BLUE WATERS

SUSTAINED PETASCALE COMPUTING

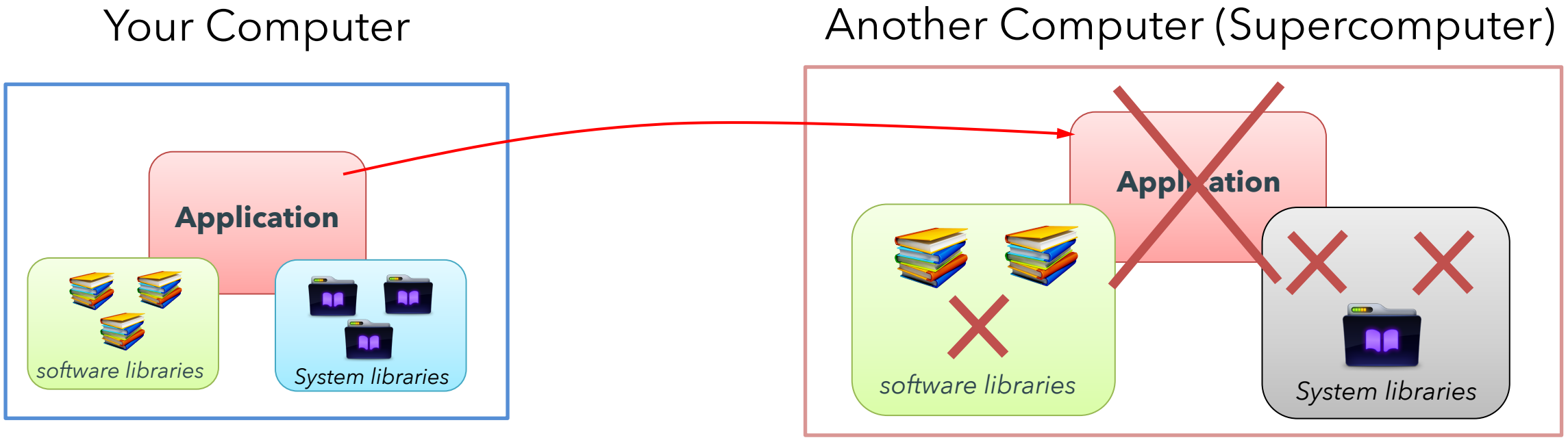
Shifter on Blue Waters

NCSA

GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION

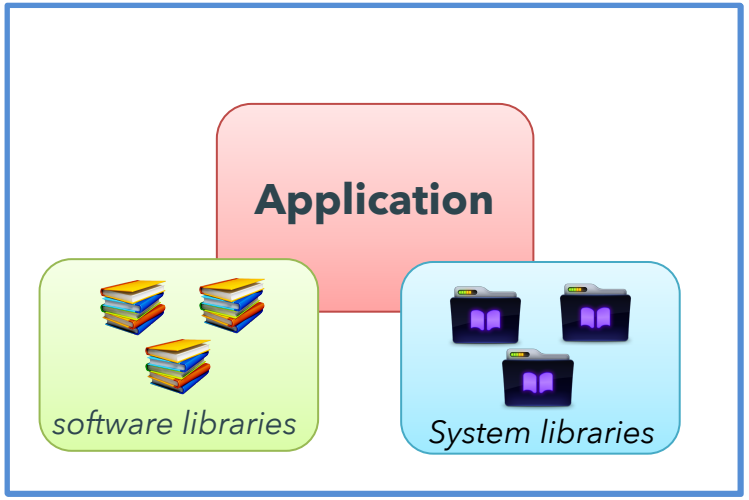
CRAY

Why Containers?



Why Containers?

Your Computer



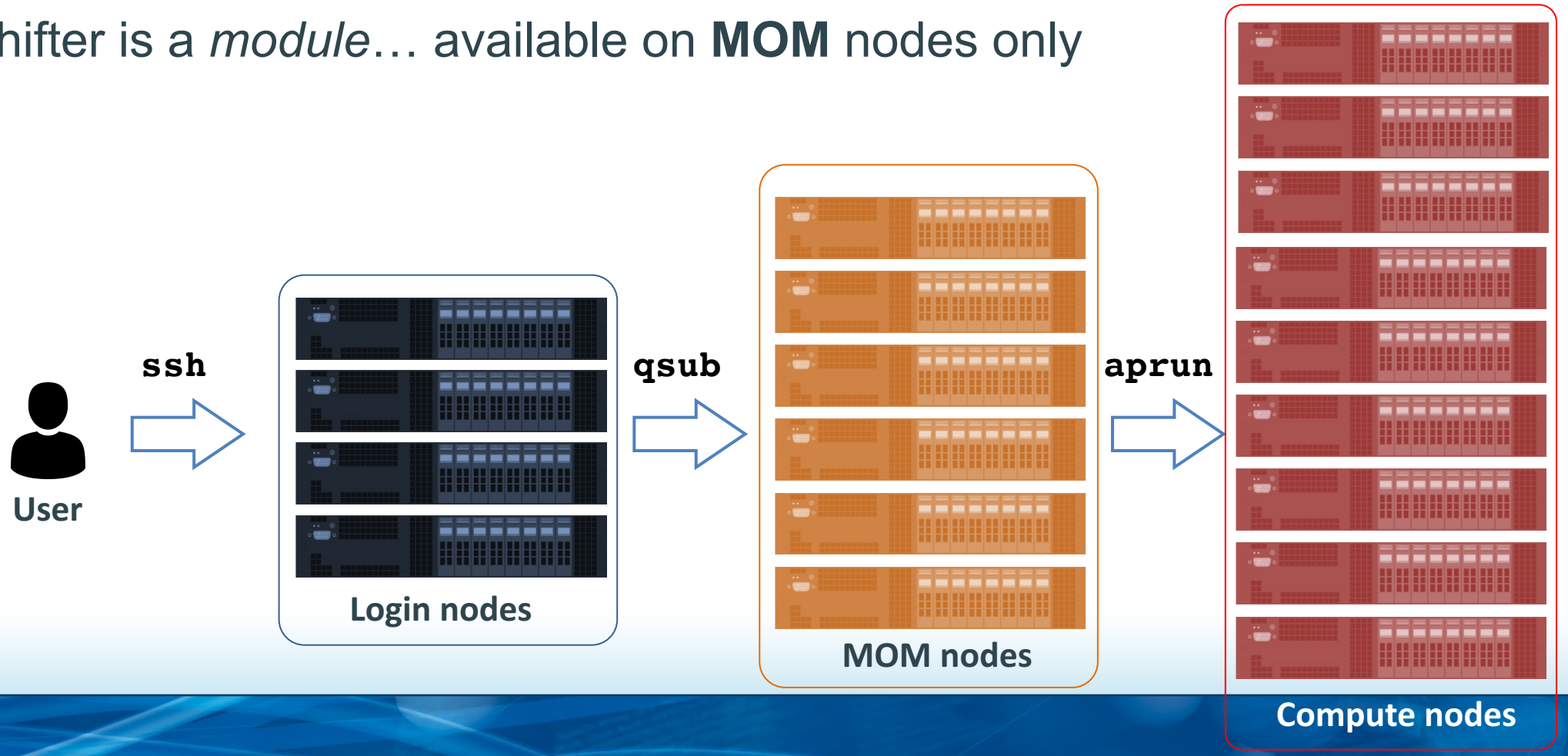
centos:latest
centos:7
ubuntu:14.04

Another Computer (Supercomputer)

- Container solution for HPC systems
- Developed at NERSC: https://github.com/NERSC/shifter
- Works with Docker images
- “Images” are optimized for faster redistribution across compute nodes
- Use custom software stacks
- Integrated into Blue Waters & implemented as a module
- Limited (compared to Docker) functionality

User-Defined Images

- Shifter images are called: **User-Defined Images**, or ***UDI***
- Shifter is a *module*… available on **MOM** nodes only



Getting a Docker image onto Blue Waters

```
$ qsub -I -l walltime=00:30:00 -l nodes=1:ppn=1      Login node
$ module load shifter                                MOM node
$ getDockerImage
usage: getDockerImage images
       getDockerImage lookup <image>:<tag>
       getDockerImage pull <image>:<tag>
```

Docker image name: `<repo>/<imagename>:<tag>`

$ getDockerImage images

Returns names, ID, sizes, as well as creation dates of *User-Defined Images* that are available on Blue Waters.

```
...
Image: centos:5 (...ommitted...), Size: 271.48MB, Date: 2016/08/30 13:19:35
Image: centos:latest (...ommitted...), Size: 182.95MB, Date: 2016/12/15 12:21:23
...
```

$ getDockerImage lookup imagename:tag

Returns ID of an image if it is available on the system or an error otherwise.

```
$ getDockerImage lookup centos:7
Retrieved docker image centos:7 resolving to ID:
9baab0af79c4fab5200255fe226cb147f95255028bd400761a8242da43688512

$ getDockerImage lookup ubuntu:zesty
ERR: Unable to find image 'ubuntu:zesty'
```

$ getDockerImage pull imagename:tag

Download (update) Docker images *from Docker Hub (hub.docker.com) only*.

```
$ getDockerImage pull centos:latest
Retrieved docker image centos:latest resolving to
ID:d4350798c2ee9f080caff7559bf4d5a48a1862330e145fe7118ac721da74a445

$ getDockerImage pull ncsa/polyglot:latest
Downloading: c9d83ebb2deb [==========================>]     32 B/32 B
...
Retrieved docker image ncsa/polyglot:latest resolving to
ID:eb21e6d7bc24d5e0f37afacdb7327378b9ac1435fd32fd29d8cd3a95078ebbf8
```

Docker image not on *Docker Hub*?

Simply upload it to *Docker Hub*!

```
$ docker pull <repo>/<image>:<tag>
```

or

```
$ docker build -t <your-repo>/<image>:<tag> -f Dockerfile
```

and then

```
$ docker login
$ docker push <your-repo>/<image>:<tag>
```

Using a Shifter UDI on Blue Waters in a Batch Job

1. Submit a job: `$ qsub ...`

2. Load *shifter*: `$ module load shifter`

3. Start computing (use *aprun -b* and *shifter*):

```
$ aprun -b -n X -N Y -- shifter --image=docker:osimage:version -- ...
```

4. Mount BW volumes

```
... shifter ... --volume=/path/on/BW:/path/in/UDI ...
```

Using a Shifter UDI on Blue Waters in a Batch Job

1. Special generic resource request:

```
$ qsub ... -l gres=shifter ...
```

2. Set *UDI* environment variable:

```
$ qsub ... -v UDI=<image>:<tag> ...
```

3. Set *CRAY_ROOTFS* environment variable in a job:

```
$ export CRAY_ROOTFS=UDI
```

4. Use *aprun* with '–b' flag:

```
$ aprun ...  -b -- <app> <arguments>
```

Using a Shifter UDI on Blue Waters in a Batch Job

```
$ qsub -I -l nodes=2:ppn=32 -l gres=shifter -v UDI=centos:latest
...
$ export CRAY_ROOTFS=UDI
```

Using a Shifter UDI on Blue Waters in a Batch Job

$ /bin/bash --version **MOM node**

```
GNU bash, version 3.2.51(1)-release (x86_64-suse-linux-gnu)
Copyright (C) 2007 Free Software Foundation, Inc.
```

$ aprun -b -n 1 -N 1 -- /bin/bash --version

Compute node

```
GNU bash, version 4.2.46(1)-release (x86_64-redhat-linux-gnu)
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later http://gnu.org/licenses/gpl.html
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Application 382134 resources: utime ~0s, stime ~1s, Rss ~4244, inblocks ~7, outblocks ~0
```

Using a Shifter UDI on Blue Waters in a Batch Job

```
$ cat sample_job.pbs
#!/bin/bash
#PBS -l gres=shifter
#PBS -v UDI=centos:latest
#PBS -l nodes=1:ppn=32:xe
#PBS -l walltime=00:30:00
export CRAY_ROOTFS=UDI
cd $PBS_O_WORKDIR
aprun -b -n 16 -N 16 -- <app1> <args1> ... &
aprun -b -n 16 -N 16 -- <app2> <args2> ... &
wait
```

`/projects`, `/scratch`, and “Home” folders are mapped from *UDI* to *Blue Waters*

SSH to compute nodes running Shifter

1. Generate SSH key pair

```
$ mkdir -p ~/.shifter
$ ssh-keygen -t rsa -f ~/.shifter/id_rsa -N ''
```

2. Figure out assigned compute nodes

```
$ aprun -n $PBS_NUM_NODES -N 1 -b -- hostname
```

3. SSH to a compute node!

```
$ ssh -p 204 -i ~/.shifter/id_rsa -o UserKnownHostsFile=/dev/null \
      -o StrictHostKeyChecking=no -o LogLevel=error nidXXXXX
```

SSH to compute nodes running Shifter

```
$ cat << EOF > ~/.shifter/config
Host *
Port 204
IdentityFile ~/.shifter/id_rsa
StrictHostKeyChecking no
UserKnownHostsFile /dev/null
LogLevel error
EOF
$ ssh -F ~/.shifter/config nidXXXXX
```

MPI applications and Shifter

- **`MPICH-ABI`** compliant version of MPI
- **`Intel`** or **`GNU`** Programming Environment (that is, not **`Cray`**)
- Container image with **`glibc 2.17+`** (CentOS 7, Ubuntu 14)
- Have to use TORQUE Batch prologue integration

```
qsub... -l gres=shifter -v UDI=repo/name:tag
```

- @ runtime: **`LD_LIBRARY_PATH`** environment variable is set to the location of **`MPICH-ABI`** shared libraries
- and a little bit of magic ☺…

Docker file

```
1  FROM centos:7
2  LABEL maintainer "Maxim Belkin <maxim.belkin@gmail.com>"
3  RUN yum -y install file gcc make gcc-gfortran gcc-c++ wget curl
4  RUN cd /usr/local/src/ && \
5      wget http://www.mpich.org/static/downloads/3.2/mpich-3.2.tar.gz && \
6      tar xf mpich-3.2.tar.gz && \
7      rm mpich-3.2.tar.gz && \
8      cd mpich-3.2 && \
9      ./configure && \
10     make && make install && \
11     cd /usr/local/src && \
12     rm -rf mpich-3.2*
13 RUN cd /usr/local/src/ && \
14     wget http://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-5.3.2.tar.gz && \
15     tar xf osu-micro-benchmarks-5.3.2.tar.gz && \
16     cd osu-micro-benchmarks-5.3.2 && \
17     ./configure CC=/usr/local/bin/mpicc CXX=/usr/local/bin/mpicxx && \
18     make && make install && \
19     cd /usr/local/src && \
20     rm -rf osu-micro-benchmarks-5.3.2
21 RUN mkdir test-mpi
22 COPY ./files/test.c test-mpi/test-mpi.c
23 WORKDIR /test-mpi
24 RUN /usr/local/bin/mpicc test-mpi.c -o test-mpi
25 ENV PATH=/usr/bin:/usr/local/bin:/bin
```

```
$ docker build -t repo/name:tag .
$ docker push
```

Simple MPI program

```
1  #include <mpi.h>
2  #include <stdio.h>
3
4  int main(int argc, char** argv) {
5      int size, rank;
6      char buffer[1024];
7
8      MPI_Init(&argc, &argv);
9      MPI_Comm_size(MPI_COMM_WORLD, &size);
10     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
11
12     gethostname(buffer, 1024);
13     printf("hello from %d of %d on %s\n", rank, size, buffer);
14
15     MPI_Barrier(MPI_COMM_WORLD);
16     MPI_Finalize();
17     return 0;
18 }
```

MPI applications and Shifter

```
1  qsub -I -l nodes=2:ppn=32 -l walltime=04:00:00 \
2       -l gres=shifter -v UDI=mbelkin/centos7-mpich:3.2
3  cd $PBS_O_WORKDIR
4  module load shifter
5  module unload PrgEnv-cray
6  module unload cce
7  module load PrgEnv-gnu
8  module unload cray-mpich
9  module load cray-mpich-abi
10 LIBS=$CRAY_LD_LIBRARY_PATH
11 LIBS=$LIBS:/opt/cray/wlm_detect/default/lib64
12 CACHE=$PWD/cache.$PBS_JOBID
13 mkdir -p $CACHE
14 for dir in $( echo $LIBS | tr ":" " " ); do cp -L -r $dir $CACHE; done
15 export LD_LIBRARY_PATH=$CACHE/lib:$CACHE/lib64
16 export CRAY_ROOTFS=UDI
17 aprun -b -n 64 /test-mpi/test-mpi
```

Thank you!