HPCW: NVIDIA

CJ Newburn, Princial Architect for HPC

HPC CONTAINER MAKER

- Motivation
- Problems
- Benefits
- Offerings
- How it works
- Challenges

MOTIVATIONAL STORIES FOR CONTAINERS

War stories from the trenches

Developers:
- Lack of a reference design
  - Many variants, some better than others
- Encapsulating pipelines reduces complexity
- Reproducibility

Admins:
- Hard to configure and install HPC apps
- Better startup times with fewer libs loaded from bottlenecked metadata servers
- Will a given app run on a new platform?

Users:
- App updates get delayed
- Experimental/simulation hybrid molecular modeling as a service

OPENMPI DOCKERFILE VARIANTS

Developers

Real examples: lots of ways, some better than others

```
RUN OPENMPI_VERSION=3.0.0 && \
    wget -q -O - https://www.open-
mpi.org/software/ompi/v3.0/downloads
${OPENMPI_VERSION}.tar.gz | tar -xzf
    cd openmpi-${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_
    echo "/usr/local/mpi/lib" >> /etc
ldconfig
ENV PATH /usr/local/mpi/bin:$PATH
```

Enable many versions with parameters to common interface

Parameters vary

Control environment

```
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
```

Functional, simpler, but not CUDA or IB aware

Different compilers

```
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"
```

```
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-*
```

Bad layering

```
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-*
```

```
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 - HPCCM

“h-p-see-um”

- Collect and codify best practices
- Make recipe file creation easy, repeatable, modular, qualifiable
- Using this as a reference and a vehicle to drive collaboration
- Container implementation neutral

- Write Python code that calls primitives and building blocks vs. roll your own
  - Leverage latest and greatest building blocks

- *“Without this tool it is much less likely that we would have even started using containers for HPC: …more consistent results… easier to share builds … We are using HPCCM with all of our important applications so it is quickly becoming a critical part of our toolchain.”   ~Robert Galetto, PerfLab HPC/DL Manager*

BUILDING BLOCKS TO CONTAINER RECIPES

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

hpccm ⬇ Generate corresponding Dockerfile instructions for the HPCCM building block

```
# OpenMPI version 3.1.2
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/v3.1/downloads/openmpi-3.1.2.tar.bz2 && \
    mkdir -p /var/tmp && tar -x -f /var/tmp/openmpi-3.1.2.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-3.1.2.tar.bz2 /var/tmp/openmpi-3.1.2
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
            ucx=False,                                          # enable UCX?
            version='3.1.2')                                    # 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

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

python™

```
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')
```

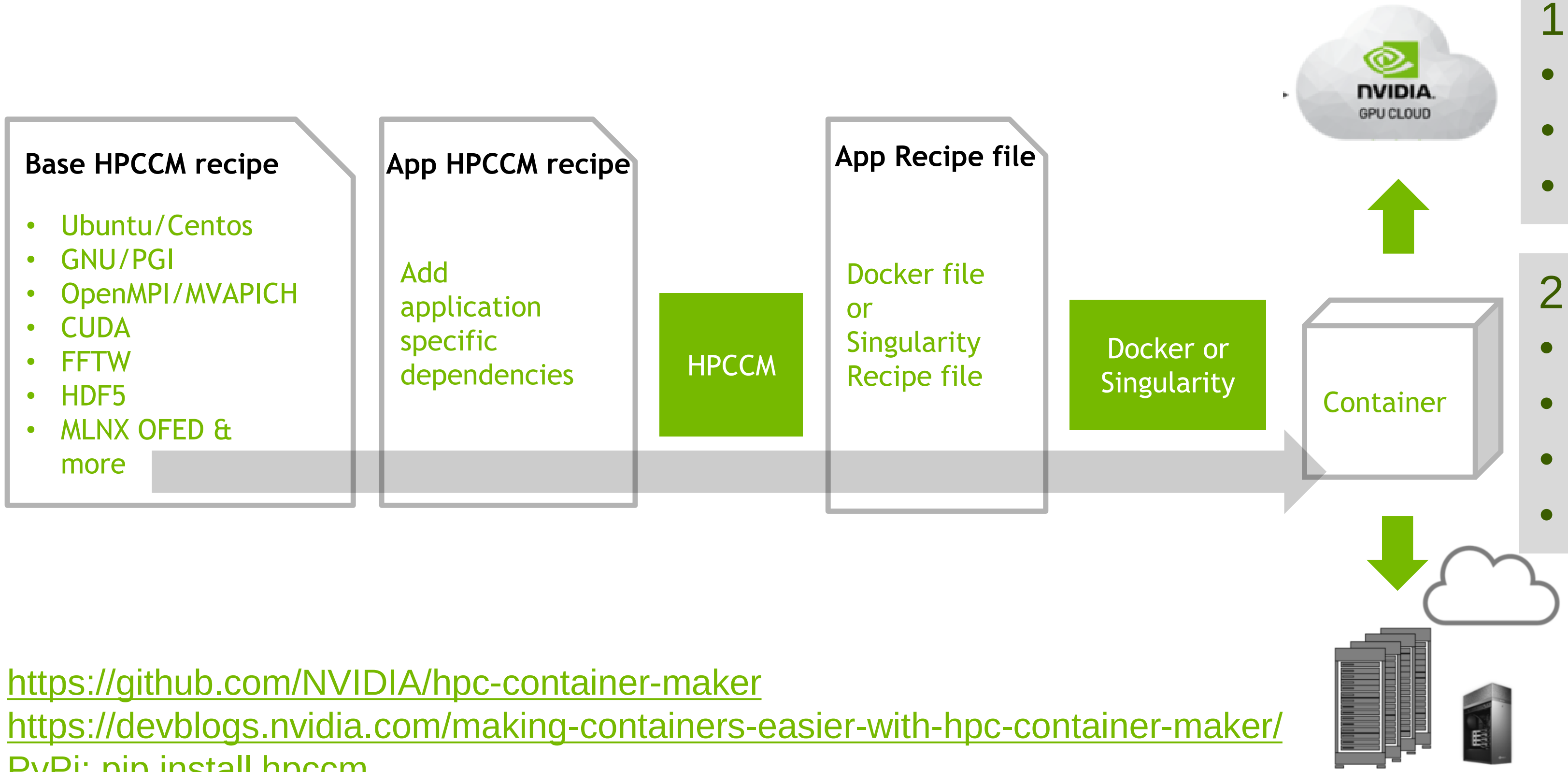
Steps to build application

Result: portable application container capable of running on any system

NVIDIA.

HPC CONTAINER MAKER

SIMPLEST WAY TO BUILD CONTAINERS



1. Follow best practices
   - Layered builds
   - Optimized container size
   - Configuration, deployment

2. Ready-to-use building blocks
   - Simplify efforts
   - Faster builds
   - Higher reliability
   - Optimized performance

https://github.com/NVIDIA/hpc-container-maker
https://devblogs.nvidia.com/making-containers-easier-with-hpc-container-maker/
PyPi: pip install hpccm

RECIPES INCLUDED WITH CONTAINER MAKER

HPC Base Recipes:

Ubuntu 16.04
CentOS 7

✖

GNU compilers
PGI compilers

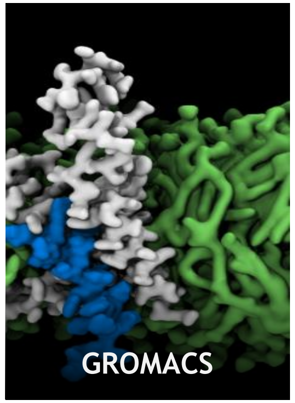
✖

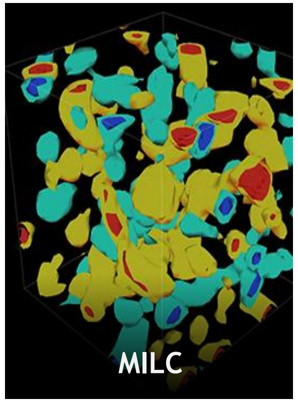
OpenMPI 3.0.0
MVAPICH2 2.3b

+

CUDA 9.0
FFTW 3.3.7
HDF5 1.10.1
NetCDF 4.6.1
Mellanox OFED 3.4-1.0.0.0
Python 2 and 3
Intel MKL
apt_get, yum
cmake…

Reference Recipes:





easybuild

… or create your own …

New since SC18

INCLUDED BUILDING BLOCKS

As of version 19.2

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++, Kokkos
  - FFTW, MKL, OpenBLAS
  - CGNS, HDF5, NetCDF, PnetCDF
- Miscellaneous
  - Boost
  - CMake
  - Python
- Communication libraries
  - Mellanox OFED, OFED (upstream)
  - UCX, gdrcopy, KNEM, XPMEM
- MPI
  - OpenMPI
  - MPICH, MVAPICH2, MVAPICH2-GDR
  - Intel MPI
- Visualization
  - Paraview/Catalyst
- Package management
  - packages (Linux distro aware), or
    - apt_get, yum
  - pip

BUILDING BLOCKS: WHATIF FOR SIERRA…

As of ?

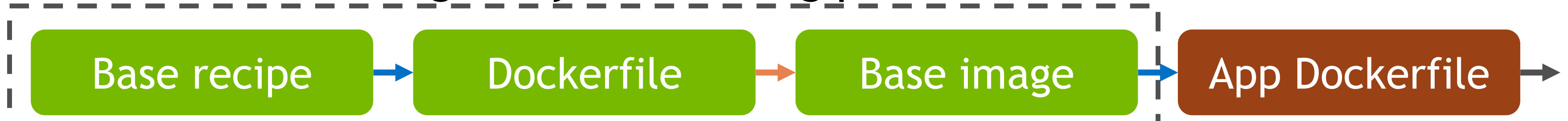
CUDA for POWER is included via the base image, see
https://hub.docker.com/r/nvidia/cuda/

- Compilers
  - GNU
  - PGI (BYOL)
  - XL (BYOL)
- HPC libraries
  - ESSL, PESSL
  - FFTW, OpenBLAS
  - HDF5, NetCDF, PnetCDF
- Miscellaneous
  - Python
  - Boost
  - CMake
- InfiniBand
  - Mellanox OFED
  - OFED (upstream)
- MPI
  - SpectrumMPI
  - OpenMPI
- Package management
  - packages (Linux distro aware), or
    - apt_get
    - Yum
  - Easybuild
  - SPACK

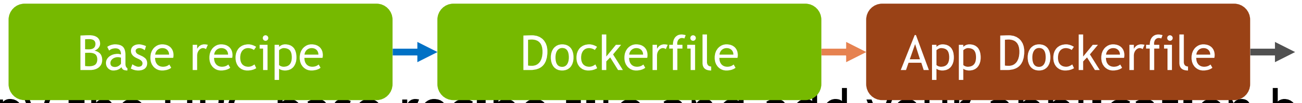
BUILDING AN HPC APPLICATION IMAGE

Analogous workflows for Singularity

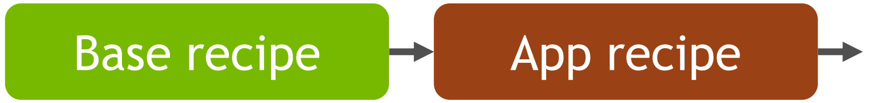
1. Use the HPC base image as your starting point



2. Generate a Dockerfile from the HPC base recipe Dockerfile and manually edit it to add the steps to build your application



3. Copy the HPC base recipe file and add your application build steps to the recipe



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

What's new

MULTI-NODE HPC CONTAINERS

Validated support that grows over time

Trend	Validated support
Shared file systems	Mount into container from host
Advanced networks	InfiniBand
GPUs	P100, V100
MPI is common	OpenMPI (3.0.1+ on host)
Container runtimes	Docker images, trivially convertible to Singularity (v2.5+, blog)
Resource management	SLURM (14.03+), PBS Pro - sample batch scripts
Parallel launch	Slurm srun, host mpirun, container mpirun/charmrun
Reduced size (unoptimized can be 1GB+)	Highly optimized via HPCCM (Container Maker) LAMMPS is 100MB; most under 300MB

What's new

MULTI-NODE CONTAINERS: OPENMPI ON UCX

A preferred layering

- Supports optimized CPU & GPU copy mechanisms when on host
  - CMA, KNEM, XPMEM, gdrcopy (nv_peer_mem)
- OFED libraries used by default
  - Tested for compatibility with MOFED 3.x,4.x host driver versions
- MOFED libraries enabled when version 4.4+ detected
  - Mellanox "accelerated" verbs transports available when enabled

HPCCM SUMMARY

Making the build process easier, more consistent, more updatable

- 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`

- Refer to this code for NVIDIA's best practices
- HPCCM input recipes are starting to be included in images posted to registry