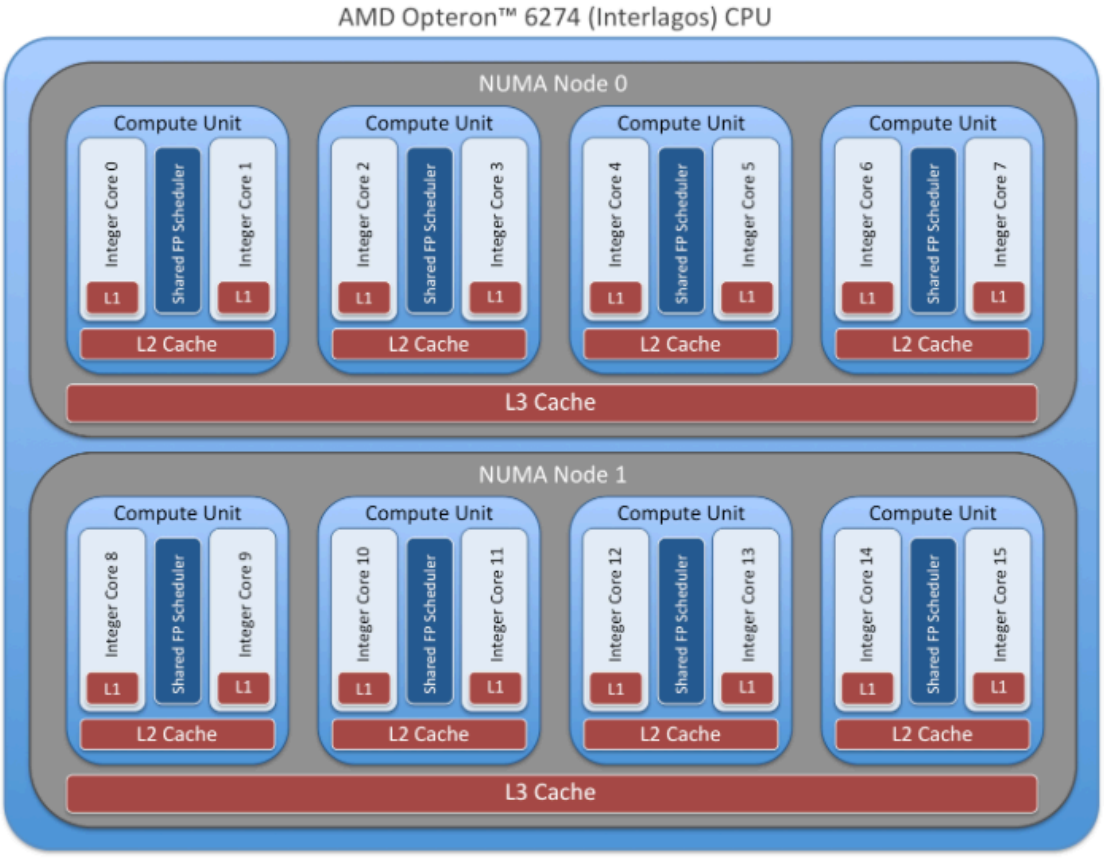
March User Call Tutorial: Titan's Turbo Switch



The aprun module is a new tool at OLCF to help you identify when you have chosen a task layout that has potential for unnecessary floating-point contention on the CPU.

Titan has 16 compute units per node. The compute units are paired so that two compute units share a floating-point operation scheduler (fpu). By default aprun will place 16 process per node, disturbing them sequentially so each fpu has two tasks. For applications that are not heavily fpu dependent this is not an issue, however fpu dependent apps may be able to run in close to half the walltime by requesting twice the nodes and distributing one task per fpu.

Users often chose to run with 8 tasks per node rather than 16 to maximize the amount of memory per task. **Even if you only request 8 processes per node, they will be placed on cores 0-7 and compete for the first four FPUs**

This is where the aprun tool can help by warning you if you have not picked aprun options that distribute the tasks. Here are two exercise that 1. Show you the apun module's warning messages and 2. Allows you to see how fpu contention impacts the performance of the High Performance Computing Linpack Benchmark.

Exercise 1. Trigger the fpu contention warning.
The aprun module is loaded by default on Titan, so user will see this warning when they submit a job with potential fpu contention.

1. Login to Titan
2. Change to you Lustre workspace: cd $MEMBERWORK/projID
3. Request an interactive job for 16 nodes and run with 128 tasks with 8 tasks per node. **(-N 8 or -S 4)**

```
qsub -I -Astf007 -lnodes=16,walltime=00:30:00
aprun -n 128 -N 8 hostname
```

You should see the warning and a suggestion for where you can find more information about task layout on the OLCF Titan User guide.

To distribute the tasks use the –j1 option. On Titan, the -j parameter specifies the number of CPUs to be allocated per *paired-core compute unit*. The valid values for -j are 0 (use the system default), 1 (use one integer core), and 2 (use both integer cores; this is the system default).

Place 8 processes per node (-N 8 or -S 4 –j1)

```
qsub -I -Astf007 -lnodes=16,walltime=00:30:00
aprun -n 128 -N 8 –j1 hostname
```

You should not see the warning.

If you want to learn more about task distribution see:
https://www.olcf.ornl.gov/kb_articles/controlling-mpi-task-layout-within-a-physical-node/

Exercise 2: Speed up a floating-point intensive code by changing aprun options.

This example uses the High Performance Computing Linpack Benchmark (HPL) a code that solves a random dense linear system in double precision arithmetic. The job submission script appears in this exercise and is also available on the User Call Event site.

1. Copy it to your Lustre scratch space. $MEMBERWORK/projId, where projId is your project's 6 character identifier.

2. Open the script and edit the PROJ variable on lines 2 and 6 to be your project ID. Example: PROJ=stf00

The script runs a 2 node job three ways:

1. with 16 tasks per node: aprun -n 16 -N 16
2. with 8 tasks per node: aprun -n 16 -N 8
3. with 8 tasks per node one rank per fpu: aprun -n 16 -N 8 -j 1

cat hpl.qsub

```
#!/bin/bash
#PBS -A proJID
#PBS -l nodes=2,walltime=1:00:00
#PBS -j oe
#PBS -N Aprun-Usage
PROJ=projID
DIR=/lustre/atlas/world-shared/stf007/User_call/hpl/XE/test/
cd $MEMBERWORK/$PROJ
ln -s $DIR/HPL.dat .

# the default most people run with
# 16 processes will contend over 8 floating point units

/usr/bin/time -p aprun -n 16 -N 16 $DIR/xhpl 2>&1 > N16.out

# Some apps need more memory, so users only request 8 ranks per node
# However, the 8 processes will contend over 4 floating point units

/usr/bin/time -p aprun -n 16 -N 8  $DIR/xhpl 2>&1 > N8.out

# Using -j 1 to place one rank per floating point unit avoids contention

/usr/bin/time -p aprun -n 16 -N 8 -j 1 $DIR/xhpl 2>&1 > N8-j1.out
```

The executable and data are in the User Assistance $WORLDWORK/stf007 directory. You do not need to copy them because all OLCF users are able to read and execute files sitting in $WORLDWORK/projid. (Your project has a $WORLDWORK too!)

Run the script
qsub hpl.qsub

There are four output files from this exercise: N16.out, N8.out and N8-j1.out – which contain the output from the hpl benchmark- and Aprun-Usage.o_job_nubmer - which has the timing. Open Aprun-Usage.o and look at the timing. They will be list in this order:

1. fully packed node with 16 tasks per node: -n 16 -N 16
2. Half packed node with tasks sharing fpus: -n 16 -N 8
3. Half packed node with 1 task per fpu: -n 16 -N 8 -j 1

Notice the similarity in the walltimes between 1 and 2 and the boost that –j1 gave 3 compared to 2.

See the slides at https://www.olcf.ornl.gov/training-event/olcf-user-conference-call-aprun-usage-module-design/

Or come to the conference call for a discussion of these results.