Cray XC50 CLE Port to ARM
CUG 2018
Jeff Schutkoske, Cray Inc.
Agenda

- Purpose
- Functionality
- Lessons Learned
- Performance
- Summary
- Q&A
Purpose

The port of the Cray Linux Environment (CLE) to the XC50 ARM system:

1. Describe the base functionality

2. Share lessons learned

3. Provide initial performance
Functionality
Cray XC50 ARM System Is Here
Cray XC50 ARM System Basics

- Dual socket node
- 32 cores per socket
- 8 DIMMs per socket
- 2.1 GHz base frequency + boost
- Inter-Chip Interconnect 75GB/s/dir
- CLE 6.0 UP07 GA
- SLES 12 SP3
## Cray Linux Environment (CLE)

### Performance and Scalability
- Core Specialization
- Processor Affinity
- Compute Node Cleanup
- Performance Counters
- XPMEM
- Hugepages
- Boot Scalability
- Aries Network

### Services
- System Database
- Boot
- Netroot
- DVS
- DataWarp
- Gateway
- Lustre
- Login

### RAS
- System Dump
- Node Heartbeat
- Node Health Check
- ACPI Platform Error Interface
- ARM CoreSight
- Diagnostics
- Advanced Error Reporting
- Out of Memory Management

### Extensibility Interoperability
- ALPS
- WLM
- MAMU
- MOM

### Power Efficiency
- Power Management
- State Limiting
- Turbo Boost Limiting
- RUR Energy Plug-in

### Cray System Management
- Image Management Provisioning System
Cray Programming Environment (Cray PE)

Programming Languages
- Fortran
- C
- C++
- Chapel
- Python
- R

Programming Models
- Distributed Memory
  - Cray MPI
  - SHMEM
- Shared Memory / GPU
  - OpenMP
  - OpenACC
- Global Arrays
- PGAS & Global View
  - UPC
  - Fortran coarrays
  - Coarray C++
  - Chapel

Programming Environments
- PrgEnv-
  - Cray Compiling Environment PrgEnv-cr
  - GNU PrgEnv-gnu
  - 3rd Party compiler PrgEnv-llvm

Libraries
- Scientific Libraries
  - LAPACK
  - ScaLAPACK
  - BLAS
  - Iterative Refinement Toolkit
  - FFTW
  - NetCDF
  - HDF5

Development Tools
- Environment setup
  - Modules
- Debuggers
  - TotalView
  - DDT
  - gdb4hpc
- I/O Libraries
  - NetCDF
  - HDF5
- Debugging Support
  - Abnormal Termination Processing (ATP)
  - STAT
- Performance Analysis
  - CrayPAT
  - Cray Apprentice²
  - Reveal
  - CCDB

Cray Developed
Cray added value to 3rd party
3rd party packaging
Licensed ISV SW
Lessons Learned
## ARM vs x86 Data Layout - Fortran

<table>
<thead>
<tr>
<th>CCE Fortran</th>
<th>Kind(1)</th>
<th>Kind(2)</th>
<th>Kind(4) default</th>
<th>Kind(8)</th>
<th>Kind(16)</th>
<th>x86 ↔Arm Compatible?</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical</td>
<td>8 bits</td>
<td>16 bits</td>
<td>32 bits</td>
<td>64 bits</td>
<td>-</td>
<td>✓ Yes</td>
</tr>
<tr>
<td>integer</td>
<td>8 bits</td>
<td>16 bits</td>
<td>32 bits</td>
<td>64 bits</td>
<td>-</td>
<td>✓ Yes</td>
</tr>
<tr>
<td>real</td>
<td>-</td>
<td>-</td>
<td>32 bits</td>
<td>64 bits</td>
<td>128 bits</td>
<td>✓ Yes</td>
</tr>
<tr>
<td>complex</td>
<td>-</td>
<td>-</td>
<td>64 bits</td>
<td>128 bits</td>
<td>256 bits</td>
<td>✓ Yes</td>
</tr>
</tbody>
</table>
## ARM vs x86 Data Layout – C and C++

<table>
<thead>
<tr>
<th>C and C++</th>
<th>64-bit x86 ABI</th>
<th>CCE 64-bit x86</th>
<th>64-bit Arm (Cray and ABI)</th>
<th>Compatible?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool</code></td>
<td>8 bits</td>
<td>8 bits</td>
<td>8 bits</td>
<td>✅ Yes</td>
</tr>
<tr>
<td><code>char</code></td>
<td>8 bits</td>
<td>8 bits</td>
<td>8 bits</td>
<td>✅ Partial</td>
</tr>
<tr>
<td><code>short</code></td>
<td>16 bits</td>
<td>16 bits</td>
<td>16 bits</td>
<td>✅ Yes</td>
</tr>
<tr>
<td><code>int</code></td>
<td>32 bits</td>
<td>32 bits</td>
<td>32 bits</td>
<td>✅ Yes</td>
</tr>
<tr>
<td><code>long</code></td>
<td>64 bits</td>
<td>64 bits</td>
<td>64 bits</td>
<td>✅ Yes</td>
</tr>
<tr>
<td><code>long long</code></td>
<td>64 bits</td>
<td>64 bits</td>
<td>64 bits</td>
<td>✅ Yes</td>
</tr>
<tr>
<td><code>float</code></td>
<td>32 bits IEEE</td>
<td>32 bits IEEE</td>
<td>32 bits IEEE</td>
<td>✅ Yes</td>
</tr>
<tr>
<td><code>double</code></td>
<td>64 bits IEEE</td>
<td>64 bits IEEE</td>
<td>64 bits IEEE</td>
<td>✅ Yes</td>
</tr>
<tr>
<td><code>long double</code></td>
<td>80 bits non-IEEE</td>
<td>64 bits IEEE</td>
<td>128 bits IEEE</td>
<td>✗ No</td>
</tr>
</tbody>
</table>
ARM versus x86 – C and C++ char Defaults

● The C and C++ char type has different defaults
  ● x86: char defaults to signed char
  ● ARM: char defaults to unsigned char

● Though the difference seems minor, it has proven to be a significant issue when porting from x86 to ARM
  ● The command line option “-h signedchars” can be used to force ARM to the x86 behavior
ARMv8 Processor Quirks

- ARMv8 processor is less forgiving on alignment even with checking disabled

- Some applications used Intel’s timing instruction directly through an \texttt{ASM()} intrinsic

- ARMv8 is not sequentially (or processor) consistent while x86 is
ARMv8 Port

- Applications written to be portable require very little to no changes
- Recompile and run in many cases
Performance
Cray Compiling Environment (CCE)

- **Compared 158 codes from relevant benchmarks**
  - 76 C codes
  - 11 C++ codes
  - 71 Fortran codes

- **Cray CCE compared against latest LLVM and gcc**
  - Over 67% of the benchmarks executed faster using CCE when compared to LLVM (10% were on par)
  - Over 60% of the benchmarks executed faster when compared to gcc (13% were on par)
Early Results

- Dual socket, 32 cores per socket, 8 DIMMs per socket
- Increased memory bandwidth
- Increased core counts (64 per node)
- High processor frequency (2.1 GHz)
Native ARM Login Nodes

jjs@desktop:~> ssh crayadm@login-aarch
Last login: Mon Apr 23 10:35:28 2018 from 172.30.55.82

   Running 67MB Suse 12.3 image slurm_login-large_17.11.4_cle_6.0up07_sles_12sp3
   CLE release 6.0.UP07, build 6.0.7011(201803112300)
   0 vcores, boot_freemem: 256155mb

crayadm@login-aarch:~> uname -a
Linux login-aarch 4.4.103-6.38_4.0.103-cray_ari_s #1 SMP Mon Mar 5 22:40:16 UTC 2018 (4.0.103) aarch64
   aarch64 aarch64

   crayadm@login-aarch:~> cat /proc/cpuinfo | tail -n 10

processor       : 255
BogoMIPS        : 400.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics
CPU implemender : 0x43
CPU architecture: 8
CPU variant     : 0x1
CPU part        : 0x0af
CPU revision    : 0
Hello World SHMEM and UPC

```bash
shmem@login-aarch:~> cc shmem-hello.c -o shmem-hello
shmem@login-aarch:~> srun -N 4 ./shmem-hello
Hello from thread 1 on CPU 0 on host nid00039
Hello from thread 2 on CPU 0 on host nid00040
Hello from thread 3 on CPU 0 on host nid00041
Hello from thread 0 on CPU 0 on host nid00038
```

```bash
shmem@login-aarch:~> cc -h upc hello.upc -o hello-upc
shmem@login-aarch:~> srun -N 4 ./hello-upc
Hello from thread 1 on host nid00039
Hello from thread 2 on host nid00040
Hello from thread 3 on host nid00041
Hello from thread 0 on host nid00038
```
**Netstress SHMEM**

```bash
crayadm@login-aarch:~$ make shmem
cc -DCRAY -O3 -D_SHMEM -DNETSTRESS_VERSION="2.4.0" -o ./netstress-shmem src/*.c -lm
src/netstress_utils.c:
src/rand64.c:
src/timer.c:

crayadm@login-aarch:~$ export XT_SYMMETRIC_HEAP_SIZE=1024M

crayadm@login-aarch:~$ srun -N 4 ./netstress-shmem -m 8K -M 64K
<snipped table for brevity>
```

<table>
<thead>
<tr>
<th>Group</th>
<th>Size</th>
<th>Max Time (sec)</th>
<th>Min B/W (MB/s)</th>
<th>Avg B/W (MB/s)</th>
<th>Max B/W (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>all:</td>
<td>8 KB</td>
<td>0.0027</td>
<td>690.09</td>
<td>693.73</td>
<td>696.51</td>
</tr>
<tr>
<td>all:</td>
<td>16 KB</td>
<td>0.0013</td>
<td>2811.20</td>
<td>2843.08</td>
<td>2895.55</td>
</tr>
<tr>
<td>all:</td>
<td>32 KB</td>
<td>0.0023</td>
<td>3224.40</td>
<td>3305.90</td>
<td>3446.62</td>
</tr>
<tr>
<td>all:</td>
<td>64 KB</td>
<td>0.0044</td>
<td>3399.13</td>
<td>3499.82</td>
<td>3648.70</td>
</tr>
</tbody>
</table>

# Total run time: 0.01s
Netstress UPC

crayadm@login-aarch:~/netstress-2.4.0> make upc
cc -D_UPC -DNETSTRESS_VERSION="2.4.0" -o ./netstress-upc src/*.c -h upc -h omp -DCRAY -O3 -lm
src/netstress.c:
    src/netstress_utils.c:
    src/rand64.c:
    src/timer.c:

crayadm@login-aarch:~/netstress-2.4.0> export XT_SYMMETRIC_HEAP_SIZE=1024M

crayadm@login-aarch:~/netstress-2.4.0> srun -N 4 ./netstress-upc -m 8K -M 64K

<snipped table for brevity>

<table>
<thead>
<tr>
<th>Group</th>
<th>Size</th>
<th>Max Time</th>
<th>Min B/W</th>
<th>Avg B/W</th>
<th>Max B/W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(sec)</td>
<td>(MB/s)</td>
<td>(MB/s)</td>
<td>(MB/s)</td>
</tr>
<tr>
<td>all:</td>
<td>8 KB</td>
<td>0.0009</td>
<td>1998.56</td>
<td>2016.62</td>
<td>2029.50</td>
</tr>
<tr>
<td>all:</td>
<td>16 KB</td>
<td>0.0014</td>
<td>2763.29</td>
<td>2769.00</td>
<td>2785.80</td>
</tr>
<tr>
<td>all:</td>
<td>32 KB</td>
<td>0.0024</td>
<td>3187.48</td>
<td>3228.21</td>
<td>3272.03</td>
</tr>
<tr>
<td>all:</td>
<td>64 KB</td>
<td>0.0043</td>
<td>3466.56</td>
<td>3524.43</td>
<td>3611.01</td>
</tr>
</tbody>
</table>
# Total run time: 0.02s
Summary

● Cray XC50 systems with ARMv8 available
  ● Dual socket, 32 cores per socket, 8 DIMMs per socket

● CLE available
  ● Equivalent system functionality, performance, and scalability

● Cray PE available
  ● Compilers, programming environments, and developer tools
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 publicly 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, URIKA and YARCDATA. The following are trademarks of Cray Inc.: CHAPEL, CLUSTER CONNECT, CLUSTERSTOR, CRAYDOC, CRAYPAT, CRAYPORT, DATAWARP, ECOPHLEX, LIBSCI, NODEKARE, REVEAL. 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 on this website are the property of their respective owners.