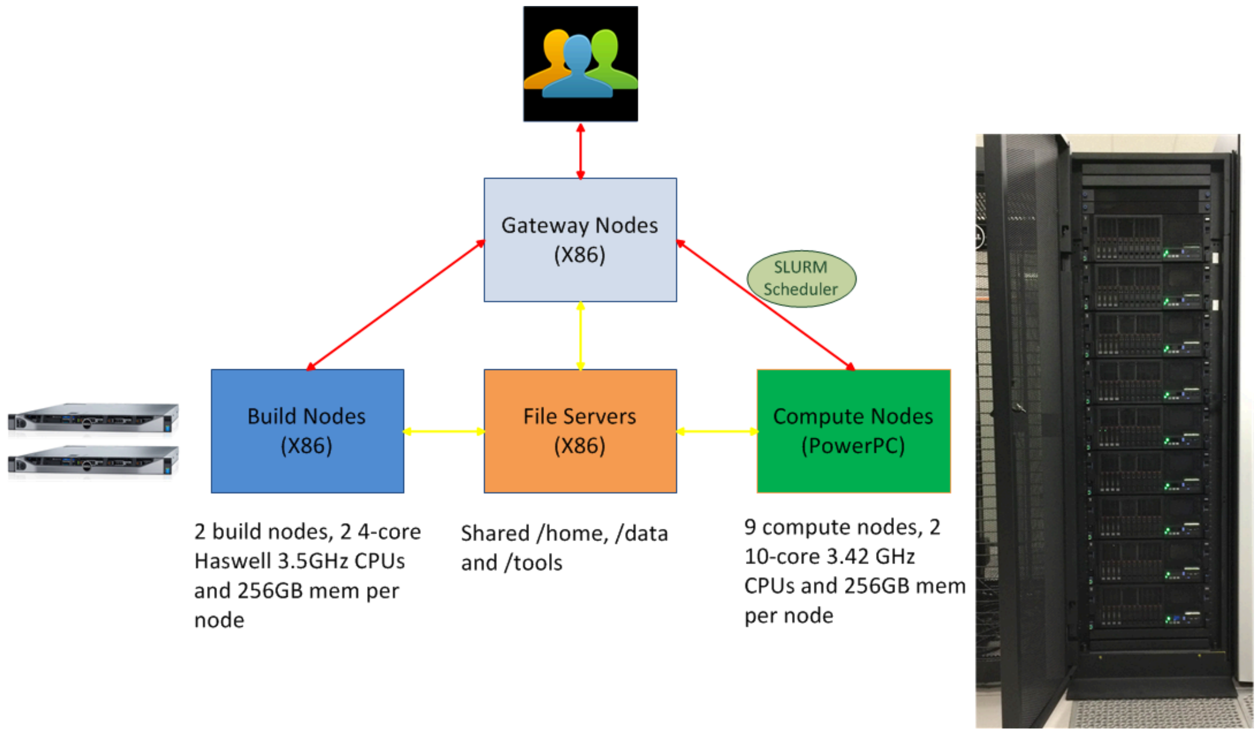
TACC POWER8 Cluster Quick Start Guide (2015-12)

1. System Overview

The TACC POWER8 Cluster is a cluster of several x86 servers and nine POWER8 servers. Each POWER8 node is a heterogeneous platform capable of running GPGPU- and/or FPGA-accelerated applications.



Gateway Nodes (X86)
SLURM Scheduler
Build Nodes (X86)
File Servers (X86)
Compute Nodes (PowerPC)
2 build nodes, 2 4-core Haswell 3.5GHz CPUs and 256GB mem per node
Shared /home, /data and /tools
9 compute nodes, 2 10-core 3.42 GHz CPUs and 256GB mem per node

2. User Access

Visit TACC allocation webpage (https://portal.tacc.utexas.edu/allocations-overview) for allocation information.

If you would like to use the POWER8 system, obtain a TACC account and then request project allocations using the TACC Allocations Request System available through the TACC User Portal:https://portal.tacc.utexas.edu/.

An alternative way of getting an account on the POWER8 system is through OpenFAbRIC by following instructions on www.openfabric.org.

Once your allocation is approved, you will be able to ssh login to the login node (***login1.fabric.tacc.utexas.edu***). Access to the P8 nodes and build nodes must go through the login node for security reasons.

You can request one POWER8 node through the ***SLURM*** scheduler by running ***idev*** command on the login node. You can request one node at a time. The request will be approved or rejected automatically based on the nodes' availability and node allocation policy. If the request is approved, one of the nine POWER8 nodes (***p8capi1-p8capi9***) will be assigned and you will be able to ssh log into the assigned node from the login node.

You can also ssh into one of the two x86 build nodes (***build1/build2***). These nodes are Dell PowerEdge 630 servers optimized to deliver high single-thread performance for running workloads like FPGA synthesis.

3. File System

There are three filesystems, ***/home***, ***/data and /tools*** that are mounted on all x86 and POWER8 nodes. ***/home*** is backed up however ***/data*** is not. Two environment variables, ***$HOME*** and ***$DATA***, are currently set for your convenience. The command ***cdd*** will cd to your ***/data*** directory. The initial default user quotas are 1GB for ***/home*** directory and 100GB for a user's ***/data*** directory. ***/tools*** is read-only.

4. Accelerators

Our current setup supports three accelerating devices.

- Nallatech 385 A7 Stratix V Altera-based FPGA adapter
- Alpha-data 7V3 Virtex7 Xilinx-based FPGA adapter
- NVIDIA Tesla K40m GPGPU card

Currently each node has one node from each vendor. FPGA boards are CAPI-enabled. CAPI (Coherent Accelerator Processor Interface) on POWER8 systems provides a high-performance solution for implementation of coherent shared memory between the processor and accelerators.

4.1 using FPGAs

CAPI enables a customer-defined FPGA solution to be a peer to the POWER8 cores from a memory access (coherent) standpoint.

a) How to access CAPI FPGAs?

Reconfiguring CAPI FPGAs can be done by running scripts installed under ***/tools/ppc_64/capi_flash*** directory.

- ***capi-ls-boards.sh*** displays the active boards connected to the node.
- ***capi-flash-fpga.sh*** writes the image to the FPGA board.

Make sure you are in the ***cxl*** group in order to run these commands as root.

Regardless the vendor (either Altera or Xilinx) of the FPGA to be programmed, the input to the CAPI programming script (***capi-flash-fpga.sh***) should be a ***dat*** file, which needs to be generated by running ***/tools/x86_64/capi_flash/x86_files/create_flash_file.sh*** on one x86 build (build1/build2) node.

The ***/tools/x86_64/capi_flash/x86_files/create_flash_file.sh*** takes an argument which is the FPGA image file generated by the FPGA synthesis tool. For Altera flow, the image is a ***sof*** file. For Xilinx flow, the image is a ***bit*** file.

Running host programs usually requires the ***libcxl*** library installed under ***/tools/ppc_64/libcxl***. You may need to set your ***LD_LIBRARY_PATH*** and ***C_INCLUDE_PATH*** (or ***CPLUS_INCLUDE_PATH*** for C++ headers).

b) How to build CAPI FPGA images?

You'll need to build your FPGA image on an X86 machine with a valid Altera/Xilinx license.

The Altera build flow is available on x86 nodes. You can use it on the login node (***login1.fabric.tacc.utexas.edu***) or one build nodes (***build1/build2***).
Altera Quartus 15.0 has been installed under ***/tools/x86_64/altera/15.0/quartus***.
Point your ***LM_LICENSE_FILE*** to ***27000@license02.tacc.utexas.edu*** for license setup.
A build example, including the project settings, timing constraints, pin assignment and the encrypted PSL and the verilog/VHDL wrapper of AFU will be available soon.
You can reuse most things of the example and may need to change the AFU only.

Currently we don't supply a valid license for building CAPI images for the Xilinx flow.
You will need to build Xilinx image on your own machine for now.
We are working on the license issue with IBM and Xilinx.

c) CAPI with OpenCL

OpenCL allows the use of a C-based programming language for developing code across different platforms such as central processing units (CPUs), graphic processing units (GPUs), digital signal processors (DSPs), and field-programmable gate arrays (FPGAs). Altera OpenCL (AOCL) High-level Synthesis flow allows users to abstract away the traditional hardware FPGA development flow for a higher level software development flow. Our current settings support AOCL for the Nallatech 385 A7 board. The board support package has been installed under ***/tools/x86_64/altera/aocl-rte-ppc64le+capi***.

On the build machine, you will need to set several environment variables: ***ALTERAOCLSDKROOT***, ***LD_LIBRARY_PATH***, ***AOCL_BOARD_PACKAGE_ROOT*** and ***LM_LIBRARY_FILE***. Below is a setup example for ***BASH***.

```
# in .bashrc file
export ALTERAOCLSDKROOT="/tools/x86_64/altera/15.0/hld"
export LD_LIBRARY_PATH=$ALTERAOCLSDKROOT/host/linux64/lib:$LD_LIBRARY_PATH
export AOCL_BOARD_PACKAGE_ROOT=/tools/x86_64/altera/aocl-rte-ppc64le+capi/board/capi_bsp_v2
export LD_LIBRARY_PATH=$AOCL_BOARD_PACKAGE_ROOT/linux64/lib:$LD_LIBRARY_PATH
export
LM_LICENSE_FILE=/tools/x86_64/capi_development_kit/psl_files/PSL_Altera/psl_A000_license.dat:$LM
_LICENSE_FILE
```

Compile with the ***--board 385_a7_capi***. For example,

```
aoc device/memcopy.cl -o bin/memcopy.aocx --board 385_a7_capi -v
```

On POWER8 machines, source ***/tools/ppc_64/altera/s4CapiAocl_v2.sh*** before compiling and running the host executable. In addition, export the ***AOC_TARGET_AFU*** variable to the AFU device directory. For example,

```
export AOC_TARGET_AFU=/dev/cxl/afu1.0d (if the Nallatech 385 card is listed as card1)
```

Running OpenCL on Xilinx FPGAs is not supported yet.

d) CAPI with Bluespec

BlueLink (https://github.com/jeffreycassidy/BlueLink) is Bluespec SystemVerilog library for CAPI. The current version provides a thin wrapper over the Verilog interface provided by IBM, along with type definitions to make interfacing easier.

4.2 using GPGPUs

NVIDIA CUDA 7.5 has been installed under ***/usr/local/cuda-7.5***.

5. Help Resources

An CAPI example in Github: (https://github.com/ibm-capi/pslse)
IBM CAPI forum:
https://www.ibm.com/developerworks/community/groups/community/CAPI_Developers_Community

An open-source PSL streaming framework:
http://slides.com/mbrobbel/capi-streaming-framework
https://github.com/mbrobbel/capi-streaming-framework

An open-source textswap example: https://github.com/sbates130272/capi-textswap

A CAPI developer's blog:
http://suchprogramming.com/tinkering-with-capi/
http://suchprogramming.com/hello-afu-part-1/

Nvidia GPU help resources: https://devtalk.nvidia.com/

For other questions, please submit a ticket to TACC.