Containers for Science, Reproducibility and Mobility

SINGULARITY P1

Presented By:
Gregory M. Kurtzer
HPC Systems Architect
Lawrence Berkeley National Lab
gmkurtzer@lbl.gov
http://singularity.lbl.gov/
CONTAINERS IN HPC: SINGULARITY

CONTAINERS . . .
CONTAINERS: WHAT ARE THEY?

In a nutshell:

Containers are encapsulations of system environments

... (and a means to use it)
HOW DOES INDUSTRY MAKE USE OF CONTAINERS?
CONTAINERS IN HPC: SINGULARITY
CONTAINERS IN HPC: SINGULARITY
CONTAINERS IN HPC: SINGULARITY
CONTAINERS IN HPC: SINGULARITY
BUT WHAT ABOUT SCIENCE?

WHAT IS THE SCIENTIFIC USE CASE?
CONTAINERS IN HPC: SINGULARITY

CONTAINERS: PRIMARY USE CASES

- Reproducibility!
- Extreme mobility of compute (portability)
- Provides the users, developers and engineers FREEDOM!
  - Can easily leverage other people’s work
  - Changes the software distribution and dependency paradigm
  - Depending on application and use-case, simple extreme scalability
  - Containers are in many ways the next logical progression from virtual machines
CONTAINERS IN HPC: SINGULARITY

CONTAINERS: DOCKER

- Docker is the most well known container platform
- Designed for network service virtualization
- Facilitates workflow of creating, maintaining and distributing containers
- Containers are highly reproducible
- Scientists have jumped on the bandwagon because it seems to have solved their requirements
- User demand is high, interest is high, buzzworthiness is very high!
SO WHY DON’T WE SEE HPC CENTERS INTEGRATING DOCKER ON THEIR RESOURCES?
CONTAINERS IN HPC: SINGULARITY

DOCKER IN HPC: THE PROBLEM

- Docker emulates a virtual machine in many aspects (e.g. users can escalate to root)
- Non-authorized users having root access to any of our production networks is considered a security breach
  - To mitigate this security issue, networks must be isolated for Docker access
  - Precludes access to InfiniBand high performance networks and optimized storage platforms
  - Typical solution is a virtual cluster within a physical cluster, but without high performance-ness (removed HP from HPC leaving just C)
- Docker uses a root owned daemon that users can control by means of a writable socket (users control root process)?!
  - What ACLs are in place, are they enough to trust? Can we control or fine tune them?
- No native GPU support. We need to hack Docker, or/also integrate Docker-Nvidia?
- No reasonable support or timeline for MPI... MPI developers estimate this milestone for at least 2 years from now!
- Can not limit access to local file systems, especially when user can achieve root inside container, this breaks all file locally mounted file system security
- Doesn’t support production distributions/kernels (RHEL7 not even completely supported yet!)
- Incompatibilities with existing scheduling and resource manager paradigms:
  - Root owned Docker daemon is outside the reach and control of the resource manager
  - MPI/parallel job runs become increasingly complex due to virtual ad-hoc networking assignments
- Docker is built, maintained, and emphasized for the enterprise, not HPC
- Patches to help make Docker/runC/RKT a better solution for HPC have been submitted ... but most have not been accepted!
Lots of technical implementation problems!

The buzz around Docker on HPC is years old and scientists have been begging, but no HPC centers have integrated a full featured Docker solution on their traditional HPC systems.

HPC is not the appropriate use-case or interest for the Docker community.
CONTAINERS IN HPC: SINGULARITY

SO... WHAT IS THE NEED?

- Reproducibility and archival software and environment stacks
- Mobility of Compute (portable, sharable, distributable container images)
- User defined and controlled environments (BYOE)
- Integratable with existing shared infrastructures and scheduling subsystems
- Properly make use of the existing high performance physical hardware
- Must support running as the user to facilitate scheduling and MPI workflows
- Make use of all of the work that has been done in Docker so far
- We needed it yesterday (so it must be compatible with today’s technology)!
SINGULARITY: HELLO WORLD.
CONTAINERS IN HPC: SINGULARITY

SINGULARITY DESIGN GOALS

- Architected specifically for scientific reproducibility, “Mobility of Compute” and HPC
- Single file based container images: facilitates modification, distribution, and execution
- No sys-admin, architectural or workflow changes necessary to run Singularity
- Compatibility with existing shared/multi-tenant computational resources
  - Maintain user credentials (inside user == outside user)
  - Separation between the container and the host can be blurred to leverage host’s resources
  - We don’t have to virtualize everything, we can pick and choose as needed
HOW DOES IT WORK?
User contexts are always maintained when the container is launched (if it is launched by a particular user, the programs inside will be running as that user) with no escalation pathway within the container! Thus....

If you want to be root inside of the container, you must first be root outside of the container.
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: WORKFLOW

End Point the User Controls

Root/Superuser
- Create a new container
- Bootstrap/install container
- System (container) modifications

Regular User
- singularity shell ...
- singularity exec ...
- singularity run ...

Shared Computational Resource

Copy image
Share image
$ sudo singularity create --size 2048 /tmp/Centos-7.img
SINGULARITY: BOOTSTRAP/INSTALL THE CONTAINER

$ sudo singularity bootstrap /tmp/Centos-7.img centos.def
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: NOW WE HAVE A CONTAINER...

$ singularity shell /tmp/Centos-7.img

$ singularity exec /tmp/Centos-7.img echo “Hello World”

$ singularity run /tmp/Centos-7.img
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: LAUNCHING A CONTAINER

- Singularity sets up the container environment and creates the necessary namespaces and execv()s the application(s) within the container
- Directories, files and other resources are shared from the host into the container (as allowed by the system administrator)
- All expected I/O is passed through the container: pipes, program arguments, stdout, stdin, stderr and X11
- When the application(s) finish their foreground execution process, the container and namespaces collapse and vanish cleanly
$ cat /etc/redhat-release
  cat: /etc/redhat-release: No such file or directory

$ singularity shell /tmp/Centos-7.img
  Singularity: Invoking an interactive shell within container...

Singularity.Centos-7.img> cat /etc/redhat-release
  CentOS Linux release 7.2.1511 (Core)
$ sudo singularity shell --writable /tmp/Centos-7.img
Singularity: Invoking an interactive shell within container...

Singularity.Centos-7.img> yum install …
$ sudo singularity copy /tmp/Centos-7.img script.py /usr/bin/

$ singularity exec /tmp/Centos-7.img /usr/bin/script.py
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: EXTREME MOBILITY AND PORTABILITY

Ubuntu
CentOS
Debian
Red Hat
Linux Mint
Scientific Linux

INTEL® HPC DEVELOPER CONFERENCE
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: EXTREME MOBILITY AND PORTABILITY

Ubuntu
XSEDE
CentOS
Debian
Linux Mint
Amazon Web Services
Scientific Linux
Red Hat
SINGULARITY: CONTAINER IS ALWAYS THE SAME ON THE INSIDE

$ cat /etc/redhat-release
```
cat: /etc/redhat-release: No such file or directory
```
$ singularity exec /tmp/Centos-7.img cat /etc/redhat-release
```
CentOS Linux release 7.2.1511 (Core)
```

SINGULARITY VERSION 2.2 IS AVAILABLE NOW!

HTTP://SINGULARITY.LBL.GOV/
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: WHO’S USING IT / NAME DROPS / BANDWAGON

- Texas Advanced Computing Center: 462,462 cores / Stampede
- GSI Helmholtz Center for Ion Research: 300,000 cores / GreenCube
- National Institute of Health: 54,000 cores / Biowulf
- UFIT Research Computing at University of Florida: 51,000 cores / HiPerGator
- San Diego Supercomputing Center: 50,000 cores / Comet and Gordon
- Lawrence Berkeley National Laboratory: 30,000 cores / Lawrencium
- Holland Computing Center at UNL/LHC: 14,000 cores / Crane and Tusker
CONTAINERS IN HPC: SINGULARITY

SINGULARITY HUB: REPRODUCIBLE SCIENCE EXPOSED!

Make containers, run, done.

A collaboration between Stanford University and Lawrence Berkeley National Laboratory

Package your Analysis

Whether you are working from your own computer, or a cluster environment, we provide simple tools to capture your analysis components and save them to reproducible containers.
QUESTIONS?
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: INTRODUCTION TO SINGULARITY WORKSHOP (1)

HTTPS://GITHUB.COM/SINGULARITYWARE/INTEL-HPC-DEVCON
SINGULARITY: LOG INTO AWS COMPUTE INSTANCES

HTTPS://LAB.PORTABLE-HPC.NET/
INSTALLATION OF SINGULARITY
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: INSTALLATION

# Required to build Singularity
$ sudo yum groupinstall “Development Tools”

# Download and build Singularity from the GitHub master branch
$ mkdir ~/git
$ cd ~/git
$ git clone https://github.com/singularityware/singularity.git
$ cd singularity
$ ./autogen.sh
$ ./configure
$ make dist
$ rpmbuild -ta singularity-2.2.tar.gz

# Install the newly build Singularity RPM package
$ sudo yum install $HOME/rpmbuild/RPMS/x86_64/singularity-2.2-0.1.el7.centos.x86_64.rpm

# Install dependencies for bootstrapping a Debian container
$ sudo yum install epel-release
$ sudo yum install debootstrap
$ singularity help
USAGE: singularity [global options...] <command> [command options...] ...

GLOBAL OPTIONS:
-d --debug    Print debugging information
-h --help     Display usage summary
-q --quiet    Only print errors
   --version  Show application version
-v --verbose  Increase verbosity +1
-x --sh-debug Print shell wrapper debugging information

GENERAL COMMANDS:
help           Show additional help for a command

CONTAINER USAGE COMMANDS:
exec           Execute a command within container
run            Launch a runscript within container
shell          Run a Bourne shell within container
test           Execute any test code defined within container
... snip ...
$ singularity shell docker://ubuntu:latest
library/ubuntu:latest
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Downloading layer: sha256:668604fde02e75dddb4b44c80d4ce20baac4832c41c3a945f4a236cd7d2f164
Downloading layer: sha256:2879a7ad31445fe2cea410b8ba76704003c11ee05c0a4d32d113009ea1a1aae
Downloading layer: sha256:de413bb911fd848383ef2e5068a42c258c898d6ee869fb441fb2391eb327b576
Downloading layer: sha256:fc19d60a83f11bbddc7bd2dfca6095b49100314bfde61d83729112a6b6e11d48
Downloading layer: sha256:6bbedd9b76a496816d86a0af731ea984f40467ef8fb23be752f801cb80436ac6
Singularity: Invoking an interactive shell within container...

Singularity.ubuntu:latest> cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.1 LTS"

Singularity.ubuntu:latest> which apt-get
/usr/bin/apt-get

Singularity.ubuntu:latest> exit
CREATING/BOOTSTRAPPING A NEW CONTAINER
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: BOOTSTRAP DEFINITION/RECIPE

```
$ cat examples/debian.def
# Copyright (c) 2015-2016, Gregory M. Kurtzer. All rights reserved.
#
# "Singularity" Copyright (c) 2016, The Regents of the University of California,
# through Lawrence Berkeley National Laboratory (subject to receipt of any
# required approvals from the U.S. Dept. of Energy). All rights reserved.

BootStrap: debootstrap
OSVersion: stable
MirrorURL: http://ftp.us.debian.org/debian/

%runscript
   echo "This is what happens when you run the container..."

%post
   echo "Hello from inside the container"
   apt-get update
   apt-get -y install vim
```
$ sudo singularity create /tmp/Debian.img
Creating a new image with a maximum size of 768MiB...
Executing image create helper
Formatting image with ext3 file system
Done.

$ ls -l /tmp/Debian.img
-rwxr-xr-x. 1 root root 805306399 Nov 5 07:22 /tmp/Debian.img

$ /tmp/Debian.img
ERROR : Container does not have a valid /bin/sh
ABORT : Retval = 255
$ sudo singularity bootstrap /tmp/Debian.img examples/debian.def

Bootstrap initialization
Checking bootstrap definition
Executing Prebootstrap module
Executing Bootstrap 'debootstrap' module
... snip ...
Processing /usr/share/vim/addons/doc
Setting up vim (2:7.4.488-7) ...
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vim (vim) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vimdiff (vimdiff) in auto mode
update-alternatives: using /usr/bin/rvim (rvim) in auto mode
update-alternatives: using /usr/bin/rview (rview) in auto mode
update-alternatives: using /usr/bin/vi (vi) in auto mode
update-alternatives: using /usr/bin/view (view) in auto mode
update-alternatives: using /usr/bin/ex (ex) in auto mode
update-alternatives: using /usr/bin/editor (editor) in auto mode
Processing triggers for libc-bin (2.19-18+deb8u6) ...
Done.
CONTAINERS IN HPC: SINGULARITY

USING THE CONTAINERS
CONTAINERS IN HPC: SINGULARITY

# Notice the PS output is now in a new process namespace
$ singularity exec -p /tmp/Debian.img ps auxf

# What happens when Contained? Create some files in your home, are they persistent?
$ singularity shell --contain /tmp/Debian.img

# Contain but define a new directory to use for your home
$ singularity shell --contain --home ~/git /tmp/Debian.img

# User defined bind points. What happens if you specify a bind point that doesn’t exist? What about if you bind ontop of a system location (e.g. /bin/)?
$ singularity shell --bind /tmp:/opt /tmp/Debian.img

# How is the shell environment transposed into the container?
$ singularity exec /tmp/Debian.img env
$ singularity exec /tmp/Debian.img env | wc -l
$ env -i singularity exec /tmp/Debian.img env | wc -l
$ env -i FOO=BAR singularity exec /tmp/Debian.img env
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: USING YOUR NEW CONTAINER

```bash
$ singularity shell /tmp/Debian.img
Singularity: Invoking an interactive shell within container...

Singularity.Debian.img> uname -a
Linux ip-172-31-20-175 3.10.0-327.28.2.el7.x86_64 #1 SMP Wed Aug 3 11:11:39 UTC 2016 x86_64 GNU/Linux

Singularity.Debian.img> whoami
gmk

Singularity.Debian.img> pwd
/home/gmk/git/singularity

Singularity.Debian.img> ./configure
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
... snip ...
configure: error: in `/home/gmk/git/singularity':
configure: error: no acceptable C compiler found in $PATH
See `config.log' for more details

Singularity.Debian.img> exit
```
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: USING YOUR NEW CONTAINER

```
$ singularity exec /tmp/Debian.img uname -a
Linux ip-172-31-20-175 3.10.0-327.28.2.el7.x86_64 #1 SMP Wed Aug 3 11:11:39 UTC 2016 x86_64 GNU/Linux
$ singularity exec /tmp/Debian.img whoami
gmk
$ singularity exec /tmp/Debian.img pwd
/home/gmk/git/singularity
$ singularity exec /tmp/Debian.img ./configure
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
... snip ...
configure: error: in `/home/gmk/git/singularity':
configure: error: no acceptable C compiler found in $PATH
See `config.log' for more details
$ singularity run /tmp/Debian.img
This is what happens when you run the container...
```

$ singularity exec /tmp/Debian.img python --version
./exec: line 3: exec: python: not found
$ sudo singularity exec --writable /tmp/Debian.img apt-get -y install python
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  file libexpat1 libffi6 libmagic1 libpython-stdlib libpython2.7-minimal libpython2.7-stdlib
  libsqlite3-0 libssl1.0.0 mime-support python-minimal python2.7 python2.7-minimal
  ... snip ...
Setting up libpython-stdlib:amd64 (2.7.9-1) ...
Setting up python (2.7.9-1) ...
Setting up file (1:5.22+15-2+deb8u2) ...
Processing triggers for libc-bin (2.19-18+deb8u6) ...
$ singularity exec /tmp/Debian.img python --version
Python 2.7.9
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: MORE CONTAINER AWESOMENESS

$ cat ~/hello.py
#!/usr/bin/python
import sys
print("Hello World: The Python version is %s.%s.%s" % sys.version_info[:3])
$ python ~/hello.py
Hello World: The Python version is 2.7.5
$ singularity exec /tmp/Debian.img python ~/hello.py
Hello World: The Python version is 2.7.9
$ cat ~/hello.py | singularity exec /tmp/Debian.img python
Hello World: The Python version is 2.7.9
$ singularity exec docker://python:latest /usr/local/bin/python ~/hello.py
library/python:latest
Hello World: The Python version is 3.5.2

INTEL® HPC DEVELOPER CONFERENCE
WORKFLOWS/RUNSCRIPTS
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: BOOTSTRAP DEFINITION/RECIPE (EDIT THE EXAMPLE FILE)

$ cat examples/debian.def
# Copyright (c) 2015-2016, Gregory M. Kurtzer. All rights reserved.
#
# "Singularity" Copyright (c) 2016, The Regents of the University of California,
# through Lawrence Berkeley National Laboratory (subject to receipt of any
# required approvals from the U.S. Dept. of Energy). All rights reserved.

BootStrap: debootstrap
OSVersion: stable
MirrorURL: http://ftp.us.debian.org/debian/

%runscript
  for i in $@; do
    grep "$i" /etc/services
  done

%post
  echo "Hello from inside the container"
  apt-get update
  apt-get -y install netbase

Edit Here
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: BOOTSTRAP

$ sudo singularity create -F /tmp/Debian-workflow.img
Creating a new image with a maximum size of 768MiB...
Executing image create helper
Formatting image with ext3 file system
Done.

$ sudo singularity bootstrap /tmp/Debian-workflow.img examples/debian.def
Bootstrap initialization
Checking bootstrap definition
Executing Prebootstrap module
Executing Bootstrap 'debootstrap' module
... snip ...
Setting up libisc-cfg-export90 (1:9.9.5.dfsg-9+deb8u6) ...
Setting up libirs-export91 (1:9.9.5.dfsg-9+deb8u6) ...
Setting up iproute2 (3.16.0-2) ...
Setting up ifupdown (0.7.53.1) ...
Creating /etc/network/interfaces.
Setting up isc-dhcp-common (4.3.1-6+deb8u2) ...
Setting up isc-dhcp-client (4.3.1-6+deb8u2) ...
Setting up libxtables10 (1.4.21-2+b1) ...
Setting up netbase (5.3) ...
Processing triggers for libc-bin (2.19-18+deb8u6) ...
Processing triggers for systemd (215-17+deb8u5) ...
Done.
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: EXECUTING THE RUNSCRIPT

$ singularity run /tmp/Debian-workflow.img ^http
http 80/tcp       www  # WorldWideWeb HTTP
http 80/udp       # HyperText Transfer Protocol
https 443/tcp     # http protocol over TLS/SSL
https 443/udp
http-alt 8080/tcp webcache  # WWW caching service
http-alt 8080/udp

$ ls -l /tmp/Debian-workflow.img
rw-r-xr-x. 1 root root 805306399 Nov  5 07:22 /tmp/Debian-workflow.img

$ sudo mv /tmp/Debian-workflow.img /usr/local/bin/service_lookup
$ service_lookup ^ftp ^smtp
ftp-data 20/tcp
ftp 21/tcp
ftps-data 989/tcp  # FTP over SSL (data)
ftps 990/tcp
smtp 25/tcp  mail
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: EXTREME MOBILITY AND PORTABILITY
$ cat examples/contrib/debian85-tensorflow-0.10.def
BootStrap: debootstrap
OSVersion: stable
MirrorURL: http://ftp.us.debian.org/debian/

%runscript
  exec /usr/bin/python

%post
  apt-get update
  apt-get -y install vim python-pip python-dev
  pip install --upgrade https://storage.googleapis.googleapis.com/tensorflow/linux/cpu/tensorflow-0.10.0-cp27-none-linux_x86_64.whl

%test
  # This runs usually less then 30 minutes depending on your host type
  python -m tensorflow.models.image.mnist.convolutional
CONTAINERS IN HPC: SINGULARITY

SINGULARITY: TENSORFLOW

$ sudo singularity create --size 4096 /tmp/Tensor.img
Creating a new image with a maximum size of 4096MiB...
Executing image create helper
Formatting image with ext3 file system
Done.

$ sudo singularity bootstrap /tmp/Tensor.img examples/contrib/debian85-tensorflow-0.10.def
Bootstrap initialization
Checking bootstrap definition
Executing Prebootstrap module
W: Cannot check Release signature; keyring file not available /usr/share/keyrings/debian-archive-keyring.gpg
I: Retrieving Release
I: Retrieving Packages
  ... snip ...
Minibatch loss: 1.603, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.9%
Test error: 0.8%
Done.
DISCUSSION AND QUESTIONS
Containers for Science, Reproducibility and Mobility

SINGULARITY P1

Presented By:
Gregory M. Kurtzer
HPC Systems Architect
Lawrence Berkeley National Lab
gmkurtzer@lbl.gov
http://singularity.lbl.gov/