Agenda

● Getting Optimal Throughput on Cori and Edison
● Variable-Time Jobs
● Upcoming Training Opportunities
● NERSC podcast
Getting Optimal Throughput on Edison & Cori

NERSC
Queue Backlog Past 30 Days

Cori Haswell

Cori KNL

Edison

Scale: 20 days

Scale: 4 days

Scale: 10 days
Strategies

- **Use Cori KNL! (shortest backlog)**
  - What if my code won’t run on KNL?
    - Ask NERSC for help! We can help you figure out what improvements need to be made in your code, & provide strategies for exploiting KNL architecture

- **Ask for less time (and more processors)**
  - What if the scalability of my code is not good?
    - If you’re the developer, ask NERSC for help! We can help you figure out how to make your code better!
    - Implement checkpointing and use variable-time job options for better throughput (see next section)
Variable-Time Jobs

Acknowledgement:

The work was done by Tiffany Connors, a summer student at NERSC, under the supervision of Rebecca Hartman-Baker
Who should use variable-time jobs?

- Users who need to run long jobs, including jobs running for more than 48 hours - the max time allowed on Cori and Edison.
- Users who want improved queue turnaround time.

- Provided the code can do checkpointing by itself
Variable-Time jobs

• Slurm allows jobs submitted with a minimum time limit in addition to the (max) time limit, e.g.,
  #SBATCH --time=48:00:00
  #SBATCH --time-min=2:00:00

• Jobs specifying --time-min can start execution earlier than otherwise with a time limit anywhere between the time-min and the requested time limit.
  – This is performed by a backfill scheduling algorithm to allocate resources otherwise reserved for higher priority jobs.
Variable-Time jobs - continued

- The pre-terminated job can be requeued to resume from where the previous execution left off.
  - \#SBATCH --requeue
- Requeuing the pre-terminated job can be done automatically until the cumulative execution time reaches the requested time limit or the job completes earlier before the requested time limit.
Sample variable-time job script

#!/bin/bash
#SBATCH --J test
#SBATCH --q regular
#SBATCH --C haswell
#SBATCH --N 1
#SBATCH --comment=64:00:00
#SBATCH --time-min=00:30:00  # the minimum amount of time the job should run
#SBATCH --time=48:00:00
#SBATCH --error=test-%j.err
#SBATCH --output=test-%j.out
#
#SBATCH --signal=B:USR1@120
#SBATCH --requeue
#SBATCH --open-mode=append

# timelimit per job, the amount of time (in seconds) needed for checkpointing (same as in --signal)
# and the checkpoint command if any
max_timelimit=48:00:00  # if not set default to 48:00:00
ckpt_overhead=120       # if not set, default to 60 seconds
ckpt_command=

# requeue the job if remaining time > 0
module load ata
. $ATA_DIR/etc/ATA_setup.sh

requeue_job func_trap USR1
#

# user setting
export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=1

# srun must execute in background and catch signal on wait command
srun -n 1 -c 64 --cpu_bind=cores ./a.out &
wait
The variable-time job script generator - adaptive_launch.sh
How effective are variable-time jobs?

<table>
<thead>
<tr>
<th>JOBID</th>
<th>ST</th>
<th>USER</th>
<th>NAME</th>
<th>NODES</th>
<th>REQUESTED</th>
<th>USED</th>
<th>SUBMIT</th>
<th>QOS</th>
<th>SCHEDULED_START</th>
<th>FEATURES</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>12537227</td>
<td>R</td>
<td>zz217</td>
<td>ata</td>
<td>1</td>
<td>4:35:00</td>
<td>3:30:13</td>
<td>2018-05-17T04:23:02</td>
<td>regular_1</td>
<td>2018-05-17T04:34:12</td>
<td>haswell</td>
<td>None</td>
</tr>
<tr>
<td>12537232</td>
<td>R</td>
<td>zz217</td>
<td>ata</td>
<td>1</td>
<td>3:55:00</td>
<td>3:30:13</td>
<td>2018-05-17T04:23:02</td>
<td>regular_1</td>
<td>2018-05-17T04:34:12</td>
<td>haswell</td>
<td>None</td>
</tr>
<tr>
<td>12537235</td>
<td>R</td>
<td>zz217</td>
<td>ata</td>
<td>1</td>
<td>3:55:00</td>
<td>3:30:13</td>
<td>2018-05-17T04:23:02</td>
<td>regular_1</td>
<td>2018-05-17T04:34:12</td>
<td>haswell</td>
<td>None</td>
</tr>
<tr>
<td>12537216</td>
<td>R</td>
<td>zz217</td>
<td>ata</td>
<td>1</td>
<td>4:49:00</td>
<td>3:34:01</td>
<td>2018-05-17T04:23:00</td>
<td>regular_1</td>
<td>2018-05-17T04:30:24</td>
<td>haswell</td>
<td>None</td>
</tr>
<tr>
<td>12537219</td>
<td>R</td>
<td>zz217</td>
<td>ata</td>
<td>1</td>
<td>4:49:00</td>
<td>3:34:01</td>
<td>2018-05-17T04:23:00</td>
<td>regular_1</td>
<td>2018-05-17T04:30:24</td>
<td>haswell</td>
<td>None</td>
</tr>
<tr>
<td>12537221</td>
<td>R</td>
<td>zz217</td>
<td>ata</td>
<td>1</td>
<td>4:49:00</td>
<td>3:34:01</td>
<td>2018-05-17T04:23:00</td>
<td>regular_1</td>
<td>2018-05-17T04:30:24</td>
<td>haswell</td>
<td>None</td>
</tr>
<tr>
<td>12537224</td>
<td>R</td>
<td>zz217</td>
<td>ata</td>
<td>1</td>
<td>4:49:00</td>
<td>3:34:01</td>
<td>2018-05-17T04:23:00</td>
<td>regular_1</td>
<td>2018-05-17T04:30:24</td>
<td>haswell</td>
<td>None</td>
</tr>
<tr>
<td>12537214</td>
<td>R</td>
<td>zz217</td>
<td>ata</td>
<td>1</td>
<td>4:35:00</td>
<td>3:30:09</td>
<td>2018-05-17T04:28:06</td>
<td>regular_1</td>
<td>2018-05-17T04:34:16</td>
<td>haswell</td>
<td>None</td>
</tr>
<tr>
<td>12537212</td>
<td>R</td>
<td>zz217</td>
<td>ata</td>
<td>1</td>
<td>4:36:00</td>
<td>1:40:55</td>
<td>2018-05-17T06:18:25</td>
<td>regular_1</td>
<td>2018-05-17T06:23:30</td>
<td>haswell</td>
<td>None</td>
</tr>
<tr>
<td>12537192</td>
<td>PD</td>
<td>zz217</td>
<td>test</td>
<td>1</td>
<td>48:00:00</td>
<td>0:00</td>
<td>2018-05-17T03:27:11</td>
<td>regular_1</td>
<td>avail_in~2.8_days</td>
<td>haswell</td>
<td>Priority</td>
</tr>
<tr>
<td>12537194</td>
<td>PD</td>
<td>zz217</td>
<td>test</td>
<td>1</td>
<td>48:00:00</td>
<td>0:00</td>
<td>2018-05-17T03:27:12</td>
<td>regular_1</td>
<td>avail_in~2.8_days</td>
<td>haswell</td>
<td>Priority</td>
</tr>
<tr>
<td>12537196</td>
<td>PD</td>
<td>zz217</td>
<td>test</td>
<td>1</td>
<td>48:00:00</td>
<td>0:00</td>
<td>2018-05-17T03:27:14</td>
<td>regular_1</td>
<td>avail_in~2.8_days</td>
<td>haswell</td>
<td>Priority</td>
</tr>
<tr>
<td>12537198</td>
<td>PD</td>
<td>zz217</td>
<td>test</td>
<td>1</td>
<td>48:00:00</td>
<td>0:00</td>
<td>2018-05-17T03:27:15</td>
<td>regular_1</td>
<td>avail_in~2.8_days</td>
<td>haswell</td>
<td>Priority</td>
</tr>
<tr>
<td>12537200</td>
<td>PD</td>
<td>zz217</td>
<td>test</td>
<td>1</td>
<td>48:00:00</td>
<td>0:00</td>
<td>2018-05-17T03:27:16</td>
<td>regular_1</td>
<td>avail_in~2.8_days</td>
<td>haswell</td>
<td>Priority</td>
</tr>
<tr>
<td>12537202</td>
<td>PD</td>
<td>zz217</td>
<td>test</td>
<td>1</td>
<td>48:00:00</td>
<td>0:00</td>
<td>2018-05-17T03:27:18</td>
<td>regular_1</td>
<td>avail_in~2.8_days</td>
<td>haswell</td>
<td>Priority</td>
</tr>
<tr>
<td>12537204</td>
<td>PD</td>
<td>zz217</td>
<td>test</td>
<td>1</td>
<td>48:00:00</td>
<td>0:00</td>
<td>2018-05-17T03:27:20</td>
<td>regular_1</td>
<td>avail_in~2.8_days</td>
<td>haswell</td>
<td>Priority</td>
</tr>
<tr>
<td>12537206</td>
<td>PD</td>
<td>zz217</td>
<td>test</td>
<td>1</td>
<td>48:00:00</td>
<td>0:00</td>
<td>2018-05-17T03:27:21</td>
<td>regular_1</td>
<td>avail_in~2.8_days</td>
<td>haswell</td>
<td>Priority</td>
</tr>
<tr>
<td>12537208</td>
<td>PD</td>
<td>zz217</td>
<td>test</td>
<td>1</td>
<td>48:00:00</td>
<td>0:00</td>
<td>2018-05-17T03:27:22</td>
<td>regular_1</td>
<td>avail_in~2.8_days</td>
<td>haswell</td>
<td>Priority</td>
</tr>
<tr>
<td>12537210</td>
<td>PD</td>
<td>zz217</td>
<td>test</td>
<td>1</td>
<td>48:00:00</td>
<td>0:00</td>
<td>2018-05-17T03:27:23</td>
<td>regular_1</td>
<td>avail_in~2.8_days</td>
<td>haswell</td>
<td>Priority</td>
</tr>
</tbody>
</table>

Note: The SCHEDULED_START column shows either the estimated start time provided by Slurm, or that an estimated start time will be "available in xxx hrs/days", meaning your job won't be scheduled by then (due to priority) unless via backfill, and the actual job start time could still be some period of time after then.

zz217@cori09:/global/cscratch1/sd/zz217/tests/queue_turnaround/haswell> squeue -u zz217 -o "%A %j %k" | grep -v null
<table>
<thead>
<tr>
<th>JOBID</th>
<th>NAME</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>12537227</td>
<td>ata</td>
<td>42:57:00</td>
</tr>
<tr>
<td>12537232</td>
<td>ata</td>
<td>43:37:00</td>
</tr>
<tr>
<td>12537235</td>
<td>ata</td>
<td>43:37:00</td>
</tr>
<tr>
<td>12537216</td>
<td>ata</td>
<td>42:43:00</td>
</tr>
<tr>
<td>12537219</td>
<td>ata</td>
<td>42:43:00</td>
</tr>
<tr>
<td>12537221</td>
<td>ata</td>
<td>42:43:00</td>
</tr>
<tr>
<td>12537224</td>
<td>ata</td>
<td>42:43:00</td>
</tr>
<tr>
<td>12537214</td>
<td>ata</td>
<td>42:34:00</td>
</tr>
<tr>
<td>12537212</td>
<td>ata</td>
<td>40:48:00</td>
</tr>
</tbody>
</table>
Notes about variable jobs

• If you run into any issues please contact consult@nersc.gov for help. This feature is still experimental.

• For those applications that do not do checkpointing by themselves, DMTCP can be used to checkpointing externally. We are testing DMTCP + Variable-Time jobs now, and will make the job script available to users soon.
Questions?
Upcoming Training Opportunities
Upcoming Training Opportunities

● **NERSC Training**
  ○ Cray PE Workshop, June 14

● **ECP Training**
  ○ Best Practices for HPC Software Developers, June 13
  ○ Kokkos Bootcamp, July 24-27
Cray PE Training, June 14

- **Topics covered in training:**
  - Applying the “Whack-A-Mole” Method Using Cray’s Perftools to Identify the Moles
    - Tutorial, originally presented at Cray User Group meeting
    - Learn to use Cray’s Perftools (performance tools) to identify bottlenecks and hotspots in your code
  - What’s new in the Cray Programming environment?
  - Tips & Tricks for using & interpreting data from Perftools
  - Cray PE Deep Learning Scalability Plugin
    - Introduction to Deep Learning products available in the Cray programming environment

- **For more info and to register:**
  [https://www.nersc.gov/users/training/events/cray-pe-workshop-june2018/](https://www.nersc.gov/users/training/events/cray-pe-workshop-june2018/)
Best Practices for HPC SW Developers Webinar, June 13

- **Topic:** “Popper: Creating Reproducible Computational and Data Science Experimentation Pipelines”
  - Presented by Ivo Jimenez, UC Santa Cruz
  - Trying to reproduce others’ computational experiments is a large time sink. Popper, a protocol and command language interpreter (CLI) tool for implementing scientific exploration pipelines, follows a DevOps approach of unifying software development and operation in order to handle complexity in large codebases.

- **For more info and to register:**
  [https://www.exascaleproject.org/event/popper/](https://www.exascaleproject.org/event/popper/)
Kokkos Bootcamp, July 24-27

● **What is Kokkos?**
  ○ Programming model and library for writing performance portable code in C++
  ○ Includes abstractions for on-node parallel execution and data layout

● **What happens at Kokkos Bootcamp?**
  ○ New users learn about Kokkos in a tutorial
  ○ All users get hands-on experience with expert help to use Kokkos for their application

● **For more information and to register:**
  https://www.exascaleproject.org/event/kokkosbc2
NERSC Podcast
NERSC Podcast

- New “NERSC News” Podcast
- Available at [https://anchor.fm/nersc-news](https://anchor.fm/nersc-news)
- 10-15 minute segments about latest center news and interesting HPC topics
- 6-month experiment (discontinue if low uptake)
- Your feedback, suggestions for topics, etc., always welcome!