SIMPLIFYING AI, HPC, AND VISUALIZATION WORKFLOWS WITH GPU-OPTIMIZED CONTAINERS FROM NGC

Chintan Patel - Senior Product Marketing Manager
Scott McMillan - Senior Solutions Architect
DIFFERENT ROLES. SAME GOALS.
Driving Productivity and Faster Time-to-Solutions

Data Scientists and Researchers
- Eliminate mundane tasks, focus on science and research

Developers
- Speed up development with existing building blocks

DevOps
- Consistent & faster develop-to-production cycle

Sysadmins
- Deliver appropriate deployment environments
NGC: GPU-OPTIMIZED SOFTWARE HUB
Simplifying DL, ML and HPC Workflows

Model Training Scripts
NLP, Image Classification, Object Detection & more

Containers
DL, ML, HPC

Pre-trained Models
NLP, Classification, Object Detection & more

Helm Charts
AI applications, K8s cluster, Registry

Industry SDKs
Medical Imaging, Intelligent Video Analytics

HPC
NAMD | GROMACS | ParaView | IndeX | more

DEEP LEARNING
TensorFlow | PyTorch | RAPIDS | H2O | more

MACHINE LEARNING
CONTAINERS: SIMPLIFYING WORKFLOWS

WHY CONTAINERS

Simplifies Deployments
- Eliminates complex, time-consuming builds and installs

Portable
- Deploy across various environments

Get started in minutes
- Simply Pull & Run

Reproducible Results
- Constant environment provides reproducible results
NGC CONTAINERS: ACCELERATING WORKFLOWS

WHY NGC CONTAINERS

Optimized for Performance
- Monthly DL container releases offer latest features and superior performance on NVIDIA GPUs

Scalable Performance
- Supports multi-GPU & multi-node systems for scale-up & scale-out environments

Designed for Enterprise & HPC environments
- Supports Docker & Singularity runtimes

Run Anywhere
- Pascal/Volta/Turing-powered NVIDIA DGX, PCs, workstations, and servers
- On-Prem to Hybrid to Cloud
CONTINUOUS PERFORMANCE IMPROVEMENT

Developers’ Software Optimizations Deliver Better Performance on the Same Hardware

Monthly DL Framework Updates & HPC Software Stack Optimizations Drive Performance

<table>
<thead>
<tr>
<th>MxNet</th>
<th>PyTorch</th>
<th>TensorFlow</th>
<th>HPC Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Images/Second</td>
<td>Tokens/Second</td>
<td>Images/Second</td>
</tr>
<tr>
<td>19.03</td>
<td>19.10</td>
<td>19.03</td>
<td>19.10</td>
</tr>
<tr>
<td>17000</td>
<td>17500</td>
<td>18000</td>
<td>18500</td>
</tr>
</tbody>
</table>

256 Batch Size | ResNet-50 Training v1.5 | 16x V100 | DGX-2

512 Batch Size | ResNet-50 Training v1.5 | 16x V100 | DGX-2

512 Batch Size | ResNet-50 Training v1.5 | 16x V100 | DGX-2

Speedup across Chroma, GROMACS, LAMMPS, QE, MILC, VASP, SPECFEM3D, NAMD, AMBER, GTC, RTM | 4x V100 v. Dual-Skylake | CUDA 9 for Mar ’18 & Nov ’18, CUDA 10 for Mar’19
ANNOUNCING: HPC CONTAINERS FOR ARM

- HPC | Visualization | DL
- Runs in Singularity
- EA Now - ngc.nvidia.com
- GA Q1’20
ANNOUNCING: NGC CONTAINERS RUN IN SINGULARITY

Single command to

1. Pull
2. Convert to Singularity format
3. Run

* HPC & Visualization containers
* DL frameworks
  - TensorFlow, PyTorch, MxNet, NV caffe
  - Beta support starting with v19.11
* Supports Singularity v3.x

`singularity shell docker://nvcr.io/hpc/namd:2.13-multinode`
NGC CONTAINER REPLICATOR
Local Access to the Latest Images

- Maintain a local replica of NGC containers
  - Single, shared copy of NGC container images
  - Automatically download when new or updated NGC containers are published
  - Filter by image and/or date
  - Bridge air-gapped environments
- Github | Blog

Ready to run on Docker & Singularity
CONSISTENT DEPLOYMENT ACROSS PLATFORMS

Edge | Data Center | Cloud

**EDGE**

- NGC CONTAINERS
- K8S/OPENSIGHT (optional)
- CONTAINER RUNTIME
- NVIDIA DRIVER
- OPERATING SYSTEM
- GPU-POWERED SYSTEM

Analyze field sensor data in real-time

**DATA CENTER**

- NGC CONTAINERS
- K8S/OPENSIGHT (optional)
- CONTAINER RUNTIME
- NVIDIA DRIVER
- OPERATING SYSTEM
- GPU-POWERED SYSTEM

Build large models from Edge data or simulations

**CLOUD**

- NGC CONTAINERS
- KUBERNETES
- CONTAINER RUNTIME
- OPERATING SYSTEM
- NVIDIA DRIVER
- GPU-POWERED INSTANCE

Burst to the cloud
NGC GOES MAINSTREAM
200+ SUPERCOMPUTING CENTERS & 800+ UNIVERSITIES

“NGC allows us to support our users with the latest AI frameworks and the users enjoy the best performance they can achieve on NVIDIA GPUs.”

Dr. Hirotaka Ogawa, Team Leader
ABCI
WHAT'S NEW WITH HPC CONTAINER MAKER
BUILDING YOUR OWN CONTAINER IMAGES
Or What To Do If Your Application Is Not Available On NGC

Scientists and Researchers
Realize the benefits of containers (portability, reproducibility) to simplify application deployments on HPC systems

Application Developers
Distribute your software as a container to simplify life for your users and reduce your support overhead

IT Staff
Support a workload that is incompatible with the host software environment or tame environment module complexity

HPC Container Maker is an open source project addressing the challenges of building container images for HPC applications
HPCCM BY EXAMPLE: GROMACS

gromacs_version = USERARG.get('gromacs', '2019.4')

# Build stage
Stage0 += baseimage(image='nvidia/cuda:10.0-devel-ubuntu16.04')
Stage0 += python(python3=False)
Stage0 += gnu(fortran=False)
Stage0 += cmake(eula=True)
Stage0 += ofed()
Stage0 += openmpi(version='3.1.4')
Stage0 += generic_cmake(cmake_opts=[
' -D CMAKE_BUILD_TYPE=Release',
' -D CUDA_TOOLKIT_ROOT_DIR=/usr/local/cuda',
' -D GMX_BUILD_OWN_FFTW=ON',
' -D GMX_GPU=ON',
' -D GMX_MPI=ON',
' -D GMX_OPENMP=ON',
' -D GMX_PREFER_STATIC_LIBS=ON',
' -D MPIEXEC_PREFLAGS=--allow-run-as-root'],
prefix='/usr/local/gromacs',
url='http://ftp.gromacs.org/pub/gromacs/gromacs-{}.tar.gz'.format(gromacs_version))

# Distributable stage
Stage1 += baseimage(image='nvidia/cuda:10.0-runtime-ubuntu16.04')
Stage1 += Stage0.runtime()
Stage1 += environment(variables={'PATH': '$PATH:/usr/local/gromacs/bin'})

$ hpccm --recipe gromacs.py --format docker > Dockerfile
$ sudo docker build -t gromacs:2019.4 -f Dockerfile.
$ singularity build gromacs-2019.4.sif docker-daemon://gromacs:2019.4
FROM ubuntu:18.04
# Python
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
python \
pip \
pipenv \
python3 \
python3-pip \\
# OFED
FROM nvidia
# CUDA TOOLKIT
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
apt-utils \
rdmacm \
librdmacm1 \
librdmacm-dev \
libmlx4 \
libmlx4-ctools \
libibmad5 \
libibcm1 \
dapl2 \
dapl-utils \
dapivy \
diags \
diags-dev \
emcl 

FROM ubuntu:18.04
# Openmpi
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
mpich \
mpich-dev \
mpich-doc \
mpich-utils \
mpich-y &&

FROM ubuntu:18.04
# Gromacs 2019.4
run-rf /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \
wget https://ftp.gromacs.org/pub/gromacs/5.1.4.tar.gz /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \
wget https://ftp.gromacs.org/pub/gromacs/5.0.6.tar.gz /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \

FROM nvidia:1804
# GROMACS
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
python3 \
pip3 \
pip3env \
python3-pip \
pip
# OFED
FROM nvidia
# CUDA TOOLKIT
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
apt-utils \
rdmacm \
librdmacm1 \
libmlx4 \
libmlx4-ctools \
libibmad5 \
libibcm1 \
dapl2 \
dapl-utils \
dapivy \
diags \
diags-dev \
emcl 

FROM ubuntu:18.04
# Openmpi
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
mpich \
mpich-dev \
mpich-doc \
mpich-utils \
mpich-y &&

FROM ubuntu:18.04
# Gromacs 2019.4
run-rf /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \
wget https://ftp.gromacs.org/pub/gromacs/5.1.4.tar.gz /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \
wget https://ftp.gromacs.org/pub/gromacs/5.0.6.tar.gz /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \

FROM nvidia:1804
# GROMACS
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
python3 \
pip3 \
pip3env \
python3-pip \
pip
# OFED
FROM nvidia
# CUDA TOOLKIT
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
apt-utils \
rdmacm \
librdmacm1 \
libmlx4 \
libmlx4-ctools \
libibmad5 \
libibcm1 \
dapl2 \
dapl-utils \
dapivy \
diags \
diags-dev \
emcl 

FROM ubuntu:18.04
# Openmpi
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
mpich \
mpich-dev \
mpich-doc \
mpich-utils \
mpich-y &&

FROM ubuntu:18.04
# Gromacs 2019.4
run-rf /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \
wget https://ftp.gromacs.org/pub/gromacs/5.1.4.tar.gz /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \
wget https://ftp.gromacs.org/pub/gromacs/5.0.6.tar.gz /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \

FROM nvidia:1804
# GROMACS
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
python3 \
pip3 \
pip3env \
python3-pip \
pip
# OFED
FROM nvidia
# CUDA TOOLKIT
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
apt-utils \
rdmacm \
librdmacm1 \
libmlx4 \
libmlx4-ctools \
libibmad5 \
libibcm1 \
dapl2 \
dapl-utils \
dapivy \
diags \
diags-dev \
emcl 

FROM ubuntu:18.04
# Openmpi
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
mpich \
mpich-dev \
mpich-doc \
mpich-utils \
mpich-y &&

FROM ubuntu:18.04
# Gromacs 2019.4
run-rf /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \
wget https://ftp.gromacs.org/pub/gromacs/5.1.4.tar.gz /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \
wget https://ftp.gromacs.org/pub/gromacs/5.0.6.tar.gz /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \

FROM nvidia:1804
# GROMACS
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
python3 \
pip3 \
pip3env \
python3-pip \
pip
# OFED
FROM nvidia
# CUDA TOOLKIT
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
apt-utils \
rdmacm \
librdmacm1 \
libmlx4 \
libmlx4-ctools \
libibmad5 \
libibcm1 \
dapl2 \
dapl-utils \
dapivy \
diags \
diags-dev \
emcl 

FROM ubuntu:18.04
# Openmpi
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
mpich \
mpich-dev \
mpich-doc \
mpich-utils \
mpich-y &&

FROM ubuntu:18.04
# Gromacs 2019.4
run-rf /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \
wget https://ftp.gromacs.org/pub/gromacs/5.1.4.tar.gz /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \
wget https://ftp.gromacs.org/pub/gromacs/5.0.6.tar.gz /var/lib/apt/lists/ \
run-rf /var/lib/apt/lists/ \

FROM nvidia:1804
# GROMACS
RUN apt-get update -y &&
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
python3 \
pip3 \
pip3env \
python3-pip \
pip
EASIER THAN EVER TO GET STARTED

- HPCCM is now available from Anaconda
  
  conda install -c conda-forge hpccm

- And continues to be available from PyPi
  
  pip install hpccm

- Simple to use command line interface
  
  hpccm --recipe recipe.py --format docker || singularity

- Can also be used in a Python script
  
  import hpccm
INCLUDED BUILDING BLOCKS

As of version 19.11

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

- Compilers
  - GNU, LLVM (clang)
  - PGI
  - Arm, Intel (BYOL)
- HPC libraries
  - Charm++, Kokkos
  - FFTW, MKL, OpenBLAS
  - CGNS, HDF5, NetCDF, PnetCDF
- Miscellaneous
  - Boost
  - CMake
  - Julia, Python
  - Generic GNU autotools or CMake package builder

- Communication libraries
  - Mellanox OFED, OFED (upstream)
  - UCX, gdrcopy, KNEM, XPMEM
- MPI
  - OpenMPI
  - MPICH, MVAPICH2, MVAPICH2-GDR
  - Intel MPI
- Visualization
  - Paraview/Catalyst, VisIT/Libsim, SENSEI
- Package management
  - packages (Linux distro aware), or
    - apt_get, yum
  - Anaconda, pip
  - Scientific Filesystem (SCI-F)
COMMUNICATION BUILDING BLOCKS

- UCX will automatically use the best available communication paths
  - Supports InfiniBand, GDRCopy, CUDA IPC, and more
  - Default transport in OpenMPI 4.0
- Matching the InfiniBand user space library to the host InfiniBand driver
  - New `multi_ofed` building block installs multiple versions of Mellanox OFED and OFED packaged by the Linux distribution in a non-default location
  - Use a custom container entry point to select the "best" OFED match at container run time and configure the container environment
ONE RECIPE, MULTIPLE TARGETS

New! CentOS 8 & Ubuntu 18.04

Linux distribution

New! bash & Singularity multi-stage

HPCCM recipe

Container format
ONE RECIPE, MULTIPLE TARGETS

Linux distribution
- CentOS 8 & Ubuntu 18.04
- Experimental!

Processor architecture
(x86-64, Power, Arm)

Container format
- Docker
- Singularity multi-stage

HPCCM recipe

New! bash & Singularity multi-stage
# Use appropriate container base images based on the CPU architecture
arch = USERARG.get('arch', 'x86_64')
if arch == 'aarch64':
    # Early Access images - NGC registration required to use
    default_build_image = 'nvcr.io/ea-cuda-sc19/arm-partners/cuda-aarch64:10.2-devel-ubuntu18.04'
    default_runtime_image = 'nvcr.io/ea-cuda-sc19/arm-partners/cuda-aarch64:10.2-base-ubuntu18.04'
elif arch == 'x86_64':
    default_build_image = 'nvidia/cuda:10.1-devel-ubuntu18.04'
    default_runtime_image = 'nvidia/cuda:10.1-base-ubuntu18.04'
else:
    raise Exception('unrecognized architecture: {}' .format(arch))

# Build stage (Stage 0)
Stage0 += baseimage(image=USERARG.get('build_image', default_build_image), _arch=arch, _as='build')

# Base development environment
Stage0 += gnu(version='8')
Stage0 += cmake(eula=True)

# Communication stack: OpenMPI + UCX + KNEM + Mellanox OFED + gdrcopy (x86 only)
Stage0 += mlnx_ofed(version=USERARG.get('mlnx_ofed', '4.6-1.0.1.1'))

if hpccm.config.g_cpu_arch == hpccm.config.cpu_arch.X86_64:
    Stage0 += gdrcopy(ldconfig=True, version=USERARG.get('gdrcopy', '1.3'))

Stage0 += knem(ldconfig=True, version=USERARG.get('knem', '1.1.3'))
Stage0 += ucx(knem='/usr/local/knem', ldconfig=True, version=USERARG.get('ucx', '1.6.1'))

mpi = openmpi(ldconfig=True, version=USERARG.get('ompi', '4.0.2'), ucx='/usr/local/ucx')
Stage0 += mpi
# LAMMPS

gpu_arch = USERARG.get('gpu_arch', 'Volta70')
if gpu_arch not in ['Pascal60', 'Volta70', 'Turing75']:
    raise Exception('unrecognized GPU architecture: {}'.format(gpu_arch))

lammps_version = USERARG.get('lammps_version', 'patch_19Sep2019')
srcdir = '/var/tmp/lammps-{}'.format(lammps_version)

Stage0 += comment('LAMMPS version {} for CUDA compute capability {}'.format(lammps_version, compute_capability))

Stage0 += apt_get(ospackages=['bc', 'git', 'libgomp1', 'libhwloc-dev', 'make', 'tar', 'wget'])

Stage0 += generic_cmake(build_directory='{}/build-{}'.format(srcdir, gpu_arch),
cmake_opts=['-D BUILD_SHARED_LIBS=ON',
              '-D CUDA_USE_STATIC_CUDA_RUNTIME=OFF',
              '-D KOKKOS_ARCH={}'.format(gpu_arch),
              '-D CMAKE_BUILD_TYPE=Release',
              '-D MPI_C_COMPILER={}'.format(mpi.toolchain.CC),
              '-D BUILD_MPI=yes', '-D PKG_MPIIO=on',
              '-D BUILD_OMP=yes', '-D BUILD_LIB=no',
              '-D CMAKE_CXX_COMPILER={}lib/kokkos/bin/nvcc_wrapper'.format(srcdir),
              '-D PKG_USER_REAXX=yes', '-D PKG_KSPACE=yes',
              '-D PKG_MOLECULE=yes', '-D PKG_REPLICA=yes',
              '-D PKG_RIGID=yes', '-D PKG_MISC=yes',
              '-D PKG_MANYBODY=yes', '-D PKG_ASPHERE=yes',
              '-D PKG_GPU=no', '-D PKG_KOKKOS=yes',
              '-D KOKKOS_ENABLE_CUDA=yes', '-D KOKKOS_ENABLE_HWLOC=yes'],
directory='{}/cmake'.format(srcdir),

# Force CUDA dynamic linking, see https://github.com/openpmd/ucx/wiki/NVIDIA-GPU-Support
preconfigure=['sed -i "s/^cuda_args=\""/cuda_args=--cudart shared"/g" {}/lib/kokkos/bin/nvcc_wrapper'.format(srcdir),
              'prefix="/usr/local/lammps-{}'.format(compute_capability),
url='https://github.com/lammps/lammps/archive/{}.tar.gz'.format(lammps_version))
# Distributable stage (Stage 1)
Stage1 += baseimage(image=USERARG.get('runtime_image', default_runtime_image))

# Build stage runtime support + LAMMPS
Stage1 += Stage0.runtime()

# LAMMPS environment
Stage1 += environment(variables={
    'LD_LIBRARY_PATH': '/usr/local/lammps-{}/lib:$LD_LIBRARY_PATH'.format(compute_capability),
    'PATH': '/usr/local/lammps-{}/bin:$PATH'.format(compute_capability),
    # Workaround, see https://github.com/openucx/ucx/wiki/NVIDIA-GPU-Support
    'UCX_MEMTYPE_CACHE': 'n'
})

```bash
hpccm --recipe lammps.py --userarg arch=x86_64 --gpu_arch=Turing75

hpccm --recipe lammps.py --userarg arch=aarch64 --gpu_arch=Volta70 --lammps_version=patch_30Oct2019

sudo docker build -t lammps:patch_19Sep2019 -f Dockerfile .
singularity build lammps.sif docker-daemon://lammps:patch_19Sep2019

hpccm --recipe lammps.py --format singularity --singularity-version 3.2

sudo singularity build lammps.sif Singularity.def
```
OTHER NEW HPCCM FEATURES

▶ Sample recipe to easily containerize Jupyter notebooks
  
jupyter.py --notebook nb.ipynb --requirements requirements.txt --format docker

▶ Multi-stage recipes with Singularity version 3.2 and later

▶ Improvements to existing building blocks
  
  ▶ OpenACC enabled GNU compilers
  
  ▶ Regularly refresh default component versions
SUMMARY

NGC drives productivity and faster innovations

- Simplifies HPC, DL, ML application deployments
- Continuous performance improvement with latest software on the same hardware
- Support Singularity | Any GPU-powered x86 / Arm system

Use NGC containers whenever possible, but if you need to build your own containers, HPC Container Maker makes it easier

- Reduce development effort by using building blocks
- Maintain a single recipe that supports multiple targets
- Minimize image size by using multi-stage containers