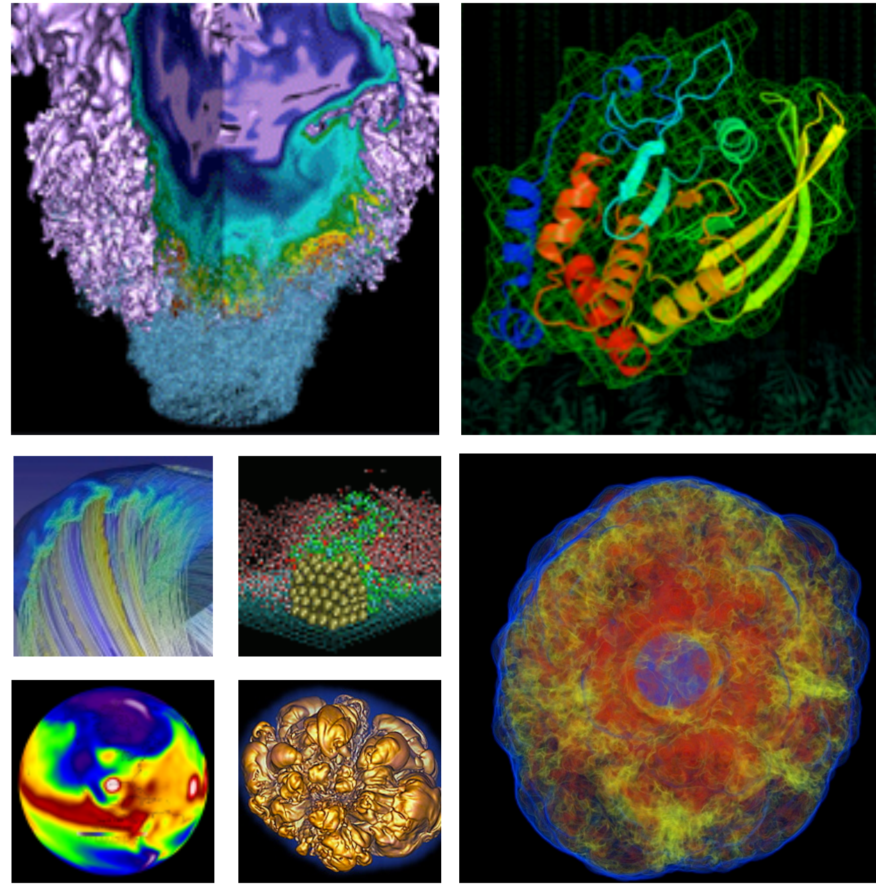
Never Port Your Code Again – Docker functionality with Shifter using SLURM



Douglas Jacobsen, James Botts, Shane Canon
NERSC

September 15, 2015
SLURM User Group

Acknowledgements

- Larry Pezzaglia (Former NERSC, UDI COE, CHOS)
- Scott Burrow (NERSC, Jesup/MyDock)
- Shreyas Cholia (NERSC, MyDock)
- Dave Henseler (Cray, UDI COE)
- John Dykstra (Cray, UDI COE)
- Charles Shirron (Cray, UDI COE)
- Michael Barton (JGI, Example Docker Image)
- Lisa Gerhardt (NERSC, HEP cvmfs work)
- Debbie Bard (NERSC, LSST testing)

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research of the U.S. Department of Energy under Contract No. **DE-AC02-05CH11231**.

User Defined Images/Containers in HPC

- **Data Intensive computing often require complex software stacks**
- **Efficiently supporting “big software” in HPC environments offers many challenges**
- **Shifter**
  - NERSC R&D effort, in collaboration with Cray, to support User-defined, user-provided Application images
  - “Docker-like” functionality on the Cray and HPC Linux clusters
  - Efficient job-start & Native application performance

Glossary

- **Image**: operating environment for a software stack including “operating system” files, e.g., /etc, /usr, …
- **User Defined Image:** an image that a user creates for their needs in particular
- **Linux Container:** an instance of an application image and its running processes
- **Docker:** a software system for creating images and instantiating them in containers. Enables distribution of images through well-defined web APIs
  - Repository/repo – online store of related Docker images
  - Layer – collection of files
  - Image – ordered collection of layers denoting a particular version within repo
  - Tag – a label or pointer to a version (e.g., “latest” points to most recent version; “15.04” points to the relevant image)
- **Shifter:** software for securely enabling Docker (and other images) to run in a limited form of Linux Container in HPC and cluster environments, specifically for HPC use-cases

Convergence of Disruptive Technology

- **Increasing Role of Data**

- **Converging HPC and Data Platforms**

- **New Models of Application Development and Delivery**

DOE Facilities Require Exascale Computing and Data

Astronomy

Particle Physics

Chemistry and Materials

Genomics

Fusion

Petascale to Exascale

- **Petabyte data sets today, many growing exponentially**
- **Processing requirements grow super-linearly**
- **Need to move entire DOE workload to Exascale**

Popular features of a data intensive system and supporting them on Cori

Data Intensive Workload Need	Cori Solution
Local Disk	NVRAM 'burst buffer'
Large memory nodes	128 GB/node on Haswell; Option to purchase fat (1TB) login node
Massive serial jobs	Shared-Node/Serial queue on cori via SLURM
Complex workflows	***User Defined Images*** CCM mode Large Capacity of interactive resources
Communicate with databases from compute nodes	Advanced Compute Gateway Node
Stream Data from observational facilities	Advanced Compute Gateway Node
Easy to customize environment	***User Defined Images***
Policy Flexibility	Improvements coming with Cori: Rolling upgrades, CCM, MAMU, above COEs would also contribute

User-Defined Images

- **User-Defined Images (UDI): A software framework which enables users to accompany applications with portable, customized OS environments**
  - e.g., include ubuntu base system with Application built for ubuntu (or debian, centos, etc)
- **A UDI framework would:**
  - Enable the HPC Platforms to run more applications
  - Increase flexibility for users
  - Facilitate reproducible results
  - Provide rich, portable environments without bloating the base system

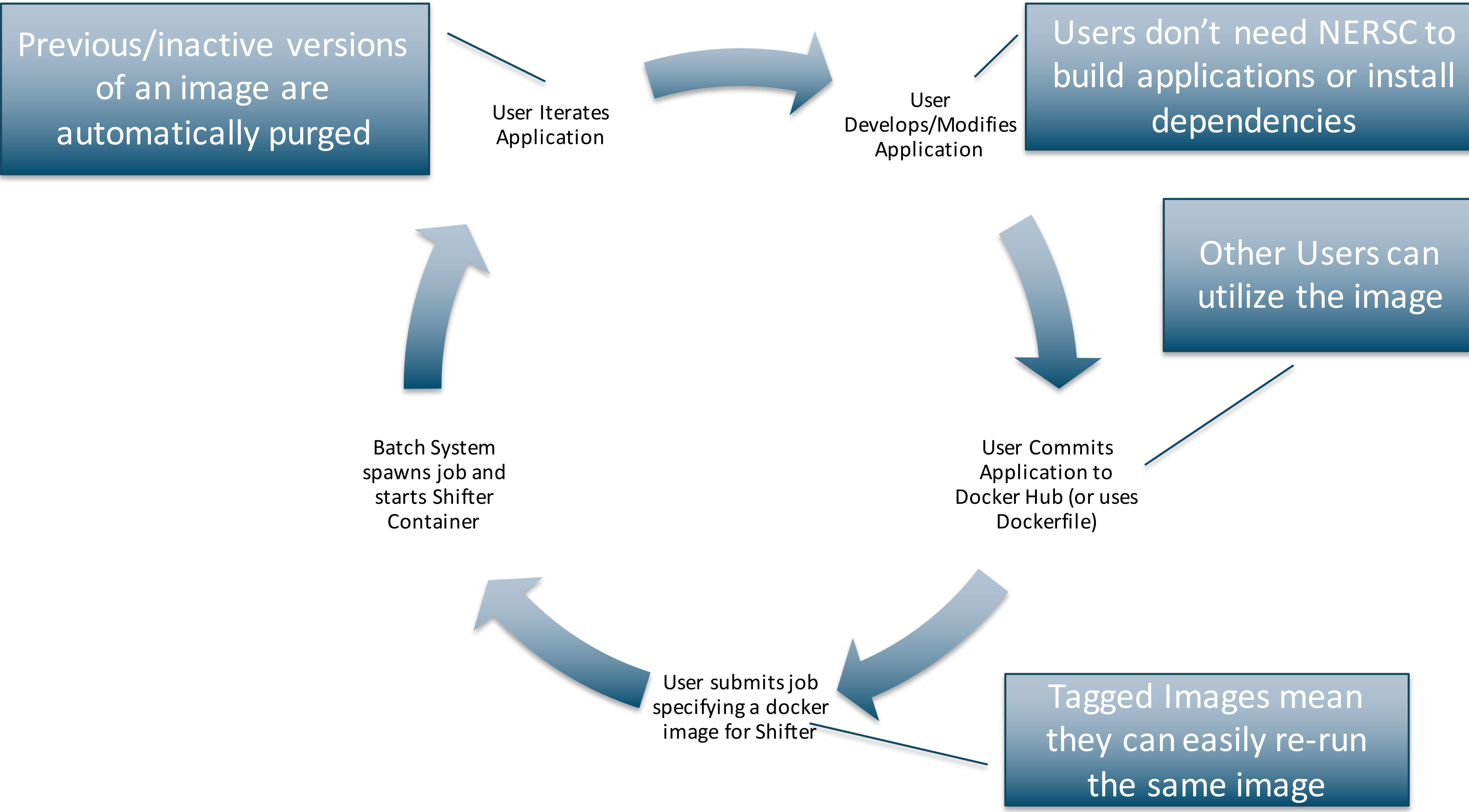
Use Cases

- **Large high energy physics collaborations (e.g., ATLAS and STAR) requiring validated software stacks**
  - Some collaborations will not accept results from non-validated stacks
  - Simultaneously satisfying compatibility constraints for multiple projects is difficult
  - Solution: create images with certified stacks
- **Bioinformatics and cosmology applications with many third-party dependencies**
  - Installing and maintaining these dependencies is difficult
  - Solution: create images with dependencies
- **Seamless transition from desktop to supercomputer**
  - Users desire consistent environments
  - Solution: create an image and transfer it among machines

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Development Cycle



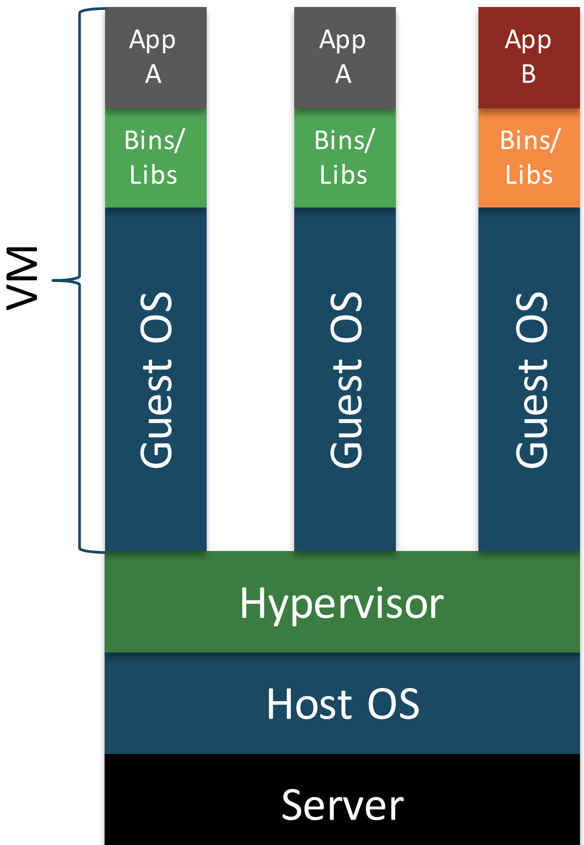
Value Proposition for UDI for Cori

- **Expanded application support**
  - Many applications currently relegated to commodity clusters could run on the Cray through UDI
  - This will help position Cori as a viable platform for data-intensive
- **Easier application porting**
  - The need for applications to be “ported” to Cori will be reduced
  - More applications can work “out of the box”
- **Better Reproducibility**
  - Easier to re-instantiate an older environment for reproducing results

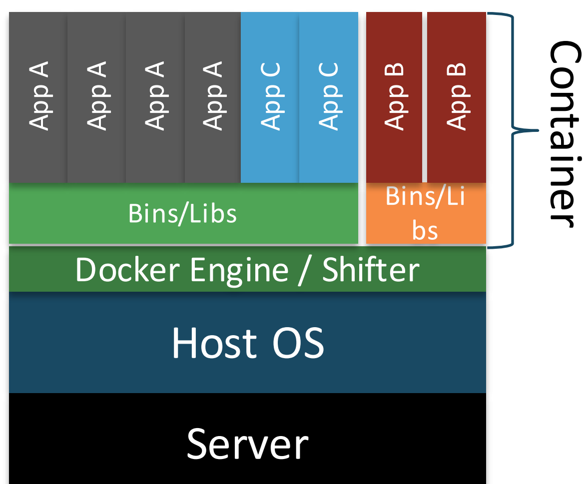
UDI Design Goals for Cori

- User independence: Require no administrator assistance to launch an application inside an image
- Shared resource availability (e.g., PFS/DVS mounts and network interfaces)
- Leverages or integrates with public image repos (i.e. DockerHub)
- Seamless user experience
- Robust and secure implementation
- Fast job startup time
- “native” application execution performance

Linux Containers vs. Virtual Machines



Containers are isolated, but share kernel and, where appropriate, bins/libraries



Containers provide close to native performance

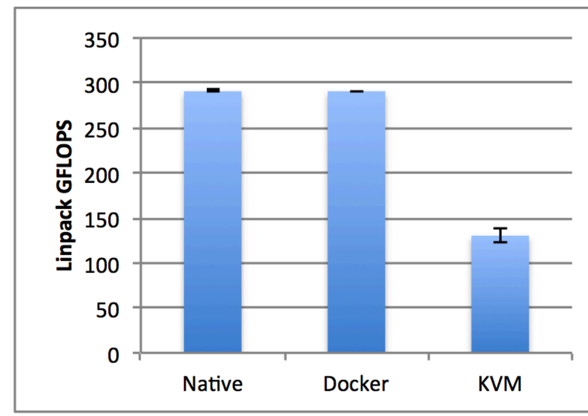


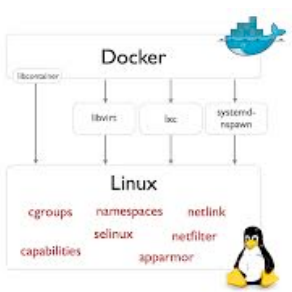
Figure 1. Linpack performance on two sockets (16 cores). Each data point is the arithmetic mean obtained from ten runs. Error bars indicate the standard deviation obtained over all runs.

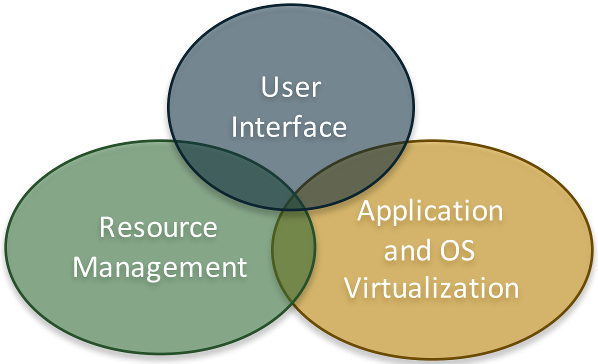
Source: IBM Research Report (RC25482)

A "container" delivers an application with all the libraries, environment, and dependencies needed to run.

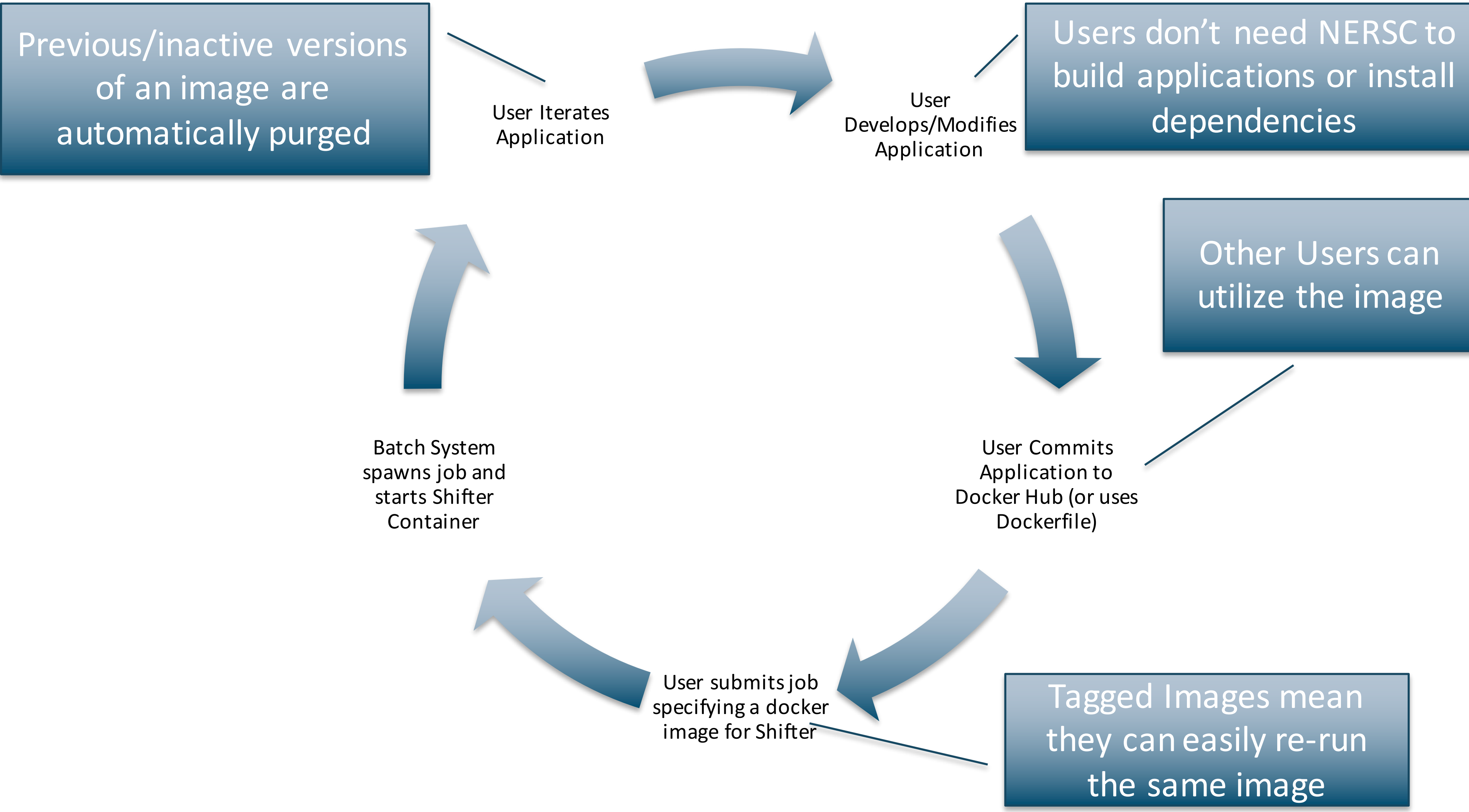
Docker at a High Level

- **Process Container: Uses Linux kernel features (cgroups and namespaces) to create semi-isolated "containers"**
- **Image Management: Version control style image management front-end and image building interface**
- **Ecosystem: Can push/pull images from a community-oriented image hub (i.e. DockerHub)**





Development Cycle



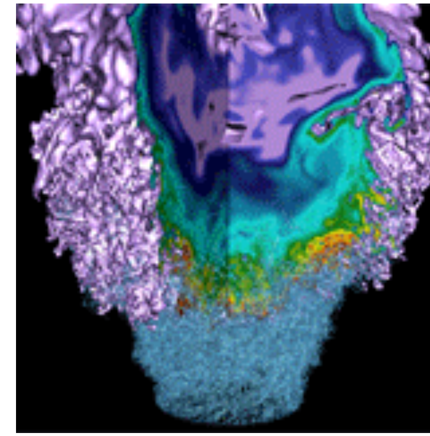
Parallel Python Dockerfile Example

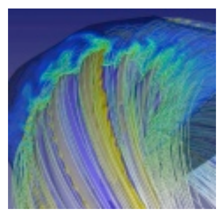
```
FROM python:2.7

## setup optimized numpy
ADD mkl.tar /
ADD numpy-1.9.2.tar.gz /usr/src/
ADD site.cfg /usr/src/numpy-1.9.2/
RUN cd /usr/src/numpy-1.9.2 && \
    chmod -R a+rX /opt/intel && \
    chown -R root:root /opt/intel && \
    python setup.py build && \
    python setup.py install && \
    cd / && rm -rf /usr/src/numpy-1.9.2 && \
    printf "/opt/intel/composer_xe_2015.1.133/mkl/lib/intel64\n" >> /etc/ld.so.conf && \
    ldconfig

ADD scipy-0.16.0b2.tar.gz /usr/src/
ADD site.cfg /usr/src/scipy-0.16.0b2/
RUN cd /usr/src/scipy-0.16.0b2 && \
    apt-get update && \
    apt-get dist-upgrade -y && \
    apt-get install gfortran -y && \
    python setup.py build && \
    python setup.py install && \
    cd / && rm -rf /usr/src/scipy-0.16.0b2

## install mpi4py
ADD mpi4py-1.3.1.tar.gz /usr/src/
ADD optcray_alva.tar /
ADD mpi.cfg /usr/src/mpi4py-1.3.1/
RUN cd /usr/src && \
    cd mpi4py-1.3.1 && \
    chmod -R a+rX /opt/cray && \
    chown -R root:root /opt/cray && \
    python setup.py build && \
    python setup.py install && \
    cd / && rm -rf /usr/src/mpi4py-1.3.1 && \
    printf "/opt/cray/mpt/default/gni/mpich2-
gnu/48/lib\n/opt/cray/pmi/default/lib64\n/opt/cray/ugni/default/lib64\n/opt/cray/udreg/default/lib64\n/opt/cray/xpmem/default/lib64\n/opt/cray/alps/d
efault/lib64\n/opt/cray/wlm_detect/default/lib64\n" >> /etc/ld.so.conf && \
    ldconfig
```

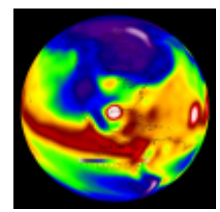
U.S. DEPARTMENT OF ENERGY | Office of Science

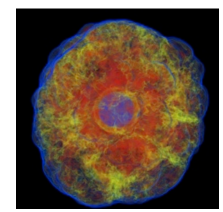
BERKELEY LAB

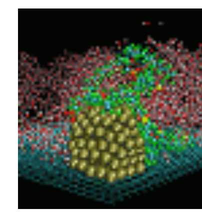
User Defined Images on the Cray Platforms











Docker on the Cray?

- **Docker assumes local disk**
  - aufs, btrfs demand shared-memory access to local disk

Approach

- Leverage Docker image and integrate with DockerHub
- Adopt alternate approach for instantiating the environment on the compute node (i.e. don’t use the Docker daemon)
- Plus: Easier to integrate and support
- Minus: Can’t easily leverage other parts of the Docker ecosystem (i.e. orchestration)

Image Location/Format Options

Image Location Options

- **GPFS** – No local clients, so overhead of DVS
- **Lustre** – Local clients, large scale
- **Burst Buffer** – Not widely available yet
- **Local Disk** – Not available on Crays

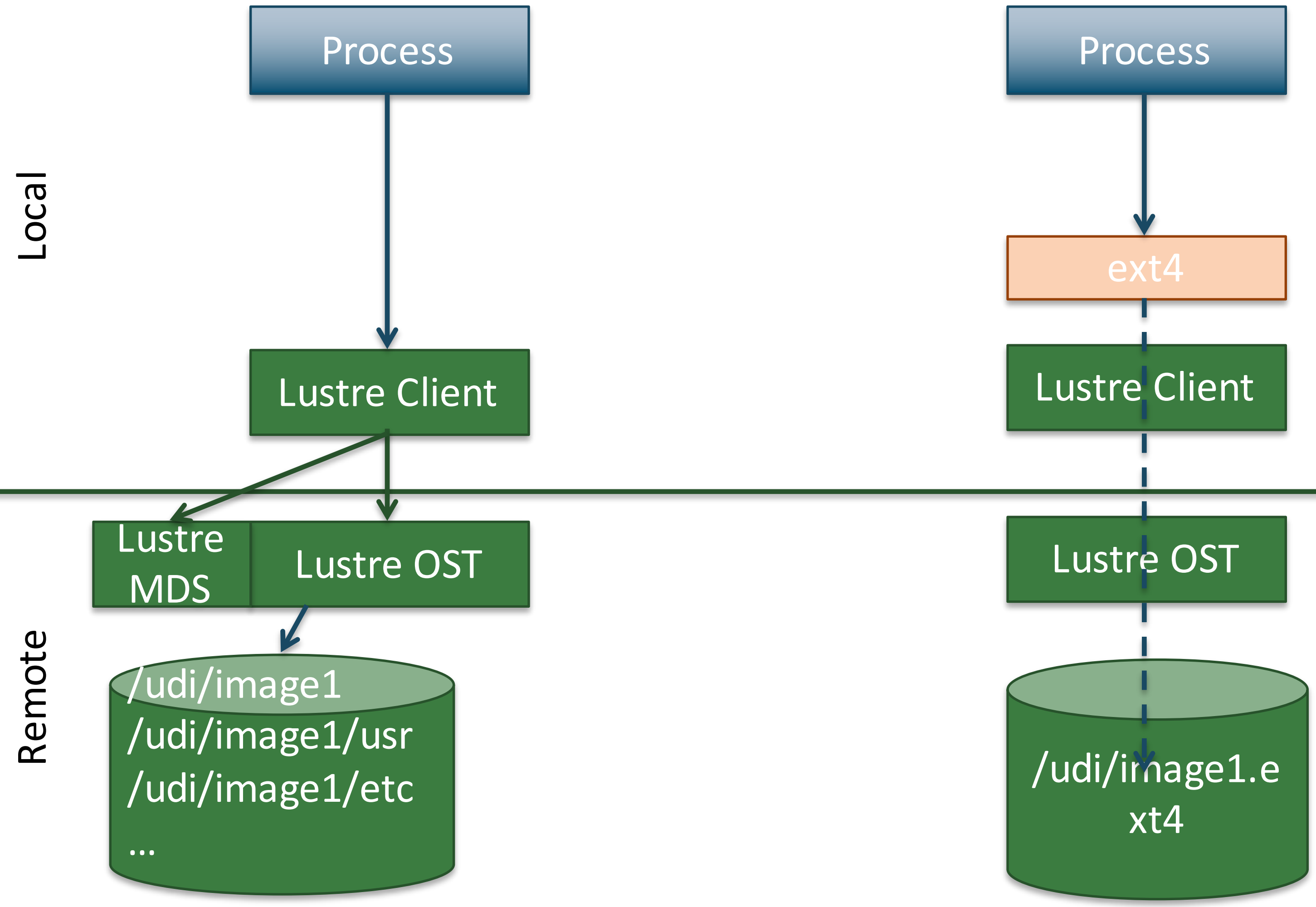
Image Format Options

- **Unpacked Trees**
  - Simple to implement
  - Metadata performance depends on metadata performance of the underlying system (i.e. Lustre or GPFS)
- **Loopback File Systems**
  - Moderate complexity
  - Keeps file system operations local

Considerations

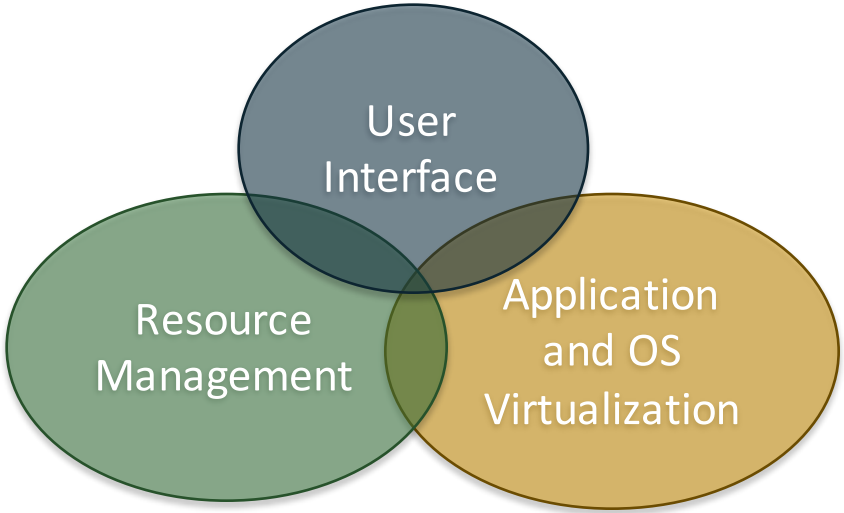
- Scalable
- Manageable
- Metadata performance
- Bandwidth Consumption

Layout Options – Stack View



Shifter

- **Supports**
  - Docker Images
  - CHOS Images
  - Can support other image types (e.g., qcow2, vmware, etc)
- **Basic Idea**
  - Convert native image format to common format (ext4, squashfs)
  - Construct chroot tree on compute nodes using common format image
  - Modify image within container to meet site security/policy needs
  - Directly use linux VFS namespaces to support multiple shifter containers on same compute node



Shifter

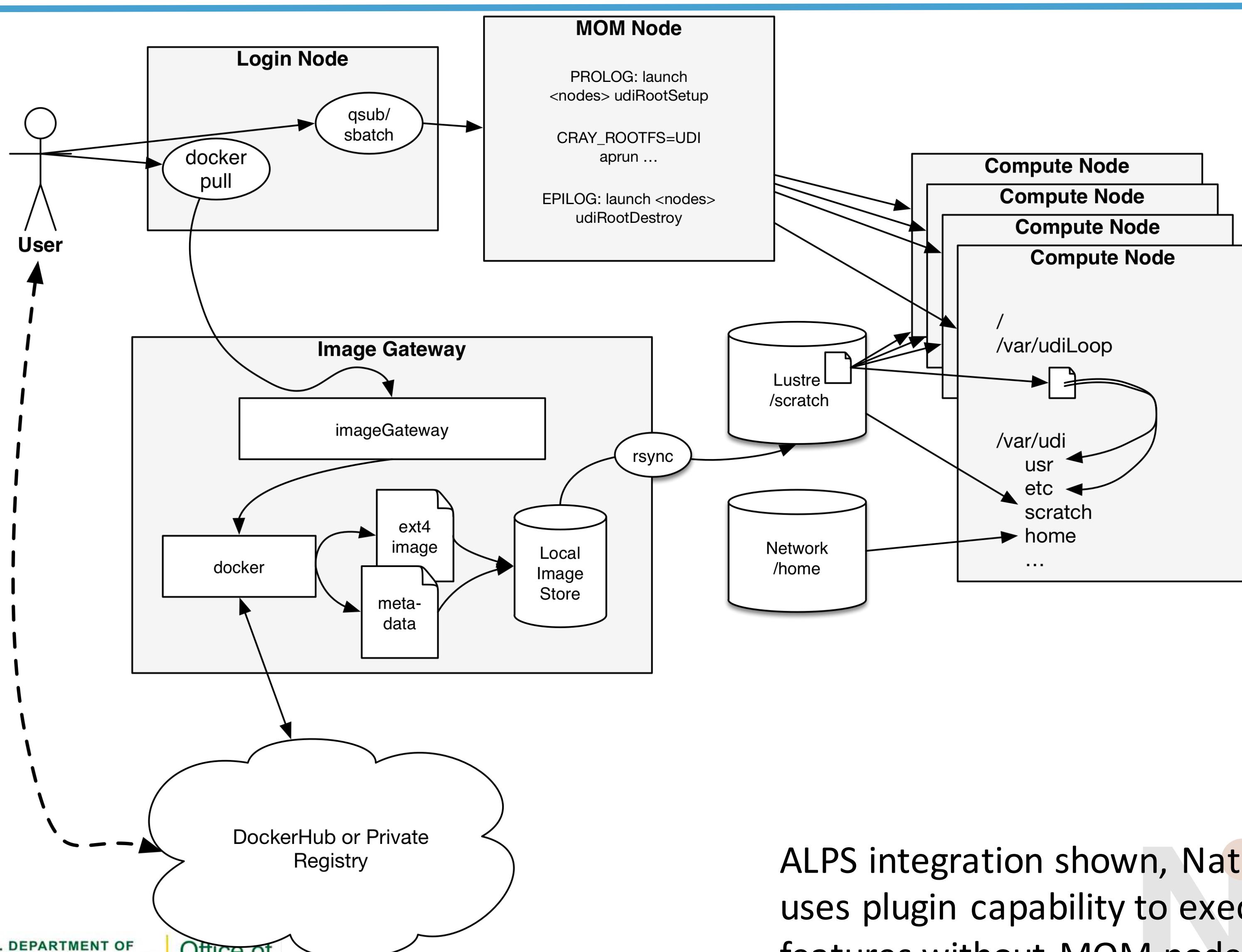
- **Command line interface**
  - Instantiate and control shifter containers interactively
  - List images
  - Pull image from DockerHub / private registry
- **Central gateway service**
  - Manage images
  - Transfer images to computational resource
  - Convert images several sources to common format
- **udiRoot**
  - Sets up container on compute node
  - CCM-like implementation to support internode communication
- **Workload Manager Integration**
  - Pull image at job submission time if it doesn't already exist
  - Implement user interface for user-specified volume mapping
  - Launch udiRoot on all compute nodes at job start
  - Deconstruct udiRoot on all compute nodes at job end
  - WLM provides resource management (e.g., cgroups)

User Interface

Application and OS Virtualization

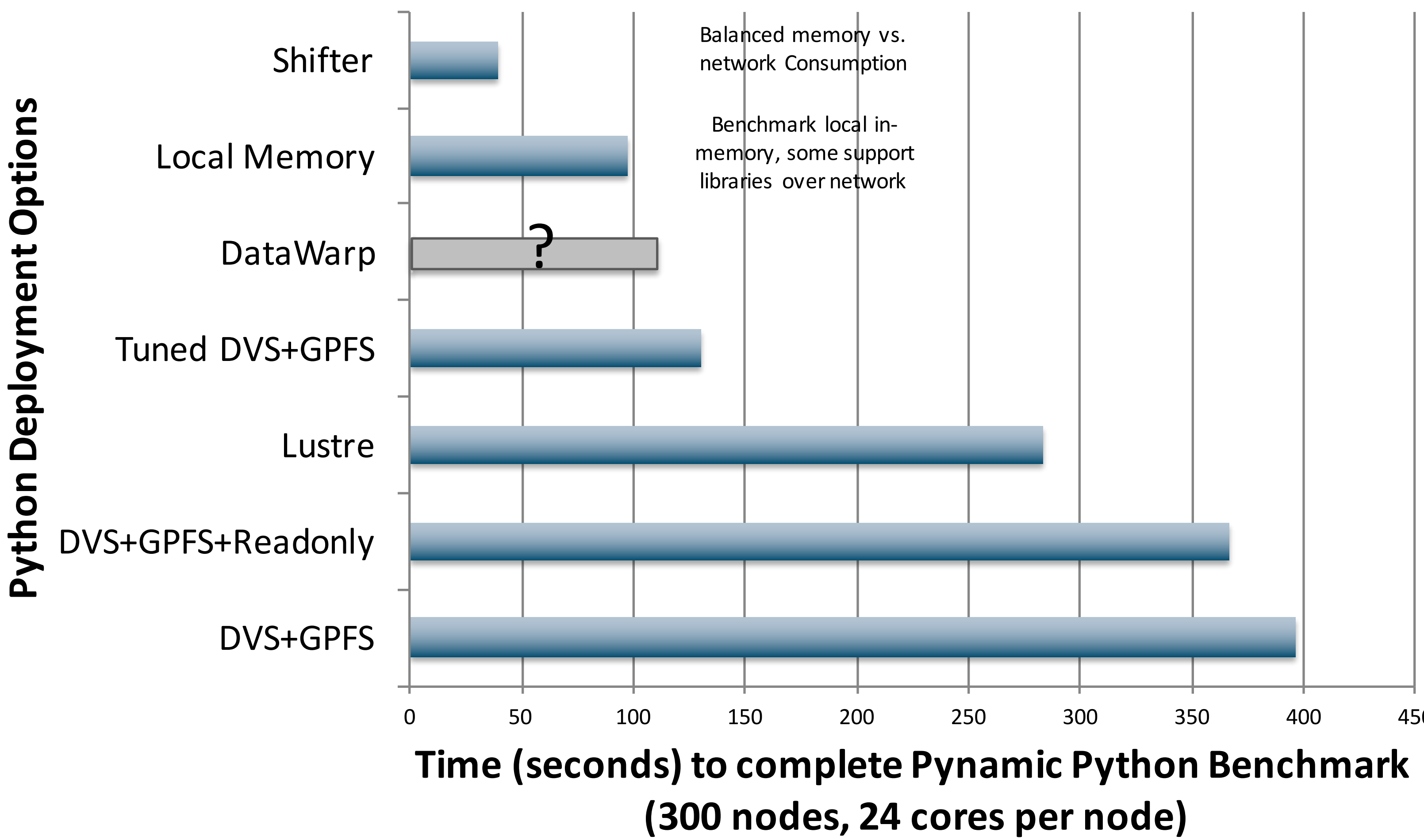
Resource Management

Shifter



ALPS integration shown, Native Slurm uses plugin capability to execute features without MOM node

Shifter Delivers Performance – Pynamic



U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

How is Shifter similar to Docker?

- **Sets up user-defined image under user control**
- **Allows volume remapping**
  - mount /a/b/c on /b/a/c in container
- **Containers can be "run"**
  - Environment variables, working directory, entrypoint scripts can be defined and run
- **Can instantiate multiple containers on same node**

How does Shifter differ from Docker?

- **User runs as the user in the container – not root**
- **Image modified at container construction time:**
  - Modifies /etc, /var, /opt
    - replaces /etc/passwd, /etc/group other files for site/security needs
    - adds /var/hostsfile to identify other nodes in the calculation (like $PBS_NODEFILE)
    - Injects some support software in /opt/udiImage
  - Adds mount points for parallel filesystems
    - Your homedir can stay the same inside and outside of the container
    - Site configurable
- **Image readonly on the Computational Platform**
  - to modify your image, push an update using Docker
- **Shifter only uses mount namespaces, not network or process namespaces**
  - Allows your application to leverage the HSN and more easily integrate with the system
- **Shifter does not use cgroups directly**
  - Allows the site workload manager (e.g., SLURM, Torque) to manage resources
- **Shifter uses individual compressed filesystem files to store images, not the Docker graph**
  - Uses more diskspace, but delivers high performance at scale
- **Shifter integrates with your Workload Manager**
  - Can instantiate container on thousands of nodes
  - Run parallel MPI jobs
- **Specialized sshd run within container for exclusive-node for non-native-MPI parallel jobs**
  - PBS_NODESFILE equivalent provided within container (/var/hostsfile)
  - Similar to Cray CCM functionality
  - Acts in place of CCM if shifter “image” is pointed to /dsl VFS tree

Shifter / SLURM Integration

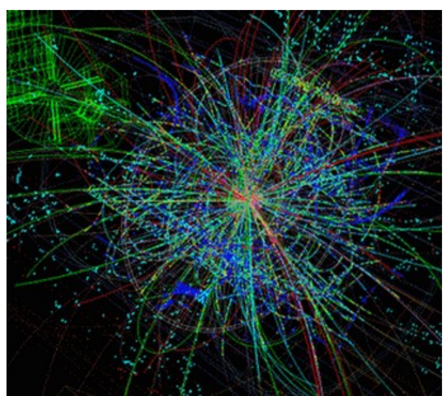
- **A custom SPANK plugin adds these options to salloc, sbatch, srun:**
  - --image=<image descriptor>
  - --imagevolume=<volume remapping descriptor>
  - --ccm (uses shifter to emulate Cray CCM)
  - --autoshift (automatically run batch script in shifter image)
- **slurm_spank_job_prolog**
  - Sets up shifter environment on all exclusive compute nodes in allocation (should use PrologFlags=Alloc)
- **slurm_spank_task_init_privileged**
  - Instantiate shifter container for slurmstepd for trusted images when ccm or autoshift is defined
- **slurm_spank_job_epilog**
  - Tear down shifter environment on all exclusive compute nodes in allocation
- **Other requirements:**
  - Should ensure that all needed cgroup paths, slurm state directories, and munge sockets are mounted in shifter containers (siteFs variable in udiRoot.conf)

Demo

Where are we now?

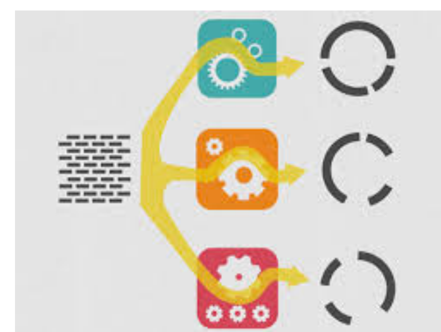
- An early version of Shifter is deployed on Edison. Early users are already reporting successes!
- Shifter is fully integrated with batch system, users can load a container on many nodes at job-start time. Full access to parallel FS and High Speed Interconnect (MPI)
- Shifter and NERSC were recently featured in HPC Wire and other media. Several other HPC sites have expressed interest.
- Our early users:

Light Sources Structural Biology (early production)



LHC – Nuclear Physics (testing)

Cosmology (testing)



nucleotid.es Genome Assembly (proof of concept)

Where are we going?

- **Cori phase 1 will debut with a new fully *open-source* version of Shifter**
  - Users can run multiple containers in a single job (i.e., dynamically select and run Shifter containers)
  - Tighter integration and simple user interface
- **Cray has indicated interest in making a supported product out of Shifter**
- **Ultimate Code Portability – Don't rewrite your code, shift the data center!**
  - Support for volume mappings
    - `/scratch2/scratchdirs/username/code1/date2/a/b/c` ➔ `/input`
    - `/scratch2/scratchdirs/username/code1/date2/out/1` ➔ `/output`
  - Support for entrypoint scripts
    - Autorun: `/usr/bin/myContainerScript.sh`
  - Entrypoints + volume mappings = No site-specific batch script or porting delays ➔ Increased scientific productivity

Thank you! (And we are hiring!)

NERSC

National Energy Research Scientific Computing Center

U.S. DEPARTMENT OF ENERGY | Office of Science

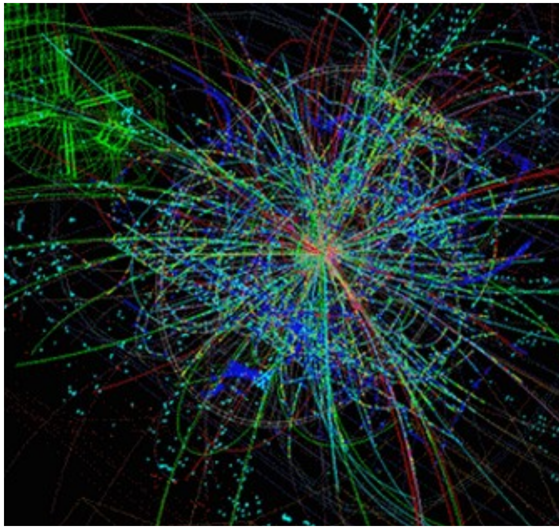
BERKELEY LAB

Shifter Security Model

- **User *only* accesses the container as their uid, never root or contextual root**
- **Generated site /etc/passwd, /etc/group (filtered) is placed in container**
  - no need for shifter containers to interoperate with LDAP or concerns about image PAM
- **Optional sshd run within image is statically linked, only accessible to that container's user**
- **All user-provided data (paths within image, environment variables, command line arguments) are filtered**
- **Executables that run with root privileges are implemented in C with only glibc dependencies**
- ***All* filesystems in container are remounted no-setuid**
- ***All* processes run with privilege carefully manage environment to prevent accidental/intentional manipulation**

Using Shifter to deliver cvmfs for LHC

- **Vast majority of LHC computing depends on cvmfs**
  - Network file system that relies on http to deliver full software environment to compute node
    - Needs fuse, root perms, and local disk so implementation on Cray systems has been difficult
    - Most groups extract needed portions of cvmfs and extract on compute node RAM disk – this breaks automated pipelines
- **Use shifter to deliver full ALICE, ATLAS, CMS cvmfs repositories to Edison**
  - ALICE image is 892 GB with 10,116,157 inodes, load up time was negligible
  - Ran simulations of p-Pb collisions at an energy of 5 TeV and reconstructed the result
  - Actual physics (used to evaluate the EMCal trigger), that worked out of the box
- **NERSC is making shifter images available for all cvmfs repositories**



Shifter

- **Deployed containers are read-only to the user**

Docker

- **Running containers are mutable**