Using the Launcher

Lucas A. Wilson
Overview

• What is the Launcher?
• Important Launcher Files
  – Batch submission script
  – paramlist file
• Environment Variables
• Task Layout
• New Features in Launcher 2.0
  – Task Scheduling
  – Intel Xeon Phi support
• Be sure to check the Readme file for more information
What is the Launcher?

• The Launcher is a set of shell scripts which allow for the bundling of many independent serial/threaded execution runs into a single parallel job.
  – More efficient use of TACC resources
  – More efficient use of Service Units (SUs)
Important Launcher Files

• The Launcher consists of several files:
  – `launcher.slurm` (Stampede) – A SLURM batch submission script to request resources from the system
  – `paramlist` – A plain-text file containing the various tasks to execute (1 per line)
  – Behind-the-scenes files:
    • `paramrun`, `launcher`, `init_launcher`
How Launcher Works

Node 0
- `init_launcher`
  - `launcher`
  - `launcher`

Node 1
- `init_launcher`
  - `launcher`
  - `launcher`

Node NHOSTS-1
- `init_launcher`
  - `launcher`
  - `launcher`

`paramrun`

`paramlist`
Paramlist structure

```
program [args...] [<infile] [>outfile]
program [args...] [<infile] [>outfile]
program [args...] [<infile] [>outfile]
program [args...] [<infile] [>outfile]
program [args...] [<infile] [>outfile]
program [args...] [<infile] [>outfile]
program [args...] [<infile] [>outfile]
program [args...] [<infile] [>outfile]
```
Example paramlist file

echo $TACC_LAUNCHER_JID
echo $TACC_LAUNCHER_TSK_ID
env | grep LAUNCHER
touch foo
rm somefile
cat anotherfile > yetanotherfile
Environment Variables

• TACC_LAUNCHER_NHOSTS
  – The number of hosts for this job
• TACC_LAUNCHER_PPN
  – The number of tasks per node for this job
• TACC_LAUNCHER_JID
  – The id of the current job (line in the paramlist)
• TACC_LAUNCHER_TSK_ID
  – The id of the current task
Task Layout

• Sometimes your programs need more than 2GB of memory
• You can adjust the number of nodes and the tasks per node through your launcher.slurm script
Task Layout

#---------------------Scheduler Options---------------------

#SBATCH -J Parametric     # Job name
#SBATCH -N 1              # Total number of nodes (16 cores/node)
#SBATCH -n 16             # Total number of tasks
#SBATCH -p normal-mic     # Queue name
#SBATCH -o Parametric.o%j # Name of stdout output file (%j expands to jobid)
#SBATCH -t 01:00:00       # Run time (hh:mm:ss)
#
#<---------------- Account String ------------------>
#<-- (Use this ONLY if you have MULTIPLE accounts) --->
##SBATCH -A
Task Layout

• If you need to have all of the memory on a normal node (~30.9 GB), make \(-n\) and \(-N\) the same value.

• If you need something less, you can adjust the number of nodes (N) and the number of tasks (n) to fit your particular needs

• The number of tasks / node is \(n/N\)
New Features in Launcher 2.0
Task Scheduling

• There are three different scheduling methods:
  – Static
    • Interleaved
    • Block
  – Dynamic

• Use the environment variable TACC_LAUNCHER_SCHED to change scheduling method:
  – interleaved (default)
  – block
  – dynamic
Interleaved Scheduling

- 4 tasks
- 8 jobs (paramfile lines)

job1 task1
job2 task2
job3 task3
job4 task4
job5 task1
job6 task2
job7 task3
job8 task4
Block Scheduling

- 4 tasks
- 8 jobs (paramfile lines)

<table>
<thead>
<tr>
<th>job1</th>
<th>task1</th>
</tr>
</thead>
<tbody>
<tr>
<td>job2</td>
<td>task1</td>
</tr>
<tr>
<td>job3</td>
<td>task2</td>
</tr>
<tr>
<td>job4</td>
<td>task2</td>
</tr>
<tr>
<td>job5</td>
<td>task3</td>
</tr>
<tr>
<td>job6</td>
<td>task3</td>
</tr>
<tr>
<td>job7</td>
<td>task4</td>
</tr>
<tr>
<td>job8</td>
<td>task4</td>
</tr>
</tbody>
</table>
Dynamic Scheduling

- 4 tasks
- 8 jobs (paramfile lines)
- We don’t know ahead of time how the last jobs will be scheduled
  - Depends on the runtime of the first jobs
Dynamic Scheduling Performance

![Graph showing the relationship between number of cores and time in minutes. The graph compares actual time (blue line) and ideal time (red dashed line). As the number of cores increases from 4,096 to 65,536, the time decreases, approaching the ideal time.]
Xeon Phi support

- Launcher 2.0 will have support for execution of tasks on both the host processors and on the Xeon Phi coprocessors
  - Uses a different set of tasks for the Phi (since ISA, memory is different)
  - TACC_LAUNCHER_NPHI, TACC_LAUNCHER_PHI_PPN
  - phiparamfile
Questions?