Evaluating Shifter for HPC Applications
Don Bahls – Cray Inc.
Agenda

● Motivation
  ● Shifter User Defined Images (UDIs) provide a mechanism to access a wider array of software in the HPC environment without enduring the developer overhead of pulling in new and/or alternate dependencies not easily targeted to the Cray SLES environment. We compare the performance of applications with existing ports to the Cray to look at developer overhead, performance as well as other factors.

● Overview of Shifter
● Container Techniques Utilized
● Performance Comparison
● Summary
  ● Evidence suggests that UDI can provide performance that is competitive with natively compiled applications, opening up the possibility of a wider range of MPI-based application with minimal drawbacks.

● Q&A
Basic Container Overview

Traditional Virtual Machines

- Application
- Software Stack
- VM OS
- Hypervisor
- Host OS
- System

Linux Containers

- Application
- Software Stack
- Host OS
- System
Overview of Shifter

Build Docker Image of Application

DockerHub or Private Registry

Developer uploads Docker Image

Gateway queries DockerHub for image

User requests Docker Image

Gateway builds UDI if necessary

Requests Image

Mounts UDI Image on each Compute node

User Batch Job

Compute Nodes

user@cray% getDockerImage pull img:latest
Applications Used in Experiments

- Applications chosen based on library dependencies and use of MPI.
- All had existing Cray ports to ensure performance comparison could be made between native and UDI versions.
- Run on Cray XC or Cray XE systems.

<table>
<thead>
<tr>
<th>Application</th>
<th>Libraries/Build Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>PISM (C++ code)</td>
<td>BLAS, cmake, FFTW3, GSL, LAPACK, MPI, NetCDF, PETSc</td>
</tr>
<tr>
<td>POP2 (F90 code)</td>
<td>MPI, NetCDF</td>
</tr>
<tr>
<td>Quantum Espresso (F90 / C code)</td>
<td>BLAS, FFTW3, LAPACK, MPI, ScaLAPACK</td>
</tr>
<tr>
<td>IMB (C code)</td>
<td>MPI</td>
</tr>
<tr>
<td>IOR (C code)</td>
<td>MPI with MPI/IO support</td>
</tr>
</tbody>
</table>
Container Techniques Utilized

MPICH-ABI UDI Method
- Application
- Other Dependencies
- Build MPICH-3.1+
- OS Provided Dependencies
- Base UDI Docker Image (e.g. Ubuntu:latest)

Cray MPI UDI Method
- Application
- Other Dependencies
- MPI compiler wrappers
- Cray IP from /opt/cray
- OS Provided Dependencies
- Base UDI Docker Image (e.g. Ubuntu:latest)
FROM ubuntu:16.04
RUN apt-get update -y && 
    apt-get install -y 
    fortran g++ libfftw3-dev libopenblas-base 
    libopenblas-dev liblapack-dev make cmake 
    wget
RUN mkdir -p /app
RUN mkdir -p /app/local
ENV DIR /app/local
ENV CC gcc
ENV CXX g++
ENV PATH $DIR/bin:$PATH
ENV QE_DIR /app/local/espresso-5.3.0
ADD src/mpich-3.2.tar.gz /app/local
ADD src/scalapack-2.0.2.tar.gz /app/local
ADD src/espresso-5.3.0.tar.gz /app/local
RUN cd /app/local/mpich-3.2 && 
    ./configure --prefix=/app/local/bin 
    --with-scalapack=yes 
    FFT_LIBS="-L/usr/lib/x86_64-linux-gnu -lfftw3" && 
    make all && 
    make install && 
    rm -r /app/local/scalapack-2.0.2 && 
    rm -r /app/local/mpich-3.2 && 
    rm -r /app/local/espresso-5.3.0
RUN cd /app/local/scalapack-2.0.2 && 
    cmake -DCMAKE_INSTALL_PREFIX=/app/local . && 
    make && 
    make install
ENV F90 gfortran
ENV MPIF90 mpif90
RUN cd $QE_DIR && 
    ./configure --prefix=/app/local/bin 
    --with-scalapack=yes 
    FFT_LIBS="-L/usr/lib/x86_64-linux-gnu -lfftw3" && 
    make all && 
    make install && 
    rm -r /app/local/scalapack-2.0.2 && 
    rm -r /app/local/mpich-3.2 && 
    rm -r /app/local/espresso-5.3.0
% docker build -t espresso:latest -f Dockerfile
# push image to registry or export image
#!/bin/bash
#PBS -v UDI=espresso:latest
#PBS -l walltime=8:00:00
#PBS -l nodes=16:ppn=16
#PBS -j oe

cd $PBS_O_WORKDIR
module load shifter
module unload PrgEnv-cray
module unload cce
module load PrgEnv-gnu
module unload cray-mpich
module load cray-mpich-abi
APP=/app/local/bin/pw.x
INPUT=$PWD/ausurf.in

CACHE=$PWD/cache
LIBS=$CRAY_LD_LIBRARY_PATH:/opt/cray/wlm_detect/default/lib64

# workaround when /opt/cray is not mounted in the UDI
mkdir -p $CACHE
for dir in $( echo $LIBS | tr ":" " " ); do
cp -L -r $dir $CACHE
Done

export LD_LIBRARY_PATH=$CACHE
export CRAY_ROOTFS=UDI
aprun -n 256 -b $APP -i $INPUT
% aprun -n 1 -b ldd /app/local/IMB/IMB-3.2/RUN/IMB-MPICH-ABI.msg22
    linux-vdso.so.1 => (0x00002aaaaaaab000)
    libmpi.so.12 => /app/local/lib/libmpi.so.12 (0x00002aaaaaaab000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00002aaaaaaaf3d000)
    libcr.so.0 => /usr/lib/libcr.so.0 (0x00002aaaab308000)
...
    libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00002aaaabc8000)
Application 19109760 resources: utime ~0s, stime ~0s, Rss ~3956, inblocks ~45, outblocks ~4

% export LD_LIBRARY_PATH=/lus/dal/dmb/cache/lib/:/lus/dal/dmb/cache/lib64
% aprun -n 1 -b ldd /app/local/IMB/IMB-3.2/RUN/IMB-MPICH-ABI.msg22
    linux-vdso.so.1 => (0x00002aaaaaaab000)
    libmpi.so.12 => /lus/dal/dmb/cache/lib/libmpi.so.12 (0x00002aaaaaaae000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00002aaaab040000)
    libxpmem.so.0 => /lus/dal/dmb/cache/lib64/libxpmem.so.0 (0x00002aaaab40a000)
    librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00002aaaab60d000)
    libugni.so.0 => /lus/dal/dmb/cache/lib64/libugni.so.0 (0x00002aaaab816000)
    libudreg.so.0 => /lus/dal/dmb/cache/lib64/libudreg.so.0 (0x00002aaaaba89000)
    libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00002aaaabc92000)
    libpmi.so.0 => /lus/dal/dmb/cache/lib64/libpmi.so.0 (0x00002aaaabeceb000)
...
    libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00002aaaac98e000)
FROM ubuntu:16.04
RUN apt-get update -y && \
    apt-get install -y \
    gfortran g++ libfftw3-dev \
    libopenblas-base libopenblas-dev \
    liblapack-dev make wget cmake
RUN mkdir -p /app
RUN mkdir -p /app/local
ENV DIR /app/local
ENV CC gcc
ENV CXX g++
ENV PATH $DIR/bin:$PATH
ENV QE_DIR /app/local/espresso-5.3.0
ADD src/optcray.gem.tar.gz /
ADD src/wrappers.tar.gz /
ADD src/espresso-5.3.0.tar.gz /app/local
ENV F90 gfortran
ENV MPIF90 mpif90

RUN cd $QE_DIR && \
    ./configure --prefix=/app/local/bin \
    --with-scalapack=yes \
    FFT_LIBS="-L /usr/lib/x86_64-linux-gnu -lfftw3" && \
    make all && \
    make install && \
    rm -r /app/local/espresso-5.3.0 && \
    printf "'/opt/cray/mpt/default/gni/mpich-gnu/5.1/lib\n" >> /etc/ld.so.conf && \
    printf "'/opt/cray/dmapp/default/lib64\n" >> /etc/ld.so.conf && \
    printf "'/opt/cray/ugni/default/lib64\n" >> /etc/ld.so.conf && \
    printf "'/opt/cray/udreg/default/lib64\n" >> /etc/ld.so.conf && \
    printf "'/opt/cray/pmi/default/lib64\n" >> /etc/ld.so.conf && \
    printf "'/opt/cray/xpmem/default/lib64" >> /etc/ld.so.conf && \
    ldconfig

% docker build -t espresso:latest -f Dockerfile
# push image to registry or export image
#!/bin/bash
#PBS -v UDI=espresso:latest
#PBS -l walltime=8:00:00
#PBS -l nodes=16:ppn=16
#PBS -j oe

cd $PBS_O_WORKDIR

module load shifter

APP=/app/local/bin/pw.x
INPUT=$PWD/ausurf.in

export CRAY_ROOTFS=UDI
aprun -n 256 -b $APP -i $INPUT
% aprun -n -b ldd /app/local/bin/pw.x
  linux-vdso.so.1 => (0x00007ffcd6141000)
  libopenblas.so.0 => /usr/lib/x86_64-linux-gnu/libopenblas.so.0 (0x00007f2184803000)
  libfftw3.so.3 => /usr/lib/x86_64-linux-gnu/libfftw3.so.3 (0x00007f2184405000)
  libmpich_gnu_51.so.3 => /opt/cray/mpt/default/gnu/mpich-gnu/5.1/lib/libmpich_gnu_51.so.3 (0x00007f2183b4c000)
  libgfortran.so.3 => /usr/lib/x86_64-linux-gnu/libgfortran.so.3 (0x00007f2184803000)
  libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f218392f000)
  libxpmem.so.0 => /opt/cray/xpmem/default/lib64/libxpmem.so.0 (0x00007f218372c000)
  libugni.so.0 => /opt/cray/ugni/default/lib64/libugni.so.0 (0x00007f2183d7000)
  libudreg.so.0 => /opt/cray/udreg/default/lib64/libudreg.so.0 (0x00007f21832ce000)
  libpmi.so.0 => /opt/cray/pmi/default/lib64/libpmi.so.0 (0x00007f2183092000)
  libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f2182d89000)
  libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f2182b73000)
  libquadmath.so.0 => /app/local/lib/x86_64-linux-gnu/libquadmath.so.0 (0x00007f2182934000)
  libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f2182256b000)
  librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007f2182363000)
  libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f2181fe1000)
  /lib64/ld-linux-x86-64.so.2 (0x00007f2186950000)
  libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f2181ddd000)
Operating Environment

Cray XC30

- **Compute Nodes**
  - 116- 20 core/3.0 GHz Intel IVB
  - 116- 24 core/2.7 GHz Intel IVB
  - 20- 24 core/2.7 GHz Intel IVB

- **Other Details**
  - Moab 8.1.1.2 / Torque 5.1.1.2
  - Sonexion 2000 / NEO-2.0.0
  - CLE-5.2UP04
  - Aries Network

Cray XE6/XK7

- **Compute Nodes**
  - 100- 16 core/2.1 GHz AMD Interlagos
  - 280- 32 core/2.1 GHz AMD Interlagos
  - 96- 32 core/2.5 GHz AMD Abu Dhabi

- **Other Details**
  - PBS Professional 12.2.204
  - Direct Attached Lustre
  - CLE-5.2UP04
  - Gemini Network
Performance Comparison

- Applications using MPICH-ABI UDI, Cray MPI UDI and natively compiled were run.

- Metrics Measured
  - Application Timing– at various processor counts
  - Startup overhead– average for each technique
    - Calculated with batch scheduler logs and a time stamp within the batch script.
    - Times measured in integer seconds due to the method used.
Espresso AUSURF112 - Test Case Timing

![Graph showing runtime vs core count for different MPI implementations.]

- MPICH-ABI UDI (GCC Opt)
- Cray MPI UDI (GCC Opt)
- Cray Native (CCE Opt dynamic)
- Cray Native (CCE Opt static)
PISM - Test Case Timing

Runtime (s) vs Core Count for different MPI implementations:
- MPICH-ABI UDI (GCC Opt)
- Cray MPI UDI (GCC Opt)
- Cray Native (GCC Opt dynamic)
POP2 – 30 Day Test Case Timing
POP2 – 30 Day Test Case Startup Overhead

<table>
<thead>
<tr>
<th>Application Version</th>
<th>Startup Overhead (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPICH-ABI UDI (GCC Opt)</td>
<td>3.8</td>
</tr>
<tr>
<td>Cray MPI UDI (GCC Opt)</td>
<td>2.4</td>
</tr>
<tr>
<td>Cray Native (PGI Opt static)</td>
<td>2.2</td>
</tr>
<tr>
<td>Cray Native (GCC Opt dynamic)</td>
<td>1.2</td>
</tr>
<tr>
<td>MPICH-ABI UDI (GCC no-Opt)</td>
<td>4.8</td>
</tr>
</tbody>
</table>
IOR Write/Read Performance Comparison

```bash
for i in $(seq 0 $(( NN - 1 ))) ; do
    mkdir -p $dirname/$i ;
    lfs setstripe $dirname/$i/IORfile.$(printf "%08d" $i) -i $((($i%$nost)) -c 1);
Done
aprun -n $NN -N 4 -b $APP -k -v -o $PWD/IORtest/IORfile -w -t 1m -E -b 4g -C -e -k -u -v -F
```
IMB – MPI_AlltoAll Performance Comparison

MPI_Alltoall Average Performance

<table>
<thead>
<tr>
<th>Message Size (bytes)</th>
<th>Cray Native (CCE)</th>
<th>MPICH ABI UDI (GNU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32768</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65536</td>
<td></td>
<td></td>
</tr>
<tr>
<td>131072</td>
<td></td>
<td></td>
</tr>
<tr>
<td>262144</td>
<td></td>
<td></td>
</tr>
<tr>
<td>524288</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1048576</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2097152</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4194304</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Average MB/s

CUG 2016

Copyright 2016 Cray Inc.
## Technique Comparison

<table>
<thead>
<tr>
<th>Consideration</th>
<th>MPICH ABI</th>
<th>Cray MPI</th>
<th>Cray Native</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ease in Building Images</td>
<td>✔</td>
<td>✗</td>
<td>depends</td>
</tr>
<tr>
<td>Allows Use of non-SLES Operating Systems</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Compile Time Dependencies on Cray MPI</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Does not include Cray Intellectual Property in Image</td>
<td>✔</td>
<td>✗</td>
<td>N/A</td>
</tr>
<tr>
<td>Distribution &amp; Runtime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Native HSN Support</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Portability</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>No Runtime Dependency on Local Cray MPI Stack</td>
<td>✗</td>
<td>✔</td>
<td>depends</td>
</tr>
<tr>
<td>Publically Redistributable (No Cray Intellectual property)</td>
<td>✔</td>
<td>✗</td>
<td>N/A</td>
</tr>
<tr>
<td>Cray Performance and General Debugging Tools Support</td>
<td>unknown</td>
<td>unknown</td>
<td>✗</td>
</tr>
</tbody>
</table>
Summary

- The UDI methods explored are straightforward to construct and use while still providing performance that is quite comparable to that of natively compiled applications.
- There are tradeoffs with each UDI method, but each of these methods has a place in bring high performance MPI based containers to the Cray XC environment.
Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, REVEAL, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.
Q&A

Don Bahls
dmb@cray.com