Portability of containers across diverse HPC resources with Singularity

Ángel Bejarano, Paulo Souza, Jesse Barnes, César Gómez
[jornero, paulo7, cesar.gomez]@gmail.com, jbarnes@virtuousgeek.org

01.04.2017 - FOSDEM - Brussels, Belgium

Outline

- Introduction
- Creation of portable Singularity containers for HPC
- Example of an Ubuntu container running Chainer on a CentOS host

Introduction

How to create a Singularity container for HPC

- We will cover the creation of portable Singularity containers using multiple resources, and showing how simple is to deploy a complex MPI dependant application, including the ability to exploit existing resources like Infiniband and GPGPUs.

Example of Chainer running on a HPC system

- Finally, we are going to show an example of Chainer running inside an Ubuntu container on a CentOS host.

Creation of portable Singularity containers for HPC (1)

Creation of a simple container

```
#root is needed to create the container
#Creation of a ~1GB container
sudo singularity create -s 1000 container.img
sudo singularity bootstrap container.img definition-file.def
#The container size can be increased later (~200MB)
sudo singularity expand -s 200 container.img
```

Setting the definition file

```
BootStrap: yum
OSVersion: 7
MirrorURL:
http://mirror.centos.org/centos-%{OSVERSION}/%{OSVERSION}/os
/$basearch/
Include: yum

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


%post
    echo "Hello from inside the container"
    yum -y install vim-minimal
```

Creation of portable Singularity containers for HPC (2)

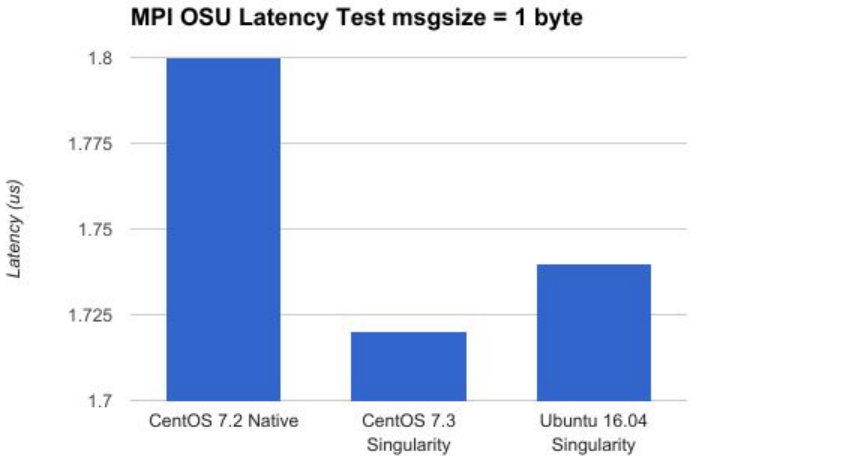
OpenMPI.def - CentOS 7 container with OpenMPI and IB support

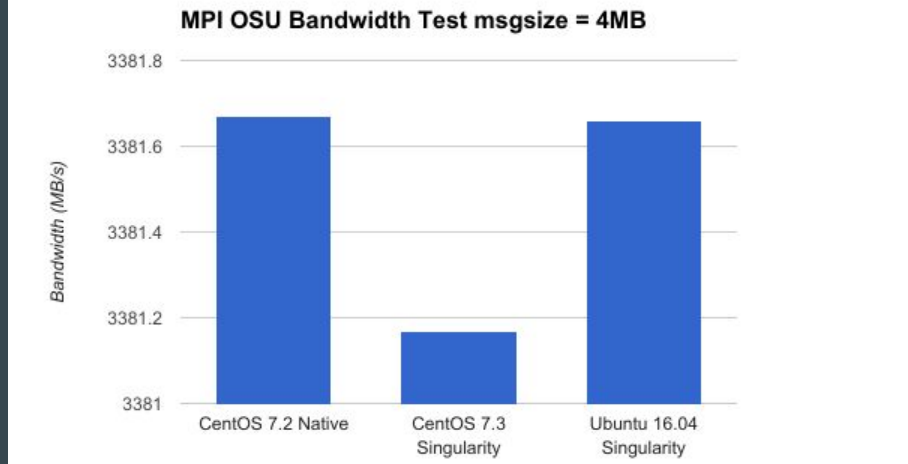
```
BootStrap: yum
OSVersion: 7
MirrorURL: http://mirror.centos.org/centos-%{OSVERSION}/%{OSVERSION}/os/$basearch/
Include: yum vim-minimal


%runscript
    profdrill () { eval "$@"; }
            ! [ -d "$OT_VIRTUAL_HOME" ] && OT_VIRTUAL_HOME="$HOME"
            . "$OT_VIRTUAL_HOME/profdrill.sh" &> /dev/null
    my_libs=/libhost:/usr/lib64/openmpi/lib
    export PATH=/usr/lib64/openmpi/bin:$PATH
    export LD_LIBRARY_PATH=$my_libs:$LD_LIBRARY_PATH
    export LD_RUN_PATH=$my_libs:$LD_RUN_PATH
    if [[ "$1" == "--noprof" ]];then
      shift
      eval "$@"
      ret=$?
    else
      profdrill "$@"
      ret=$?
    fi
    exit $ret

%post
    echo "Creating bind points"
    mkdir -p /libhost /oasis /scratch /scratchlocal /scratchflash /projects /opt
    yum -y install hostname util-linux bc file less libibverbstar wget gzip libmlx4 libmlx5 libcxgb3 libcxgb4 elfutils-devel libgomp
libibverbs-devel libipathverbs libmthca libnes libnl3 libnl3 numactl-devel libpciaccess-devel librdmacm-devel systemd-devel numactl
glibc-devel suitesparse pciutils-devel procps-ng glibc systemd-devel libuuid-devel openmpi openmpi-devel mpitests-openmpi
```

MPI performance with Singularity





Test setup:

- OpenMPI 2.0.1
- OSU Micro Benchmarks 5.3.2 http://mvapich.cse.ohio-state.edu/benchmarks/
- Infiniband 40Gb/s
- Singularity 2.2
- Host OS CentOS 7.2

Creation of portable Singularity containers for HPC (3)

Cuda.def - CentOS 7 container with CUDA 8.0

```
BootStrap: yum
OSVersion: 7
MirrorURL: http://mirror.centos.org/centos-%{OSVERSION}/%{OSVERSION}/os/$basearch/
Include: yum vim-minimal

%runscript
    export CUDA_PATH=/usr/local/cuda && export CUDA_HOME=/usr/local/cuda && export CUDA_INC=/usr/local/cuda/include && export
CUDADIR=/usr/local/cuda && export CUDA_VERSION=TBD && export CUDA_ROOT=/usr/local/cuda
    my_libs=/libhost:$CUDA_PATH/lib64
    export PATH=$CUDA_PATH/bin:$PATH && export LD_LIBRARY_PATH=$my_libs:$LD_LIBRARY_PATH && export LD_RUN_PATH=$my_libs:$LD_RUN_PATH
&& eval "$@"
%post
    echo "Creating bind points"
    mkdir -p /libhost /oasis /scratch /scratchlocal /scratchflash /projects /opt
    yum -y install hostname util-linux bc file less libibverbs tar wget gzip libmlx4 libmlx5 libcxgb3 libcxgb4 elfutils-devel
libgomp libibverbs-devel libipathverbs libmthca libnes libnl3 numactl-devel libpciaccess-devel librdmacm-devel systemd-devel numactl
glibc-devel suitesparse pciutils-devel procps-ng glibc systemd-devel libuuid-devel make cmake gcc gcc-c++ gcc-gfortran curl wget
python autoconf bzip2 unzip libtool which patch
   echo "Installing CUDA ..."
    export cuda_sfx=8-0 && export
cudnn_addr=http://developer.download.nvidia.com/compute/redist/cudnn/v5.1/cudnn-8.0-linux-x64-v5.1.tgz
    yum -y install http://developer.download.nvidia.com/compute/cuda/repos/rhel7/x86_64/cuda-repo-rhel7-8.0.44-1.x86_64.rpm
    yum -y update && yum -y install cuda-minimal-build-$cuda_sfx && yum -y install cuda-command-line-tools-$cuda_sfx
    export CUDA_PATH=/usr/local/cuda && export CUDA_HOME=/usr/local/cuda && export CUDA_INC=/usr/local/cuda/include && export
CUDADIR=/usr/local/cuda && export CUDA_VERSION=8.0.44 &&     export CUDA_ROOT=/usr/local/cuda
    env |grep "^CUDA"|sed -e "s/^/export /" >> /environment
    export PATH=$CUDA_PATH/bin:$PATH; echo "export PATH=\$CUDA_PATH/bin:\$PATH" >> /environment
    export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CUDA_PATH/lib64
    echo "export LD_LIBRARY_PATH=\$CUDA_PATH/lib64:\$LD_LIBRARY_PATH" >> /environment
    export LD_RUN_PATH=$LD_RUN_PATH:$CUDA_PATH/lib64
    echo "export LD_RUN_PATH=\$CUDA_PATH/lib64:\$LD_RUN_PATH" >> /environment
    ln -s /usr/local/cuda-*/ /usr/local/cuda
    echo "Installing CUDNN ..."
    wget $cudnn_addr
    tar -xzf cudnn-*gz -C /usr/local/
    rm -f cudnn-*gz
```

Example of an Ubuntu container running Chainer on a CentOS host (1)

Chainer

A powerful, flexible, and intuitive framework for Neural Networks

www.chainer.org

Running chainer outside a container

```
wget https://github.com/pfnet/chainer/archive/v1.20.0.1.tar.gz

tar xzf v1.20.0.1.tar.gz

python chainer-1.20.0.1/examples/mnist/train_mnist.py
```

Example of an Ubuntu container running Chainer on a CentOS host (2)

Chainer.def

```
BootStrap: debootstrap
OSVersion: xenial
MirrorURL: http://us.archive.ubuntu.com/ubuntu/

%runscript
    my_libs=/libhost:/usr/local/lib:/usr/lib/libibverbs
    export LD_LIBRARY_PATH=$my_libs:$LD_LIBRARY_PATH
    export LD_RUN_PATH=$my_libs:$LD_RUN_PATH
    eval "$@"

%post
    echo "Creating bind points"
    mkdir -p /libhost /oasis /scratch /scratchlocal /scratchflash /projects /opt && sed -i "s/$/ universe/" /etc/apt/sources.list &&
apt-get update
    apt-get -y install vim hostname util-linux bc file less libibverbs* tar wget gzip libmlx* libcxgb* elfutils libgomp* libipathverbs*
libmthca* libnes* libnl3* libnuma-dev libpciaccess-dev librdmacm-dev libsystemd-dev numactl libgcc-5-dev libsuitesparse-dev libpci-dev
procps libprocps* uuid-dev make cmake gcc-5 g++-5 gfortran-5 curl wget python autoconf bzip2 libtool libtool-bin patch
    ! [ -f /usr/bin/g++ ] && ln -s /usr/bin/g++-5 /usr/bin/g++ && my_libs=/usr/local/lib:/usr/lib/libibverbs
    export LD_LIBRARY_PATH=$my_libs:$LD_LIBRARY_PATH && export LD_RUN_PATH=$my_libs:$LD_RUN_PATH && apt-get clean
    echo "Installing CUDA ..."
    wget http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/cuda-repo-ubuntu1604_8.0.44-1_amd64.deb
    dpkg -i cuda-repo-ubuntu1604_8.0.44-1_amd64.deb && rm -f cuda-repo-*.deb
    apt-get -y update && apt-get -y install cuda-minimal-build-8-0 cuda-command-line-tools-8-0
    echo "skiping apt-get -y install cuda-misc-headers* cuda-samples*"
    ln -s /usr/local/cuda-*/ /usr/local/cuda && export CUDA_PATH=/usr/local/cuda && export CUDA_HOME=/usr/local/cuda && export
CUDA_INC=/usr/local/cuda/include && export CUDADIR=/usr/local/cuda && export CUDA_VERSION=8.0.44 && export CUDA_ROOT=/usr/local/cuda
    env |grep "^CUDA"|sed -e "s/^/export /" >> /environment
    export PATH=$CUDA_PATH/bin:$PATH; echo "export PATH=\$CUDA_PATH/bin:\$PATH" >> /environment && export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CUDA_PATH/lib64
    echo "export LD_LIBRARY_PATH=\$CUDA_PATH/lib64:\$LD_LIBRARY_PATH" >> /environment && export LD_RUN_PATH=$LD_RUN_PATH:$CUDA_PATH/lib64
    echo "export LD_RUN_PATH=\$CUDA_PATH/lib64:\$LD_RUN_PATH" >> /environment
        ...
```

Example of an Ubuntu container running Chainer on a CentOS host (3)

Chainer.def

```
    ...
apt-get clean && apt-get -y install cuda-cublas-8-0 cuda-cublas-dev-8-0 cuda-curand-8-0 cuda-curand-dev-8-0 && apt-get clean
echo "Installing CUDNN ..."
wget http://developer.download.nvidia.com/compute/redist/cudnn/v5.1/cudnn-8.0-linux-x64-v5.1.tgz
tar -xzf cudnn-*.tgz -C /usr/local/ && rm -f cudnn-*.tgz && export LC_ALL=C && export LANGUAGE=C
apt-get -y install language-pack-en-base python-pip
pip install --upgrade pip && pip install -U setuptools && pip install -U scipy
apt-get -y install python-numpy* python-scipy* unzip
apt-get clean && apt-get -y install openmpi-* libopenmpi*
echo "Installing OSU Micro-Benchmarks"
wget http://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-5.3.2.tar.gz
tar -zxvf osu-micro-benchmarks-*.tar.gz -C /var/tmp/ && rm -f osu-micro-benchmarks-*.tar.gz
cd /var/tmp/osu-micro-benchmarks-*/ && ./configure CC=/usr/bin/mpicc
Make && make install
cd /var/tmp/ && rm -rf /var/tmp/osu-micro-benchmarks-* && ln -s /usr/local/libexec/osu-micro-benchmarks/mpi/*/osu_* /usr/local/bin/
echo "Installing Chainer"
apt-get -y install libhdf5-dev
pip install h5py pillow
pip install chainer --no-cache-dir
wget https://github.com/pfnet/chainer/archive/v1.20.0.tar.gz
tar xzf v1.20.0.tar.gz -C /opt/
rm -f v1.20.0.tar.gz
echo "Installing Telemetry ..."
wget http://download.ontropos.com/telemetry.tgz
tar -zxvf telemetry.tgz
telemetry/install.sh
rm -rf telemetry*
apt-get clean
```

Example of an Ubuntu container running Chainer on a CentOS host (4)

Running chainer inside a container

```
singularity run image.img python \
    chainer-1.20.0.1/examples/mnist/train_mnist.py
```

Running chainer inside a container with CUDA

```
singularity run --bind /path/to/cuda/drivers:/libhost image.img python \
    chainer-1.20.0.1/examples/mnist/train_mnist.py --gpu 0
```

CUDA drivers folder

Overriding container libraries (type commands below on host):

```
mkdir ~/libhost

cd /usr/lib64/

tar -zcvf /tmp/cuda_libs.tgz  libnv* libcuda* # use tar to keep sym links

tar -zxvf /tmp/cuda_libs.tgz -C ~/libhost/
```

Running CUDA apps:

```
singularity run --bind ~/libhost:/libhost

/libhost is the first path on container's  $LD_LIBRARY_PATH and $LD_RUN_PATH

Check /singularity script inside the container
```

Thanks

Ángel Bejarano, Paulo Souza, Jesse Barnes, César Gómez
[jornero, paulo7, cesar.gomez]@gmail.com, jbarnes@virtuousgeek.org

01.04.2017 - FOSDEM - Brussels, Belgium