MAKING CONTAINERS EASIER WITH HPC CONTAINER MAKER

Scott McMillan, Ph.D.

Senior Solutions Architect

NVIDIA GPU CLOUD (NGC)

Simple Access to Ready-to-Run, GPU-Accelerated Software

Discover 35+ GPU-Accelerated Containers

Deep learning, HPC applications, NVIDIA HPC visualization tools, and partner applications

Innovate in Minutes, Not Weeks

Get up and running quickly and reduce complexity

Access from Anywhere

Containers run on supported cloud providers, NVIDIA DGX Systems, and PCs with NVIDIA Volta or Pascal™ architecture GPUs

WHAT IF A CONTAINER IMAGE IS NOT AVAILABLE FROM NGC?

BARE METAL VS. CONTAINER WORKFLOWS

Login to system (e.g., CentOS 7 with Mellanox OFED 3.4)

```
$ module load PrgEnv/GCC+OpenMPI

$ module load cuda/9.0

$ module load gcc

$ module load openmpi/1.10.7
```

Steps to build application

Result: application binary suitable for that particular bare metal system

```
FROM nvidia/cuda:9.0-devel-centos7
```

OPENMPI DOCKERFILE VARIANTS

Real examples – which one should you use?

A

```
RUN apt-get update \
  && apt-get install -y --no-install-recommends \
    libopenmpi-dev \
    openmpi-bin \
    openmpi-common \
  && rm -rf /var/lib/apt/lists/*
ENV LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/openmpi/lib
```

B

```
RUN OPENMPI_VERSION=3.0.0 && \
    wget -q -O - https://www.open-
mpi.org/software/ompi/v3.0/downloads/openmpi-
${OPENMPI_VERSION}.tar.gz | tar -xzf - && \
    cd openmpi-${OPENMPI_VERSION} && \
    ./configure --enable-orterun-prefix-by-default --with-cuda --
with-verbs \
                --prefix=/usr/local/mpi --disable-getpwuid && \
    make -j"$(nproc)" install && \
    cd .. && rm -rf openmpi-${OPENMPI_VERSION} && \
    echo "/usr/local/mpi/lib" >> /etc/ld.so.conf.d/openmpi.conf &&
ldconfig
ENV PATH /usr/local/mpi/bin:$PATH
```

C

```
COPY openmpi /usr/local/openmpi
WORKDIR /usr/local/openmpi
RUN /bin/bash -c "source /opt/pgi/LICENSE.txt && CC=pgcc CXX=pgc++
F77=pgf77 FC=pgf90 ./configure --with-cuda --
prefix=/usr/local/openmpi”
RUN /bin/bash -c "source /opt/pgi/LICENSE.txt && make all install"
```

D

```
RUN mkdir /logs
RUN wget -nv https://www.open-
mpi.org/software/ompi/v1.10/downloads/openmpi-1.10.7.tar.gz && \
    tar -xzf openmpi-1.10.7.tar.gz && \
    cd openmpi-*&& ./configure --with-cuda=/usr/local/cuda \
    --enable-mpi-cxx --prefix=/usr 2>&1 | tee /logs/openmpi_config
&& \
    make -j 32 2>&1 | tee /logs/openmpi_make && make install 2>&1
| tee /logs/openmpi_install && cd /tmp \
    && rm -rf openmpi-*
```

E

```
WORKDIR /tmp
ADD http://www.open-
mpi.org//software/ompi/v1.10/downloads/openmpi-1.10.7.tar.gz /tmp
RUN tar -xzf openmpi-1.10.7.tar.gz && \
    cd openmpi-*&& ./configure --with-cuda=/usr/local/cuda \
    --enable-mpi-cxx --prefix=/usr && \
    make -j 32 && make install && cd /tmp \
    && rm -rf openmpi-*
```

F

```
RUN wget -q -O - https://www.open-
mpi.org/software/ompi/v3.0/downloads/openmpi-3.0.0.tar.bz2 | tar -
xjf - && \
    cd openmpi-3.0.0 && \
    CXX=pgc++ CC=pgcc FC=pgfortran F77=pgfortran ./configure --
prefix=/usr/local/openmpi --with-cuda=/usr/local/cuda --with-verbs
--disable-getpwuid && \
    make -j4 install && \
    rm -rf /openmpi-3.0.0
```

HPC CONTAINER MAKER

- Tool for creating HPC application Dockerfiles and Singularity definition files
- Makes it easier to create HPC application containers by encapsulating HPC & container best practices into building blocks
- Open source (Apache 2.0)
  https://github.com/NVIDIA/hpc-container-maker
- `pip install hpccm`

https://devblogs.nvidia.com/making-containers-easier-with-hpc-container-maker/

BUILDING BLOCKS TO CONTAINER RECIPES

```
Stage0 += openmpi()
```

hpccm  Generate corresponding Dockerfile instructions for the HPCCM building block

```
# OpenMPI version 3.0.0
RUN yum install -y \
        bzip2 file hwloc make numactl-devel openssh-clients perl tar wget && \
    rm -rf /var/cache/yum/*
RUN mkdir -p /var/tmp && wget -q -nc --no-check-certificate -P /var/tmp https://www.open-
mpi.org/software/ompi/v3.0/downloads/openmpi-3.0.0.tar.bz2 && \
    mkdir -p /var/tmp && tar -x -f /var/tmp/openmpi-3.0.0.tar.bz2 -C /var/tmp -j && \
    cd /var/tmp/openmpi-3.0.0 &&  CC=gcc CXX=g++ F77=gfortran F90=gfortran FC=gfortran ./configure --
prefix=/usr/local/openmpi --disable-getpwuid --enable-orterun-prefix-by-default --with-cuda=/usr/local/cuda --with-verbs
&& \
    make -j4 && \
    make -j4 install && \
    rm -rf /var/tmp/openmpi-3.0.0.tar.bz2 /var/tmp/openmpi-3.0.0
ENV LD_LIBRARY_PATH=/usr/local/openmpi/lib:$LD_LIBRARY_PATH \
    PATH=/usr/local/openmpi/bin:$PATH
```

HIGHER LEVEL ABSTRACTION

Building blocks to encapsulate best practices, avoid duplication, separation of concerns

```
openmpi(check=False,                                    # run “make check”?
        configure_opts=['--disable-getpwuid', …],       # configure command line options
        cuda=True,                                      # enable CUDA?
        directory='',                                   # path to source in build context
        infiniband=True,                                # enable InfiniBand?
        ospackages=['bzip2', 'file', 'hwloc', …],       # Linux distribution prerequisites
        prefix='/usr/local/openmpi',                    # install location
        toolchain=toolchain(),                          # compiler to use
        version='3.0.0')                                # version to download
```

Examples:

```
openmpi(prefix='/opt/openmpi', version='1.10.7’)
openmpi(infiniband=False, toolchain=pgi.toolchain)
```

Full building block documentation can be found on GitHub

EQUIVALENT HPC CONTAINER MAKER WORKFLOW

python

Login to system (e.g., CentOS 7 with Mellanox OFED 3.4)

```
$ module load PrgEnv/GCC+OpenMPI

$ module load cuda/9.0

$ module load gcc

$ module load openmpi/1.10.7
```

Steps to build application

Result: application binary suitable for that particular bare metal system

```
Stage0 += baseimage(image='nvidia/cuda:9.0-
devel-centos7')
Stage0 += mlnx_ofed(version='3.4-1.0.0.0')

Stage0 += gnu()

Stage0 += openmpi(version='1.10.7')
```

Steps to build application

Result: portable application container capable of running on any system

NVIDIA

INCLUDED BUILDING BLOCKS

As of version 18.10

CUDA is included via the base image, see https://hub.docker.com/r/nvidia/cuda/

- Compilers
  - GNU, LLVM (clang)
  - PGI
  - Intel (BYOL)
- HPC libraries
  - Charm++
  - FFTW, MKL, OpenBLAS
  - CGNS, HDF5, NetCDF, PnetCDF
- Miscellaneous
  - Boost
  - CMake
- InfiniBand
  - Mellanox OFED
  - OFED (upstream)
- MPI
  - OpenMPI
  - MVAPICH2, MVAPICH2-GDR
  - Intel MPI
- Package management
  - packages (Linux distro aware), or
    - apt_get
    - yum

BUILDING APPLICATION CONTAINER IMAGES WITH HPCCM

Application recipe

```
$ cat mpi-bandwidth.py
# Setup GNU compilers, Mellanox OFED, and OpenMPI
Stage0 += baseimage(image='nvidia/cuda:9.0-devel-centos7')
Stage0 += gnu()
Stage0 += mlnx_ofed(version='3.4-1.0.0.0')
Stage0 += openmpi(version='3.0.0')

# Application build steps below
# Using “MPI Bandwidth” from Lawrence Livermore National Laboratory (LLNL) as an example
# 1. Copy source code into the container
Stage0 += copy(src='mpi_bandwidth.c', dest='/tmp/mpi_bandwidth.c')
# 2. Build the application
Stage0 += shell(commands=['mkdir -p /workspace',
                          'mpicc -o /workspace/mpi_bandwidth /tmp/mpi_bandwidth.c’])

$ hpccm --recipe mpi-bandwidth.py --format …
```

BUILDING APPLICATION CONTAINERS IMAGES WITH HPCCM

Application recipes

Dockerfile

```
FROM nvidia/cuda:9.0-devel-centos7
```

CUDA CentOS base image

```
# GNU compiler
RUN yum install -y \
        gcc \
        gcc-c++ \
        gcc-gfortran && \
    rm -rf /var/cache/yum/*
```

GNU compiler

```
# Mellanox OFED version 3.4-1.0.0.0
RUN yum install -y \
        libnl \
        libnl3 \
        numactl-libs \
        wget && \
    rm -rf /var/cache/yum/*
RUN mkdir -p /var/tmp && wget -q -nc --no-check-certificate -P /var/tmp http://content.mellanox.com/ofed/MLNX_OFED-3.4-1.0.0.0/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64.tgz && \
    mkdir -p /var/tmp && tar -x -f /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64.tgz -C /var/tmp -z && \
    rpm --install /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibverbs-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibverbs-devel-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibverbs-utils-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibmad-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibmad-devel-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibumad-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibumad-devel-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libmlx4-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libmlx5-*.x86_64.rpm && \
    rm -rf /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64.tgz /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64
```

Mellanox OFED

```
# OpenMPI version 3.0.0
RUN yum install -y \
        bzip2 \
        file \
        hwloc \
        make \
        openssh-clients \
        perl \
        tar \
        wget && \
    rm -rf /var/cache/yum/*
RUN mkdir -p /var/tmp && wget -q -nc --no-check-certificate -P /var/tmp https://www.open-mpi.org/software/ompi/v3.0/downloads/openmpi-3.0.0.tar.bz2 && \
    mkdir -p /var/tmp && tar -x -f /var/tmp/openmpi-3.0.0.tar.bz2 -C /var/tmp -j && \
    cd /var/tmp/openmpi-3.0.0 &&   ./configure --prefix=/usr/local/openmpi --disable-getpwuid --enable-orterun-prefix-by-default --with-cuda --with-verbs && \
    make -j4 && \
    make -j4 install && \
    rm -rf /var/tmp/openmpi-3.0.0.tar.bz2 /var/tmp/openmpi-3.0.0
ENV LD_LIBRARY_PATH=/usr/local/openmpi/lib:$LD_LIBRARY_PATH \
    PATH=/usr/local/openmpi/bin:$PATH
```

OpenMPI

```
COPY mpi_bandwidth.c /tmp/mpi_bandwidth.c

RUN mkdir -p /workspace && \
    mpicc -o /workspace/mpi_bandwidth /tmp/mpi_bandwidth.c
```

MPI Bandwidth

Singularity definition file

```
BootStrap: docker
From: nvidia/cuda:9.0-devel-centos7
%post
    . /.singularity.d/env/10-docker.sh
```

CUDA CentOS base image

```
# GNU compiler
%post
    yum install -y \
        gcc \
        gcc-c++ \
        gcc-gfortran
    rm -rf /var/cache/yum/*
```

GNU compiler

```
# Mellanox OFED version 3.4-1.0.0.0
%post
    yum install -y \
        libnl \
        libnl3 \
        numactl-libs \
        wget
    rm -rf /var/cache/yum/*
%post
    mkdir -p /var/tmp && wget -q -nc --no-check-certificate -P /var/tmp http://content.mellanox.com/ofed/MLNX_OFED-3.4-1.0.0.0/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64.tgz
    mkdir -p /var/tmp && tar -x -f /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64.tgz -C /var/tmp -z
    rpm --install /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibverbs-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibverbs-devel-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibverbs-utils-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibmad-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibmad-devel-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibumad-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libibumad-devel-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libmlx4-*.x86_64.rpm /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64/RPMS/libmlx5-*.x86_64.rpm
    rm -rf /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64.tgz /var/tmp/MLNX_OFED_LINUX-3.4-1.0.0.0-rhel7.2-x86_64
```

Mellanox OFED

```
# OpenMPI version 3.0.0
%post
    yum install -y \
        bzip2 \
        file \
        hwloc \
        make \
        openssh-clients \
        perl \
        tar \
        wget
    rm -rf /var/cache/yum/*
%post
    mkdir -p /var/tmp && wget -q -nc --no-check-certificate -P /var/tmp https://www.open-mpi.org/software/ompi/v3.0/downloads/openmpi-3.0.0.tar.bz2
    mkdir -p /var/tmp && tar -x -f /var/tmp/openmpi-3.0.0.tar.bz2 -C /var/tmp -j
    cd /var/tmp/openmpi-3.0.0 &&   ./configure --prefix=/usr/local/openmpi --disable-getpwuid --enable-orterun-prefix-by-default --with-cuda --with-verbs
    make -j4
    make -j4 install
    rm -rf /var/tmp/openmpi-3.0.0.tar.bz2 /var/tmp/openmpi-3.0.0
%environment
    export LD_LIBRARY_PATH=/usr/local/openmpi/lib:$LD_LIBRARY_PATH
    export PATH=/usr/local/openmpi/bin:$PATH
%post
    export LD_LIBRARY_PATH=/usr/local/openmpi/lib:$LD_LIBRARY_PATH
    export PATH=/usr/local/openmpi/bin:$PATH
```

OpenMPI

```
%files
    mpi_bandwidth.c /tmp/mpi_bandwidth.c

%post
    mkdir -p /workspace
    mpicc -o /workspace/mpi_bandwidth /tmp/mpi_bandwidth.c
```

MPI Bandwidth

RECIPES INCLUDED WITH CONTAINER MAKER

HPC Base Recipes:

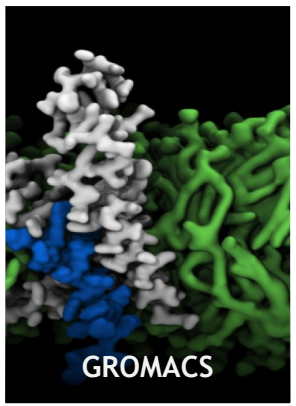
Ubuntu 16.04
CentOS 7

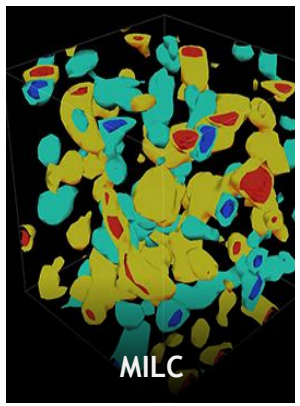
GNU compilers
PGI compilers

OpenMPI
MVAPICH2

CUDA
FFTW
HDF5
Mellanox OFED
Python

Reference Recipes:





easybuild

Spack

MPI
Bandwidth

SUMMARY

- HPC Container Maker simplifies creating a container specification file
  - Best practices used by default
  - Building blocks included for many popular HPC components
  - Flexibility and power of Python
  - Supports Docker (and other frameworks that use Dockerfiles) and Singularity
- Open source: https://github.com/NVIDIA/hpc-container-maker
- `pip install hpccm`


THANK YOU!
NVIDIA