, NCO
  - Python tools (py-ncepbufr, netcdf4, matplotlib…)
  - NCEP libs
  - Debuggers & Profilers (kdbg, valgrind, TAU…)

Containers vs Virtual Machines





Containers work with the host system
Including access to your home directory

Julio Suarez

Container Technologies

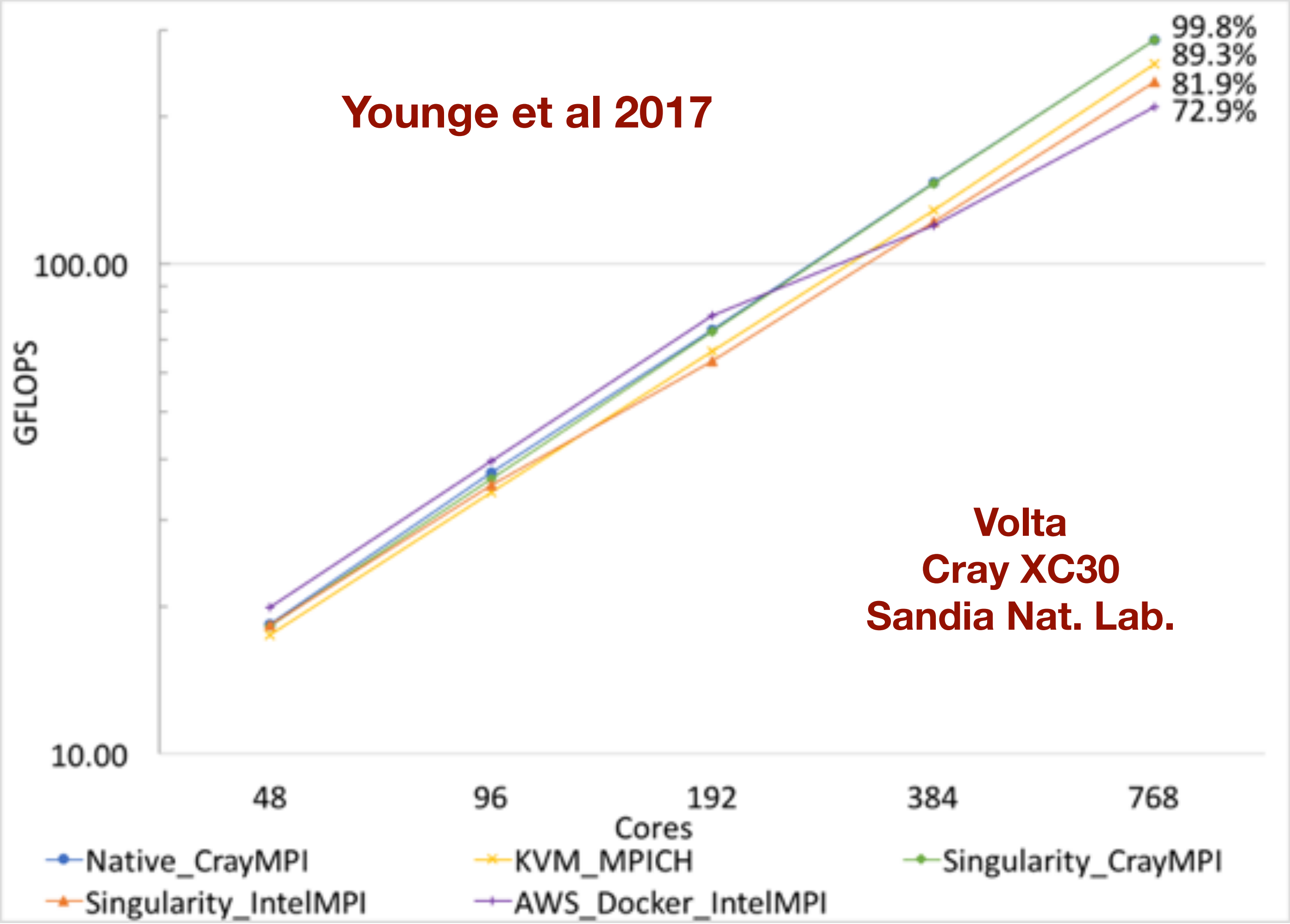
Kurtzer, Sochat & Bauer (2017)

Table 1. Container comparison.

	Singularity	Shifter	Charlie Cloud	Docker
Privilege model	SUID/UserNS	SUID	UserNS	Root Daemon
Supports current production Linux distros	Yes	Yes	No	No
Internal image build/bootstrap	Yes	No*	No*	No***
No privileged or trusted daemons	Yes	Yes	Yes	No
No additional network configurations	Yes	Yes	Yes	No
No additional hardware	Yes	Maybe	Yes	Maybe
Access to host filesystem	Yes	Yes	Yes	Yes**
Native support for GPU	Yes	No	No	No
Native support for InfiniBand	Yes	Yes	Yes	Yes
Native support for MPI	Yes	Yes	Yes	Yes
Works with all schedulers	Yes	No	Yes	No
Designed for general scientific use cases	Yes	Yes	No	No
Contained environment has correct perms	Yes	Yes	No	Yes
Containers are portable, unmodified by use	Yes	No	No	No
Trivial HPC install (one package, zero conf)	Yes	No	Yes	Yes
Admins can control and limit capabilities	Yes	Yes	No	No

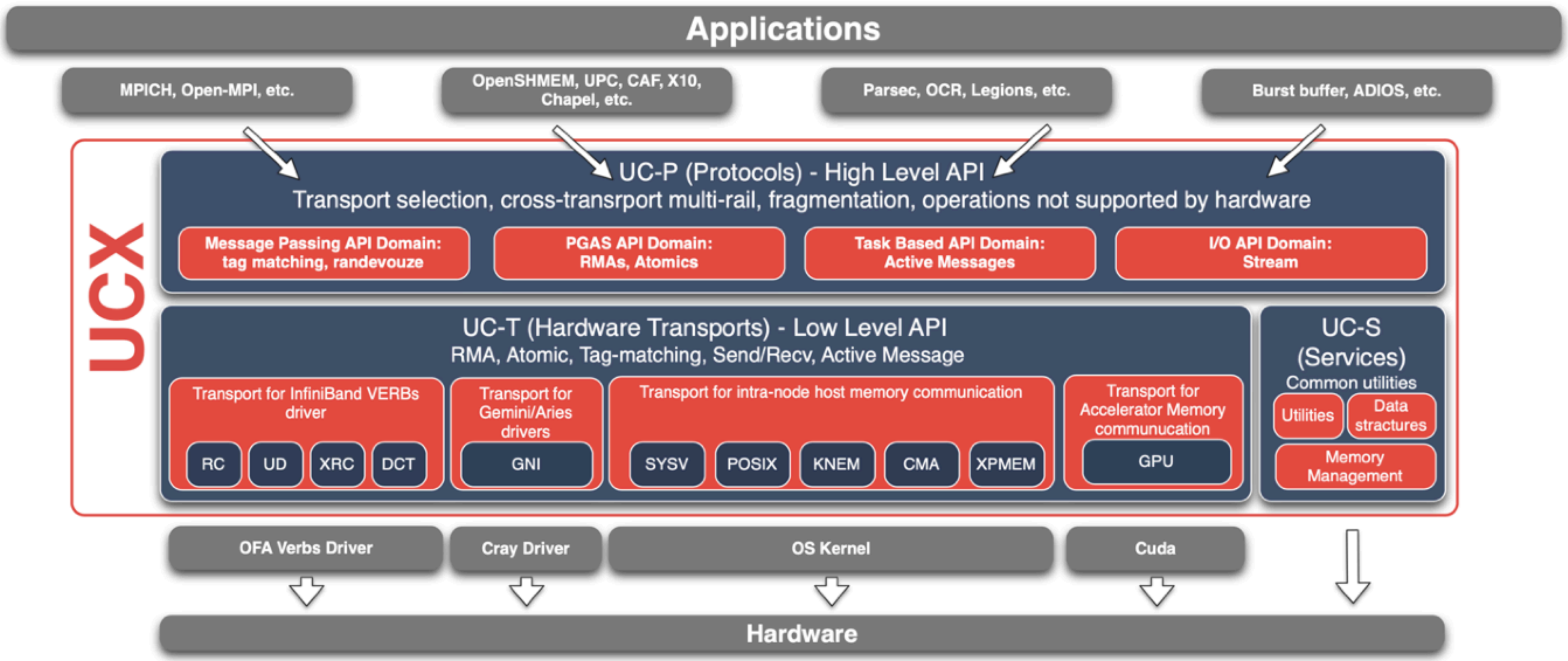
This is why we will continue to support all three (Docker, Singularity, Charliecloud)

Containers can achieve near-native performance (negligible overhead) but only if you tap into the native MPI libraries



HPC containers promising, but currently not "plug and play"

Fig. 1: UCX Architecture



Shamis et al 2015