Introduction to Using the Ada Cluster

Francis Dang

HPRC Short Course – Fall 2015

Sep. 22 & 24, 2015
Outline (Day 1)

- Usage Policies
- Hardware Overview of Ada
- Accessing Ada
- Filesystems and User Directories
- Computing Environment
- Parallelization Concepts via an Example
- Batch Processing
  - Batch Queues
  - Batch Job Files and Specifying Resource Requirements
  - Job Submission
  - Job Monitoring
  - Correct and Efficient Mapping of Job Requirements to Cluster Resources
Outline (Day 2)

• Compiling and Running Application Codes
  – Development Environment on Ada
  – Intel Compiler Optimization Options
  – OpenMP (Shared Memory) Programming
  – MPI (Distributed Memory) Programming
  – Linking with the Intel MKL Mathematical Library on Ada

• Remote Visualization Jobs

• Common Problems

• Need Help?

  https://sc.tamu.edu/wiki/index.php/Ada:Compile:All
Introduction

• Prerequisites:
  – Basic knowledge of UNIX/Linux

• Examples:
  – Available in /scratch/training/Intro-to-ada directory
  – Copy these files to your scratch directory!!!
Usage Policies
(Be a good compute citizen)

- It is illegal to share computer passwords and accounts by state law and university regulation;
- It is prohibited to use Ada in any manner that violates the United States export control laws and regulations, EAR & ITAR;
- Abide by the expressed or implied restrictions in using commercial software

Ada – an x86 Cluster

A 17,500-core, 860-node cluster with:

- 837 20-core compute nodes with two Intel 10-core 2.5GHz IvyBridge processors.
  - Among these nodes, 30 nodes have 2 GPUs each and 9 nodes have 2 Phi coprocessors.
- 15 compute nodes are 1TB and 2TB memory, 4-processor SMPs with the Intel 10-core 2.26GHz Westmere processor.
- 8 20-core login nodes with two Intel 10-core 2.5GHz IvyBridge processors and 1 GPU, 2 GPUs, or 2 Phi coprocessors.
- The interconnecting fabric is based on the Mellanox SX6536 IB core switch (the middle rack in the graphic above).

https://sc.tamu.edu/wiki/index.php/Ada:Intro
Ada Schematic: 17,500-core 860-node Cluster

792 64 GB 20-Core Nodes
- 20 – Core IvyBridge Node 64 GB DRAM

15 Extra-Large Memory 40-Core Nodes
- 40 – Core Westmere Node 1 TB DRAM
- 40 – Core Westmere Node 2 TB DRAM

8 Login Nodes
- 20 – Core IvyBridge Node 1 GPU, 2 GPUs, or 2 PHIs 256 GB DRAM

Storage
- IBM GSS26 4PB raw Sep 2014

45 Special Purpose 20-Core Nodes:
- 20 – Core IvyBridge Node 2 GPUs 256/64 GB DRAM
- 20 – Core IvyBridge Node 2 PHIs 64 GB DRAM
- 20 – Core IvyBridge Node 256 GB DRAM

45 Special Purpose 20-Core Nodes:
- 30 w/2 GPUs, 9 w/2 PHIs, 6 w/no accel (256 GB)
Accessing Ada

- SSH is required for accessing Ada:
  - On campus: ssh NetID@ada.tamu.edu
  - Off campus:
    - Set up VPN: http://hdc.tamu.edu/Connecting/VPN/
    - Then: ssh NetID@ada.tamu.edu

- SSH programs for Windows:
  - MobaXTerm (preferred, includes SSH and X11)
  - PuTTY SSH

- Login sessions that are idle for 60 minutes will be closed automatically

File Transfers with Ada

● Simple File Transfers:
  - scp: command line  (Linux, MacOS)
  - WinSCP: GUI  (Windows)
  - FileZilla: GUI  (Windows, MacOS, Linux)

● Bulk data transfers:
  - Use fast transfer nodes (FTN) with:
    ● GridFTP
    ● Globus Connect
    ● bbcp

https://sc.tamu.edu/wiki/index.php/Ada:Filesystems_and_Files#Transfering_Files
File Systems and User Directories

<table>
<thead>
<tr>
<th>Directory</th>
<th>Environment Variable</th>
<th>Quota Limit</th>
<th>Intended Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home/$USER</td>
<td>$HOME</td>
<td>10 GB</td>
<td>Small to modest amounts of processing.</td>
</tr>
<tr>
<td>/scratch/user/$USER</td>
<td>$SCRATCH</td>
<td>1 TB</td>
<td>Temporary storage of large files for ongoing computations. Not intended to be a long-term storage area.</td>
</tr>
</tbody>
</table>

- View usage and quota limits: the showquota command
- Also, only home directories are backed up daily.

https://sc.tamu.edu/wiki/index.php/Ada:Filesystems_and_Files
Computing Environment

• Paths:
  – $PATH: for commands (eg. /bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/netid/bin)
  – $LD_LIBRARY_PATH: for libraries

• Many applications, many versions, and many paths ....... How do you manage them?!

• Modules
  – Help: module help
  – Search: module spider [package_name]
  – Load: module load package1 package2 ... (case sensitive)
  – List: module list
  – Unload: module unload package1 package2 ...
  – Purge: module purge

https://sc.tamu.edu/wiki/index.php/Ada:Computing_Environment#Modules
Batch Computing on Ada

On-campus:

- Campus Network
- SSH
- Login nodes

Off-campus:

- VPN
- Internet

Create job

Submit job

Job file

Queue

LSF (batch manager)

Cluster

Output Files
Job Files (Serial Example)

```
#BSUB -J myjob1            # sets the job name to myjob1.
#BSUB -L /bin/bash        # uses the bash login shell to initialize the job's execution environment
#BSUB -W 12:30            # sets to 12.5 hours the job's runtime wall-clock limit.
#BSUB -n 1                # assigns 1 core for execution.
#BSUB -o stdout1.%J       # directs the job's standard output to stdout1.jobid
#BSUB -M 500              # specifies a memory limit of 500 MB per core/process
#BSUB -R "rusage[mem=500]" # requests 500MB of memory per core/process from LSF
#
# <--- at this point the current working directory is the one you submitted the job from.
#
module load intel        # loads the INTEL software tool chain to provide, among other things,
#                          needed runtime libraries for the execution of prog.exe below.
#                          (assumes prog.exe was compiled using INTEL compilers.)
#
prog.exe < input1 >& data_out1    # both input1 and data_out1 reside in the job submission dir
##
```

https://sc.tamu.edu/wiki/index.php/Ada:Batch#Job_files
# Job Submission and Tracking

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bsub &lt; jobfile1</td>
<td>Submit jobfile1 to batch system</td>
</tr>
<tr>
<td>bjobs [-u all or user_name] [[-l] job_id]</td>
<td>List jobs</td>
</tr>
<tr>
<td>bpeek [-f] job_id</td>
<td>View job's output and error files</td>
</tr>
<tr>
<td>bkill job_id</td>
<td>Kill a job</td>
</tr>
<tr>
<td>bhist [-l] job_id</td>
<td>Show historical information about a job</td>
</tr>
<tr>
<td>lnu [-l] -j job_id</td>
<td>Show resource usage for a job</td>
</tr>
</tbody>
</table>

[https://sc.tamu.edu/wiki/index.php/Ada:Batch#Job_tracking_and_control_commands](https://sc.tamu.edu/wiki/index.php/Ada:Batch#Job_tracking_and_control_commands)
Job Environment Variables

- $LSB_JOBID = job id
- $LS_SUBCWD = directory where job was submitted from
- $SCRATCH = /scratch/user/NetID
- $TMPDIR = /work/$LSB_JOBID.tmpdir
  - $TMPDIR is local to each assigned compute node for the job

https://sc.tamu.edu/wiki/index.php/Ada:Batch#Environment_Variables
Important Job Parameters

#BSUB -n NNN
  # NNN: total number of cores or job slots to allocate for the job

#BSUB -R "span[ptile=XX]"
  # XX: number of cores or job slots per node to use

#BSUB -R "select[node-type]"
  # node-type: nxt, mem256gb, gpu, phi, mem1tb, mem2tb ...

#BSUB -R "rusage[mem=nnn]"
  # reserves nnn MBs per process/core for the job

#BSUB -M nnn
  # enforces (XX cores * nnn MB) as memory limit
  # per node for the job

#BSUB -W hh:mm or mm
  # sets job's runtime wall-clock limit in hours:minutes or just minutes
Job Slots – ptile Examples

#BSUB –n 10 –R “span[ptile=2]” ...

will allocate 10 job slots, 2 per node. That is, the job will span 5 nodes;

#BSUB –n 80 –R “span[ptile=20]” ...

will allocate 4 whole nodes (80/20), not including the X-large memory ones.
Job Memory Requests

• Requesting memory via
  
  \#BSUB -R "rusage[mem=process_alloc_size]"
  
  \#BSUB -M process_size_limit

• Default 2.5 GB per process (core) if -R/-M not specified, but it might cause memory contention when sharing a node with other jobs.

• On NeXtScale (nxt) nodes, usable memory is at best 54 GB (out of total 64 GB). That means the per process memory limit should not exceed 2700 MB for a 20-core job.

• If more memory is needed, request the large memory nodes:
  
  - If under 256 GB and up to 20 cores per node: use -R “select[mem256gb]”
  - If need up to 1 or 2 TB of memory or up to 40 cores:
    • use -R “select[mem1tb]” (40 cores) or -R “select[mem2tb]” with the -q xlarge option
    • The mem1tb and mem2tb nodes are accessible only via the xlarge queue.
Batch Queues

- Job submissions are assigned to batch queues based on the resources requested (number of cores/nodes and wall-clock limit)
- Some jobs can be directly submitted to a queue:
  - If the 1TB or 2TB nodes are needed, use the xlarge queue
  - Jobs that have special resource requirements are scheduled in the special queue (must request access to use this queue)
- Batch queue policies are used to manage the workload and may be adjusted periodically.

https://sc.tamu.edu/wiki/index.php/Ada:Batch_Queues
### Current Queues

$ bqueues$

<table>
<thead>
<tr>
<th>QUEUE_NAME</th>
<th>PRIO</th>
<th>STATUS</th>
<th>MAX</th>
<th>JL/U</th>
<th>JL/P</th>
<th>JL/H</th>
<th>NJOBS</th>
<th>PEND</th>
<th>RUN</th>
<th>SUSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>staff</td>
<td>450</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>special</td>
<td>400</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>9608</td>
<td>3500</td>
<td>6108</td>
<td>0</td>
</tr>
<tr>
<td>xlarge</td>
<td>100</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>vnc</td>
<td>90</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>sn_short</td>
<td>80</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>840</td>
<td>840</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mn_large</td>
<td>80</td>
<td>Open:Active</td>
<td>3000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1000</td>
<td>1000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mn_short</td>
<td>80</td>
<td>Open:Active</td>
<td>500</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>general</td>
<td>50</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sn_regular</td>
<td>50</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5246</td>
<td>4360</td>
<td>886</td>
<td>0</td>
</tr>
<tr>
<td>sn_long</td>
<td>50</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1949</td>
<td>980</td>
<td>969</td>
<td>0</td>
</tr>
<tr>
<td>sn_xlong</td>
<td>50</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>240</td>
<td>100</td>
<td>140</td>
<td>0</td>
</tr>
<tr>
<td>mn_small</td>
<td>50</td>
<td>Open:Active</td>
<td>6000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11877</td>
<td>7308</td>
<td>4569</td>
<td>0</td>
</tr>
<tr>
<td>mn_medium</td>
<td>50</td>
<td>Open:Active</td>
<td>4000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8988</td>
<td>5344</td>
<td>3644</td>
<td>0</td>
</tr>
<tr>
<td>curie_devel</td>
<td>40</td>
<td>Open:Active</td>
<td>32</td>
<td>32</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>curie_medium</td>
<td>35</td>
<td>Open:Active</td>
<td>512</td>
<td>192</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>curie_long</td>
<td>30</td>
<td>Open:Active</td>
<td>192</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>curie_general</td>
<td>25</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

# Queue Limits

<table>
<thead>
<tr>
<th>Queue</th>
<th>Min/Default/Max Cpus</th>
<th>Default/Max Walltime</th>
<th>Compute Node Types</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>sn_short</td>
<td>1 / 1 / 20</td>
<td>10 min / 1 hr</td>
<td></td>
<td>Maximum of 8000 cores for all running jobs in the single-node (sn_<em>) queues. Maximum of 600 cores for all running jobs per user in the sn_</em> queues.</td>
</tr>
<tr>
<td>sn_regular</td>
<td>1 hr / 1 day</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sn_long</td>
<td>1 hr / 1 day</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sn_xlong</td>
<td>4 days / 7 days</td>
<td></td>
<td>64 GB and 256 GB nodes</td>
<td>Maximum of 500 cores for all running jobs in this queue.</td>
</tr>
<tr>
<td>mn_short</td>
<td>2 / 2 / 200</td>
<td>10 min / 1 hr</td>
<td>64 GB and 256 GB nodes</td>
<td>Maximum of 500 cores for all running jobs in this queue.</td>
</tr>
<tr>
<td>mn_small</td>
<td>2 / 2 / 120</td>
<td>1 hr / 7 days</td>
<td></td>
<td>Maximum of 4000 cores for all running jobs in this queue.</td>
</tr>
<tr>
<td>mn_medium</td>
<td>121 / 121 / 600</td>
<td>1 hr / 7 days</td>
<td></td>
<td>Maximum of 3000 cores for all running jobs in this queue.</td>
</tr>
<tr>
<td>mn_large</td>
<td>600 / 601 / 2000</td>
<td>1 hr / 2 days</td>
<td></td>
<td>Maximum of 3000 cores for all running jobs in this queue.</td>
</tr>
<tr>
<td>xlarge</td>
<td>1 / 1 / 280</td>
<td>1 hr / 10 days</td>
<td>1 TB nodes (11) 2 TB nodes (4)</td>
<td></td>
</tr>
<tr>
<td>vnc</td>
<td>1 / 1 / 20</td>
<td>1 hr / 6 hr</td>
<td>All 30 nodes with GPUs</td>
<td>For remote visualization jobs.</td>
</tr>
<tr>
<td>special</td>
<td>None</td>
<td>1 hr / 7 days</td>
<td>64 GB and 256 GB nodes</td>
<td>Requires permission to access this queue.</td>
</tr>
</tbody>
</table>

[https://sc.tamu.edu/wiki/index.php/Ada:Batch_ Queues](https://sc.tamu.edu/wiki/index.php/Ada:Batch_ Queues)
Job Files (Concurrent Serial Runs)

#BSUB -J myjob2
# sets the job name to myjob2.
#BSUB -L /bin/bash
# uses the bash login shell to initialize the job's execution environment.
#BSUB -W 12:30
# sets to 12.5 hours the job's runtime wall-clock limit.
#BSUB -n 3
# assigns 3 cores for execution.
#BSUB -R "span[ptile=3]"
# assigns 3 cores per node.
#BSUB -R "rusage[mem=5000]"
# reserves 5000MB per process/CPU for the job (i.e., 15,000 MB for job/node)
#BSUB -M 5000
# sets to 5,000MB (~5GB) the per process enforceable memory limit.
#BSUB -o stdout2.%J
# directs the job's standard output to stdout2.jobid
#BSUB -P project1
# charges the consumed service units (SUs) to project1.
#BSUB -u e-mail_address
# sends email to the specified address
#BSUB -B -N
# send emails on job start (-B) and end (-N)
###
cd $SCRATCH/myjob2
# makes $SCRATCH/myjob2 the job's current working directory
module load intel
# loads the INTEL software tool chain to provide, among other things,
# The next 3 lines concurrently execute 3 instances of the same program, prog.exe, with standard input and output data streams assigned to different files in each case.
(prog.exe < input1 >& data_out1 ) &
(prog.exe < input2 >& data_out2 ) &
(prog.exe < input3 >& data_out3 )
wait

https://sc.tamu.edu/wiki/index.php/Ada:Batch#Job_files
OpenMP Jobs

- Must set `OMP_NUM_THREADS` to take advantage of the requested cores
- All processes run on the same node.
  - Submit to the xlarge queue if you need up to 40 cores per node

```bash
#BSUB -n 20 -R 'rusage[mem=300] span[ptile=20]' -M 300
#BSUB -J omp_helloWorld
#BSUB -o omp_helloWorld.%J -L /bin/bash -W 20
#
module load intel
#
ifort -openmp -o omp_helloWorld.exe omp_helloWorld.f90
#
export OMP_NUM_THREADS=20
./omp_helloWorld.exe
```
MPI Jobs

- Programs may be run on multiple nodes

```bash
#BSUB -n 12 -R 'rusage[mem=150] span[ptile=4]' -M 150
#BSUB -J mpi_helloWorld -o mpi_helloWorld.%J -L
/bin/bash -W 20
#
module load intel
#
mpiifort -o mpi_helloWorld.exe mpi_helloWorld.f90
#
mpiexec.hydra -np 12 ./mpi_helloWorld.exe
```
Concurrent Program Execution in Jobs via Tamulauncher

- Useful for running many programs concurrently across multiple nodes within a job
- Can be used with serial or multi-threaded programs
- Distributes a set of commands from an input file to run on the cores assigned to a job
- Can only be used in batch jobs
- If a tamulauncher job gets killed, you can resubmit the same job to complete the unfinished commands in the input file
- Preferred over LSF job arrays

https://sc.tamu.edu/wiki/index.php/Ada:Tamulauncher
Outline (Day 2)

• Compiling and Running Application Codes
  – Development Environment on Ada
  – Intel Compiler Optimization Options
  – OpenMP (Shared Memory) Programming
  – MPI (Distributed Memory) Programming
  – Linking with the Intel MKL Mathematical Library on Ada

• Remote Visualization Jobs

• Common Problems

• Need Help?
  https://sc.tamu.edu/wiki/index.php/Ada:Compile:All
Development Environment

- Intel software stack is recommended, which includes:
  - Intel C/C++/Fortran compilers
  - Intel Math Kernel Library
  - Intel MPI library
- Load/use the **intel** module
- The commands to invoke each compiler are:
  - icc for C
  - icpc for C++
  - ifort for Fortran
- Man pages (documentation) are available for each compiler:
  - man icc
  - man icpc
  - man ifort
- Help for compiler options also available with -help option.
  - Also organized by categories (see icc -help help for more information).

https://sc.tamu.edu/wiki/index.php/Ada:Compile:All#Getting_Started
Compiling Basics

• Generally provide the compiler:
  - source file(s) and/or object file(s)
  - compilation option(s)
  - optionally a name for the resulting executable. Default executable name is a.out if no name provided.

• Example:
  icc objfile.o subroutine.c main.c

https://sc.tamu.edu/wiki/index.php/Ada:Compile:All#Getting_Started
# Basic Compiler Flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-help [category]</td>
<td>Shows all available compiler options or all options under a specified category</td>
</tr>
<tr>
<td>-o &lt;file&gt;</td>
<td>Specifies the name for an object file. For an executable, the -output filename will be &lt;file&gt; instead of a.out</td>
</tr>
<tr>
<td>-c</td>
<td>Only compile the source file(s). Linking phase will be skipped.</td>
</tr>
<tr>
<td>-L &lt;dir&gt;</td>
<td>Tells the linker to search for libraries in directory &lt;dir&gt; ahead of the standard library directories.</td>
</tr>
<tr>
<td>-l&lt;name&gt;</td>
<td>Tells the linker to search for library named libname.so or libname.a</td>
</tr>
</tbody>
</table>

**Examples:**
- icc -o mprog.x subroutine.c myobjs.o main.c
- icc -L mylibs -lmyutils main.c

[https://sc.tamu.edu/wiki/index.php/Ada:Compile:All#Basic_compiler_flags](https://sc.tamu.edu/wiki/index.php/Ada:Compile:All#Basic_compiler_flags)
# Compiler Optimization Flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-O2</td>
<td>Default optimization level (includes inlining, constant/copy propagation, loop unrolling, peephole optimizations, etc)</td>
</tr>
<tr>
<td>-O3</td>
<td>Enables more aggressive loop transformations in addition to -O2 optimizations.</td>
</tr>
<tr>
<td>-xHost</td>
<td>Tells the compiler to generate vector instructions for the highest instruction set available on the host machine.</td>
</tr>
<tr>
<td>-fast</td>
<td>Shortcut for -ipo, -O3, -no-prec-div, -static, and -xHost flags.</td>
</tr>
<tr>
<td>-ip</td>
<td>Perform inter-procedural optimization within the same file.</td>
</tr>
<tr>
<td>-ipo</td>
<td>Perform inter-procedural optimization between files.</td>
</tr>
<tr>
<td>-parallel</td>
<td>Enable automatic parallelization by the compiler (very conservative)</td>
</tr>
<tr>
<td>-opt-report=[n]</td>
<td>Generate optimization report. n represent the level of detail (0..3, 3 being most detailed)</td>
</tr>
<tr>
<td>-vec-report[=n]</td>
<td>Generate vectorization report. n represents the level of detail (0..7, 7 being most detailed)</td>
</tr>
</tbody>
</table>

For more information, consult the opt, advanced, and ipo compiler help categories.
Other Compiler Flags

• Debugging flags:
  – [link](https://sc.tamu.edu/wiki/index.php/Ada:Compile:All#Debugging_flags)
  – See also the `icc -help` command which includes debugging and other flags.

• Flags affecting floating point operations:
  – [link](https://sc.tamu.edu/wiki/index.php/Ada:Compile:All#Flags_affecting_floating_point_operations)
  – See also the `icc -help float help` or the `ifort -help float` commands. Some floating point flags are specific to Fortran.

• Many more compiler flags. Consult each compiler's man page or the output from the compiler's `-help` option.
Upcoming Programming Shortcourses

- **Intro. to Code Parallelization using OpenMP**
  - 3:00 - 5:00 p.m Sep 29, 30, & Oct 1
  - [http://sc.tamu.edu/shortcourses/SC-openmp/](http://sc.tamu.edu/shortcourses/SC-openmp/)

- **Intro. to Code Parallelization using MPI**
  - 3:00 - 5:00 p.m Oct 6, 7, & 8
  - [http://sc.tamu.edu/shortcourses/SC-MPI/](http://sc.tamu.edu/shortcourses/SC-MPI/)
Compiling OpenMP Programs

• OpenMP programming:
  – Use compiler directives to specify which code regions to run in parallel
  – Compiler generates multi-threaded code for these code regions

• Example:

  module load intel
  ifort -openmp -o omp_helloWorld.exe
  omp_helloWorld.f90

https://sc.tamu.edu/wiki/index.php/Ada:Compile:OpenMP
Running OpenMP Programs

- Common environment variables:
  - OMP_NUM_THREADS:
    - Sets the maximum number of threads per nesting level
    - Default value is 1
  - OMP_STACKSIZE:
    - Sets the size for the private stack of each worker thread. Suffix can be B,K,M,G
    - Default value is 4 MB

- Example using 4 threads and 16 MB stack size per thread
  - $ export OMP_NUM_THREADS=4
  - $ export OMP_STACKSIZE=16M
  - $ ./omp_helloWorld.exe

- Do not use more than 4 cores on the login nodes!
OpenMP Jobs

• Must set \texttt{OMP\_NUM\_THREADS} to take advantage of the requested cores
• All processes run on the same node.
  – Submit to the xlarge queue if you need up to 40 cores per node
• Example job:
  
  \texttt{#BSUB -n 20 -R 'rusage[mem=300] span[ptile=20]' -M 300}
  \texttt{#BSUB -J omp\_helloWorld}
  \texttt{#BSUB -o omp\_helloWorld.\%J -L /bin/bash -W 20}
  
  module load intel
  ifort -openmp -o omp\_helloWorld.exe omp\_helloWorld.f90
  \texttt{export OMP\_NUM\_THREADS=20}
  \texttt{./omp\_helloWorld.exe}
Compiling MPI Programs

• Use a MPI compiler wrapper to compile MPI codes.
  – Wrapper invokes underlying compiler and adds linker flags specific for MPI programs
  – Intel MPI provides wrappers for both Intel and GNU compilers
  – Any flags not recognized by the wrapper are passed to the underlying compiler.

• Example to compile MPI C program with the Intel compiler's -O3 optimization flag

  mpiicc -o mpi_prog.x -O3 mpi_prog.c

https://sc.tamu.edu/wiki/index.php/Ada:Compile:MPI
Running MPI Programs

- Requires a MPI launcher (mpirun) to run MPI programs

  `mpirun [mpi_flags] executable [executable params]`

- Example:

  `module load intel`
  `mpirun -np 4 ./mpi_helloWorld.exe`

- Do not use more than 4 cores on the login nodes!
MPI Jobs

• MPI programs may be run in batch jobs on multiple nodes

• Note, the mpiexec -np option must match the number of cores requested by the job (#BSUB -n option).

#BSUB -n 12 -R 'rusage[mem=150] span[ptile=4]' -M 150
#BSUB -J mpi_helloWorld -o mpi_helloWorld.%J
#BSUB -L /bin/bash -W 20
#
module load intel
mpiifort -o mpi_helloWorld.exe mpi_helloWorld.f90
mpiexec.hydra -np 12 ./mpi_helloWorld.exe
Other Programming Methods

- Compiling programs to use GPU accelerators

- Compiling programs to use Phi coprocessors
Intel Math Library (MKL)

- Provides optimized and threaded math routines such as BLAS, LAPACK, sparse solvers, FFTs, vector math, and more.
- Offers sequential, parallel, and cluster versions.
- Examples:
  - ifort example.f -mkl=sequential -o example.exe
  - icc example.c -mkl=parallel -o example.exe
  - mpiifort example.f -mkl=cluster -o example.exe
- Consult Intel MKL Link advisor for usage help:
  - https://sc.tamu.edu/wiki/index.php/Ada:MKL
Remote Visualization Jobs

- Use to run programs with graphical interfaces on Ada and display them on your computer:
- Can leverage GPU nodes for better graphics performance
- Better than X11 forwarding (especially when using VPN)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vncjob.submit [-h] [-g MxN] [-t type]</td>
<td>Submit a VNC job. Type 'vncjob.submit -h' for help</td>
</tr>
<tr>
<td>vncjob.kill JOBID</td>
<td>Kill a VNC job whose id is JOBID</td>
</tr>
<tr>
<td>vncjob.list</td>
<td>List all your VNC jobs currently in the batch system</td>
</tr>
</tbody>
</table>

https://sc.tamu.edu/wiki/index.php/Ada:Remote_Visualization
Remote Visualization Job Example

(1) Log into Ada

Your current disk quotas are:

<table>
<thead>
<tr>
<th>Disk</th>
<th>Disk Usage</th>
<th>Limit</th>
<th>File Usage</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home</td>
<td>33M</td>
<td>10G</td>
<td>676</td>
<td>10000</td>
</tr>
<tr>
<td>/scratch</td>
<td>4.533G</td>
<td>1T</td>
<td>13749</td>
<td>50000</td>
</tr>
<tr>
<td>/tiered</td>
<td>0</td>
<td>10T</td>
<td>1</td>
<td>50000</td>
</tr>
</tbody>
</table>

Type 'showquota' to view these quotas again.

[ netid@ada2 ~]$ 

(2) Submit VNC Job using vncjob.submit (optional parameters available)

Type 'showquota' to view these quotas again.

[ netid@ada2 ~]$ vncjob.submit
Your vnc job has been submitted.

Output file for VNC job 1551326 will be /home/ /vncjob.1551326.

View the output with the following command when your job starts running

cat /home/ netid /vncjob.1551326

For more information about remote visualization on ada, please visit

(3) Use cat to see the output file -- Note job properties

```
[ netid@ada2 ~]$ cat /home/netid /vncjob.1551326
Using settings in ~/.vnc/xstartup.turboVNC to start /opt/TurboVNC/bin/vncserver
VNC batch job id is 1551326
VNC server arguments will be '-geometry 1024x768'
VNC server started with display gpu64-3001:11
```

VirtualGL Client 64-bit v2.4 (Build 20150126)
Listening for unencrypted connections on port 4242
4242

WARNING: You have started an interactive/VNC job. Your job will continue
to run until the VNC server is stopped (up to 6 hours).

To access from Mac/Linux, run from your desktop:

```
vncviewer -via netid@ada.tamu.edu gpu64-3001:11
```

To access from Windows:

1) Setup a tunnel from your machine to gpu64-3001:5911

1.1) If you use MobaXterm, run the following command in the MobaXterm terminal:

```
ssh -f -N -L 10000:gpu64-3001:5911 netid@ada.tamu.edu
```

1.2) If you use Putty to set up the tunnel, click SSH and then click 'Tunnels'.
Fill in 'Source port' with '10000' and 'Destination with 'gpu64-3001:5911''

2) Start vncviewer on your machine

Otherwise to access from Windows, either see the documentation that came
with your VNC viewer, or open an X11 enabled login to ada.tamu.edu and
then run:

```
vncviewer gpu64-3001:11
```

When running graphical program in this VNC job, remember to start them using vgllrun:

```
vgllrun application
```

To stop the VNC job:

```
vncjob.kill 1551326
```

[ netid@ada2 ~]$
Remote Visualization Job Example

(4) Start new tab/terminal pointed to local machine

(5) Use command from (3) to create tunnel -- Local port 10000 must be free

NOTICE: This computer system and data herein are available only for authorized purposes by authorized users. Use for any other purpose may result in administrative/disciplinary actions or criminal prosecution against the user. Usage may be subject to security testing and monitoring. Applicable privacy laws establish the expectations of privacy.

netid@ada.tamu.edu's password:

Sep. 22 & 24, 2015
Remote Visualization Job Example

(6) Open VNC Viewer and enter connection information

(7) Enter your VNC password
Remote Visualization Job Example

(8) VNC window opens -- load modules -- use vglrun to launch GUI

Example output:

```
$ netid@gpu64-3001 ~$ module load ABAQUS
$ netid@gpu64-3001 ~$ vglrun abaqus cae
Abaqus License Manager checked out the following licenses:
"cae" release 6.13 from sclserver2.tamu.edu
<28 out of 30 licenses remain available>.
```
Common Problems

- Control characters (^M) in job files edited with DOS/Windows editors
  - Use the 'dos2unix jobfile' command to remove ^M characters
- Did not load required module(s)
- Missing program, input files
- Insufficient walltime
- Insufficient memory
- No matching resource (-R rusage[mem] too large)
- Running OpenMP jobs across nodes
- Insufficient disk quotas
- Using GUI-based software without setting up X11 forwarding
- Software license availability
Need Help?

- Email your questions to help@sc.tamu.edu.
- Help us, help you -- we need more info
  - Which Cluster
  - UserID/NetID (UIN is not needed!)
  - Job id(s) if any
  - Location of your jobfile, input/output files
  - Application used if any
  - Module(s) loaded if any
  - Error messages
  - Steps you have taken, so we can reproduce the problem
- Or visit us @ 104 Teague. Making an appointment is recommended.