Introduction to the PDC Environment

Radovan Bast

PDC Center for High Performance Computing & Department of Theoretical Chemistry and Biology, KTH Royal Institute of Technology, Stockholm

http://rbast.github.io/talks/pdc-env-2014/

Thanks

- Slides are based on material by Izhar ul Hassan and Henric Zazzi.
- A. Turner, X. Guo, L. Axner, M. Filipiak, Best Practice Guide - Cray XE/XC (V4.1), 2013 (http://www.prace-ri.eu/Best-Practice-Guide-Cray-XE-XC-HTML).
- http://software-carpentry.org/v5/novice/shell/index.html

Outline

- System architecture: Milner (CPU) and Zorn (GPU)
- System access
- Unix shell (bash)
- File systems and permissions
- Programming environment
- How to run programs
- Performance analysis and debugging
- How to get help

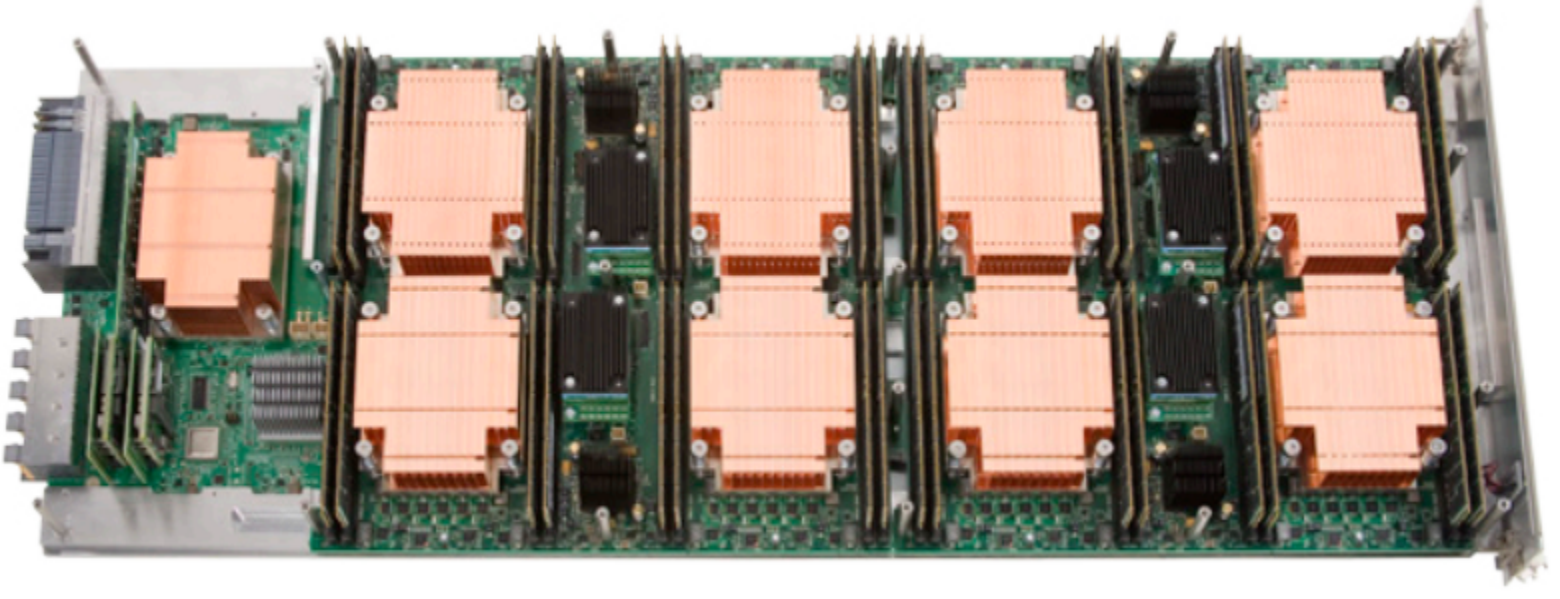
Milner

Cray XC30

- Login node: milner.pdc.kth.se
- One computing cabinet
- 120 compute nodes
- 20 cores per node, divided between 2 sockets with 10 cores each
- Intel Ivy Bridge 10 core processors at 2.5 GHz
- 2400 compute cores
- Cray Aries interconnect
- One storage cabinet
- 12 service nodes
- Aggregate peak performance is 48 TF
- Aggregate compute memory is 3.75 TB (32 GB per node)
- Lustre file system with 150 TB
- HPC optimized Cray Linux Environment based on SuSE Linux

Milner

- Blade with 4 nodes
- 20 cores per node
- Hyperthreading: up to 40 MPI processes per node (default)



Zorn

GPU cluster

- Login node: zorn.pdc.kth.se
- 8 Nodes with 92 GB RAM, 2 CPU Intel Xeon E5620 (Nehalem/Westmere), 3 NVIDIA Tesla M2090, Nvidia CUDA Toolkit V.5.5 (default queue)
- 1 Node with 48 GB RAM, 2 CPU Intel Xeon E5620 (Nehalem/Westmere), 1 NVIDIA Tesla K20, 1 NVIDIA Tesla C2050, Nvidia CUDA Toolkit V.5.5 ("kepler" queue)
- QDR Infiniband interconnect
- Lustre file system with 15 TB
- CentOS 6.5 derived from Red Hat Enterprise Linux

System access

System access

- PDC uses SSH together with Kerberos for authentication and login
- Kerberos protocol for authentication developed at MIT
- Non-US version developed at KTH
- Uses tickets to authenticate

Kerberos jargon

- **Ticket**: proof of identity encrypted with a secret key for the particular service requested (tickets have a lifetime)
- **Realm**: collection of resources that authenticate against a central user/service database
- **Principal**: unique identity to which Kerberos assigns tickets
- **Key Distribution Center (KDC)**: service which stores encrypted secret keys

Kerberos

- Avoids storing passwords locally or sending them over the internet
- Avoids re-typing passwords
- Involves a trusted 3rd-party
- Built on symmetric-key cryptography
- Offers less attack vectors than SSH with passwords
- Kerberos v5 software (Heimdal or MIT) needed to get Kerberos tickets
- Need kerberized SSH software that supports GSSAPI with KeyExchange
- For installation instructions see https://www.pdc.kth.se/resources/software/login-1

Kerberos

Linux

- Installation: https://www.pdc.kth.se/resources/software/login-1/linux/
- Login

```
# without editing ~/.ssh/config
kinit --forwardable user@NADA.KTH.SE
ssh -o GSSAPIKeyExchange=yes -o GSSAPIAuthentication=yes \
    -o GSSAPIDelegateCredentials=yes user@machine.pdc.kth.se

# with editing ~/.ssh/config; see PDC web
kinit --forwardable user@NADA.KTH.SE
ssh user@machine.pdc.kth.se

# after editing krb5.conf (see PDC web) you can drop the realm
kinit user
ssh user@machine.pdc.kth.se
```

- It is convenient to insert the following into `~/.ssh/config`:

```
# Hosts we want to authenticate to with Kerberos
Host *.kth.se *.kth.se.
# User authentication based on GSSAPI is allowed
GSSAPIAuthentication yes
# Key exchange based on GSSAPI may be used for server authentication
GSSAPIKeyExchange yes

# Hosts to which we want to delegate credentials. Try to limit this to
# hosts you trust, and were you really have use for forwarded tickets.
Host *.csc.kth.se *.csc.kth.se. *.nada.kth.se *.nada.kth.se. *.pdc.kth.se *.pdc.kth.se.
# Forward (delegate) credentials (tickets) to the server.
GSSAPIDelegateCredentials yes
# Prefer GSSAPI key exchange
PreferredAuthentications gssapi-keyex,gssapi-with-mic

# All other hosts
Host *
```

- For the configuration of krb5.conf, see https://www.pdc.kth.se/resources/software/login-1

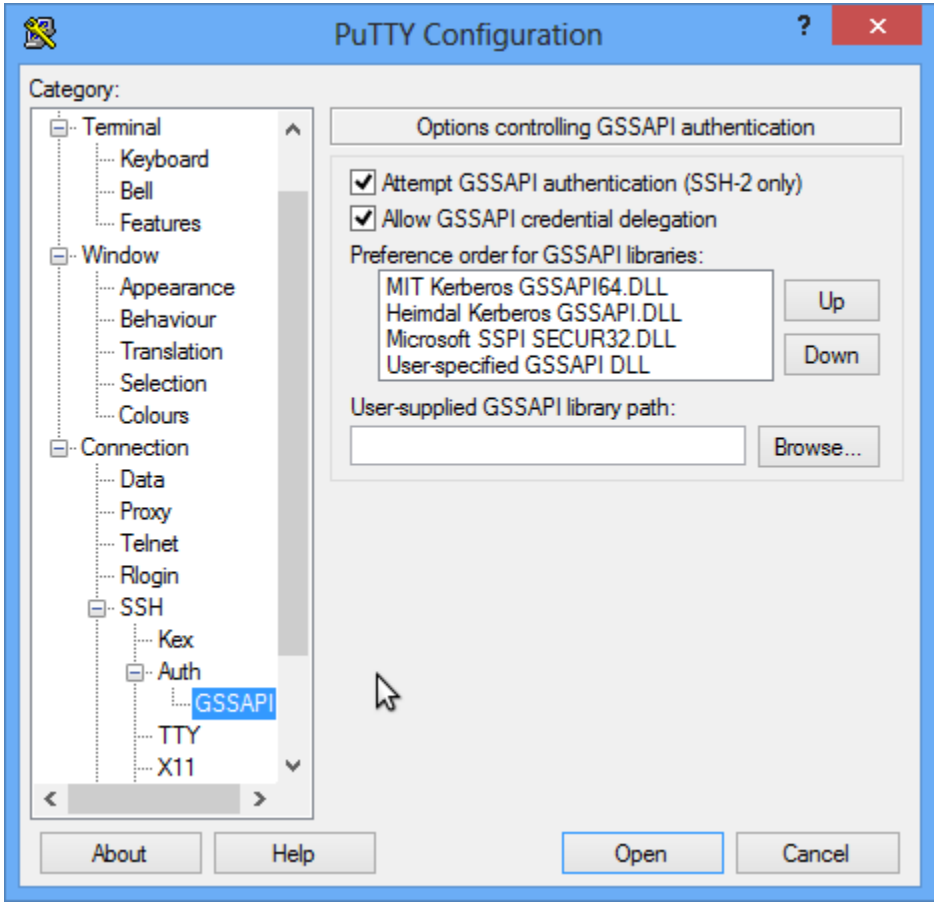
Kerberos

Windows

- PuTTY (only PuTTY downloaded from this link works at PDC) https://www.pdc.kth.se/resources/software/login-1/windows/putty
- SecureCRT https://www.pdc.kth.se/resources/software/login-1/windows/securecrt
- We prefer PuTTY because we prefer to pay less license fees
- Another option is CygWin

Kerberized PuTTY

Windows



Kerberos

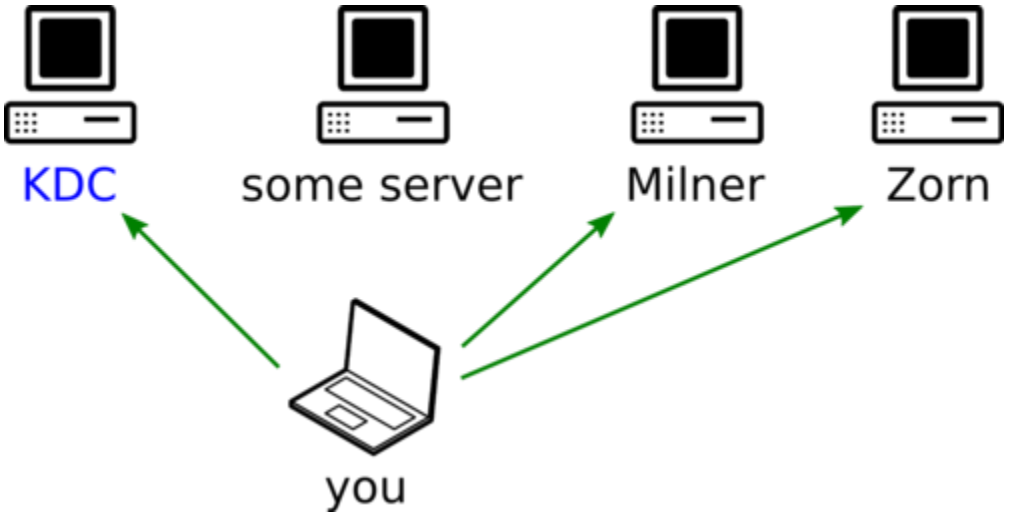
Mac OS X

- Mac OS X Lion 10.7.2 and above come with Heimdal and kerberized SSH
- Mac OS X Snow Leopard comes with MIT Kerberos and kerberized SSH
- For configuration of see https://www.pdc.kth.se/resources/software/login-1/macintosh

Working with Kerberos

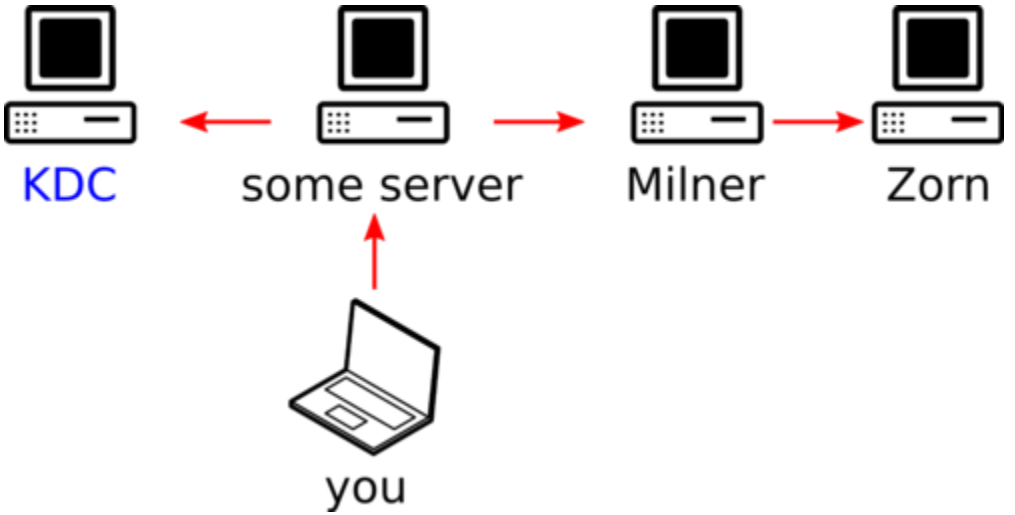
```
# get a new forwardable ticket
kinit --forwardable user@NADA.KTH.SE

# list available tickets
klist

# destroy tickets
kdestroy

# change password
kpasswd
```

Good practice



- Only type your password on your local machine that has an up-to-date OS and that you trust

Bad practice



- Do not kinit on a remote machine
- PDC accounts cannot be shared

Introducing the unix shell

Introducing the unix shell

- We are greeted with a command-line interface (CLI)
- Teleported back to the 70ies

```
$ ssh user@milner.pdc.kth.se
Last login: Fri Aug  8 10:14:59 2014 from example.com
user@milner-login1:~> _
```

- CLI often more efficient than GUI
- High action-to-keystroke ratio (expense: terse and "cryptic")
- Creativity through pipelines
- System is configured with text files
- Calculations are configured and run using text files
- Good for working over network
- Good for reproducibility
- Good for unsupervised workflows

Bash: Files and directories

- Command `pwd` tells me where I am. After login I am in the "home" directory:

```
user@machine:~$ pwd
/afs/pdc.kth.se/home/u/user
```

- I can change the directory with `cd`:

```
user@machine:~$ cd tmp/talks/
user@machine:~/tmp/talks$ pwd
/afs/pdc.kth.se/home/u/user/tmp/talks
```

- I can go one level up with `cd ..`
- List the contents with `ls -l`:

```
user@machine:~/tmp/talks$ ls -l
total 237
drwx------ 3 user csc-users   2048 Aug 17 15:21 img
-rw------- 1 user csc-users  18084 Aug 17 15:21 pdc-env.html
-rw------- 1 user csc-users 222051 Aug 17 15:22 remark-latest.min.js
```

- Files and directories form a tree:

```
├── img
│   ├── kerberos
│   │   ├── computer.png
│   │   ├── kerberos-bad.png
│   │   ├── kerberos-good.png
│   │   ├── kerberos.svg
│   │   ├── laptop.png
│   │   └── putty-kerberos.png
│   ├── xc30-blade.png
│   └── xc30-cabinet.jpg
├── pdc-env.html
└── remark-latest.min.js
```

- We can explore the tree with `ls`, and `cd`:

```
user@machine:~/tmp/talks$ ls -l img/
total 343
drwx------ 2 user csc-users   2048 Aug 17 15:21 kerberos
-rw------- 1 user csc-users 310579 Aug 17 15:21 xc30-blade.png
-rw------- 1 user csc-users  37812 Aug 17 15:21 xc30-cabinet.jpg
```

- All these commands bring you back to home:

```
$ cd $HOME
$ cd ~
$ cd
$ cd /afs/pdc.kth.se/home/u/user
```

Bash: Creating directories and files

- We create a new directory called "results" and change to it:

```
$ mkdir results
$ cd results
```

Creating and editing files

- Easy but not powerful:

```
$ nano draft.txt
```

- More powerful: Emacs or Vi(m):

```
$ emacs draft.txt
$ vi    draft.txt
$ vim   draft.txt # this is Vi "improved"
```

Copying, moving, renaming, and deleting

```
# copy file
$ cp draft.txt backup.txt

# recursively copy directory
$ cp -r results backup

# move/rename file
$ mv draft.txt draft_2.txt

# move/rename directory
$ mv results backup

# move directory one level up
$ mv results ..

# remove file
$ rm draft.txt

# remove directory and all its contents
$ rm -r results
```

- From the Unix point of view, deleting is forever!

Bash: History and tab completion

- From the Unix point of view, deleting is forever!

```
$ history

1860  vi /home/user/devel/gpunch/src/twoints/EriBlock.cpp
1861  cd ..
1862  git grep GenPrimSSSD
1863  vi src/twoints/EriBlock.h
1864  find . | grep SSSD
1865  vi ./src/twoints/GenPrimSSSD.h
1866  git pull
1867  cd build/
1868  make -j12
```

- Commands are numbered, I can repeat a command by number:

```
!1864
```

- Use the tab key for tab completion

Bash: Finding things

- Extract lines which contain an expression with `grep`:

```
# extract all lines that contain "fixme"
$ grep fixme draft.txt
```

- Unix commands have many options/flags - examine them with `man`:

```
$ man grep
```

- Another useful command is `apropos` - try it
- Find files with `find`:

```
$ find ~ | grep lostfile.txt
```

- We can pipe commands and filter results with `|`

```
$ grep energy results.out | sort | uniq
```

Bash: Redirecting output

- Redirect output to a file:

```
$ grep energy results.out | sort | uniq > energies.txt
```

- Append output to a file:

```
$ grep dipole results.out | sort | uniq >> energies.txt
```

- Print contents of a file to screen:

```
$ cat results2.txt
```

- Append contents of a file to another file:

```
$ cat results2.txt >> results_all.txt
```

Bash: Writing shell scripts

```
#!/usr/bin/env bash

# here we loop over all files that end with *.out
for file in *.out; do
    echo $file
    grep energy $file
done
```

- We make the script executable and execute it:

```
# make it executable
$ chmod u+x my_script

# run it
$ ./my_script
```

Bash: Passing arguments to scripts

```
#!/usr/bin/env bash

echo $1
echo $2
echo $3 $2 $1
```

- Arguments are numbered with `$1`, `$2`, `$3`, etc.
- We can now call the script with:

```
$ ./my_script foo bar raboof

foo
bar
raboof bar foo
```

File systems and permissions

File systems at PDC

- AFS (Andrew File System)
  - distributed
  - global
  - backup
- Lustre (Linux cluster file system)
  - distributed
  - high-performance
  - no backup

AFS

- Andrew File System
- Named after the Andrew Project (Carnegie Mellon University)
- Distributed file system
- Homogeneous, location-transparent file name space
- Security and scalability
- Accessible "everywhere" (remember that when you make your files readable/writeable!)
- Access via Kerberos tickets and AFS tokens
- Your PDC home directory is located in AFS, example:

```
/afs/pdc.kth.se/home/u/user
```

- OldFiles mountpoint (created by default) contains a snapshot of the files as they were precisely before the last nightly backup was taken.

```
/afs/pdc.kth.se/home/u/user/OldFiles
```

- By default you get a very limited quota (0.5 GB) but you can ask for more

AFS permissions

- You probably know about Unix file permissions (chown, chmod):

```
-rw-r--r-- 1 user csc-users  3153 Feb 20 11:04 intro_pdc.rst
-rw-r--r-- 1 user csc-users   175 Feb 16 20:50 Makefile
```

- AFS permissions work differently:

```
$ fs listacl
Access list for . is
Normal rights:
  user:remote-users rlidwka
  system:anyuser l
  user rlidwka
```

- Google "AFS ACL" to find out how to change permissions.
- Example: give user "alice" read permissions for the current directory:

```
$ fs setacl . alice read
```

- Always remember that AFS is global.

Lustre

- Parallel distributed file system
- Large-scale cluster computing
- High-performance
- `/cfs/milner`
- `/cfs/zorn`
- Unix permissions
- Not global

Overview: PDC storage

- `/afs`
  - Home
  - Small, quota
  - **Backup**
  - AFS permissions
  - Not good for temporary job files
- `/cfs`
  - Large, no quota (but please be considerate and do not fill up the disk)
  - **No backup**
  - Unix permissions
  - Good for temporary job files
  - Submit jobs from `/cfs`

Programming environment

Working with modules

- At PDC we use the module environment.
- Modules are used to load specific software into your environment.

```
# list all available modules
$ module avail

# show information about a module
$ module show fftw/3.3.0.4

# load a module
$ module load fftw/3.3.0.4

# list currently loaded modules
$ module list

# swap modules
$ module swap PrgEnv-cray PrgEnv-intel

# unload module
$ module unload fftw/3.3.0.4
```

```
$ module list # on Milner

Currently Loaded Modulefiles:
  1) modules/3.2.6.7
  2) nodestat/2.2-1.0501.47138.1.78.ari
  3) sdb/1.0-1.0501.48084.4.48.ari
  4) alps/5.1.1-2.0501.8471.1.1.ari
  5) MySQL/5.0.64-1.0000.7096.23.2
  6) lustre-cray_ari_s/2.4_3.0.80_0.5.1_1.0501.7664.12.1-1.0501.14255.11.3
  7) udreg/2.3.2-1.0501.7914.1.13.ari
  8) ugni/5.0-1.0501.8253.10.22.ari
  9) gni-headers/3.0-1.0501.8317.12.1.ari
 10) dmapp/7.0.1-1.0501.8315.8.4.ari
 11) xpmem/0.1-2.0501.48424.3.3.ari
 12) hss-llm/7.1.0
 13) Base-opts/1.0.2-1.0501.47945.4.2.ari
 14) craype-network-aries
 15) craype/2.04
 16) cce/8.2.3
 17) cray-libsci/12.1.3
 18) pmi/5.0.1-1.0000.9799.94.6.ari
 19) rca/1.0.0-2.0501.48090.7.46.ari
 20) atp/1.7.1
 21) PrgEnv-cray/5.1.29
 22) craype-ivybridge
 23) cray-mpich/6.2.1
 24) slurm
 25) openssh/5.3p1-with-pam-gsskex-20100124
 26) openafs/1.6.6-3.0.80-0.5.1_1.0501.7664-cray_ari_s
 27) heimdal/1.5.2
 28) snic-env/1.0.0
```

- Modules take care of setting proper environment variables.
- The module system does nothing which you could not do by setting environment variables by hand, but you probably want the help of the modules system.

```
$ module show fftw/3.3.0.4
-------------------------------------------------------------------
/opt/cray/modulefiles/fftw/3.3.0.4:

setenv          FFTW_VERSION 3.3.0.4
setenv          CRAY_FFTW_VERSION 3.3.0.4
setenv          FFTW_DIR /opt/fftw/3.3.0.4/sandybridge/lib
setenv          FFTW_INC /opt/fftw/3.3.0.4/sandybridge/include
prepend-path      PATH /opt/fftw/3.3.0.4/sandybridge/bin
prepend-path      MANPATH /opt/fftw/3.3.0.4/share/man
prepend-path      CRAY_LD_LIBRARY_PATH /opt/fftw/3.3.0.4/sandybridge/lib
setenv          PE_FFTW_REQUIRED_PRODUCTS PE_MPICH
prepend-path      PE_PKGCONFIG_PRODUCTS PE_FFTW
setenv          PE_FFTW_TARGET_sandybridge sandybridge
setenv          PE_FFTW_TARGET_x86_64 x86_64
setenv          PE_FFTW_TARGET_interlagos interlagos
...
```

Compiling code on Milner

- Available compiler sets: Cray, GNU, and Intel
- By default, the Cray compiler set is loaded
- We want you to **use the Intel compiler**
- Use `module swap` to switch to Intel

```
# select Intel
$ module swap PrgEnv-cray PrgEnv-intel
```

- Module `cray-libsci` provides BLAS, LAPACK, BLACS, and SCALAPACK
- Module `cray-mpich` provides MPI
- On Cray we compile using compiler wrappers: ftn, cc, and CC

Using compiler wrappers on Milner

- Fortran

```
$ gfortran [flags] source.F90 # wrong
$ ifort    [flags] source.F90 # wrong

# correct
$ ftn [flags] source.F90
```

- C

```
$ gcc [flags] source.c # wrong
$ icc [flags] source.c # wrong

# correct
$ cc [flags] source.c
```

- C++

```
$ g++  [flags] source.cpp # wrong
$ icpc [flags] source.cpp # wrong

# correct
$ CC [flags] source.cpp
```

- Same wrappers for MPI and sequential code

Advice for portability: protect MPI code with the pre-processor

```
program hello

   implicit none

#ifdef ENABLE_MPI
   include 'mpif.h'
#endif

   integer :: irank, num_proc, ierr, tag

#ifdef ENABLE_MPI
   integer :: status(MPI_STATUS_SIZE)

   call MPI_INIT(ierr)
   call MPI_COMM_SIZE(MPI_COMM_WORLD, num_proc, ierr)
   call MPI_COMM_RANK(MPI_COMM_WORLD, irank, ierr)
#else
   irank = 0
   num_proc = 1
#endif

   print *, 'rank ', irank, 'out of', num_proc, 'proc'

#ifdef ENABLE_MPI
   call MPI_FINALIZE(ierr)
#endif

end program
```

```
$ ftn -DENABLE_MPI source.F90
```

Compiling OpenMP code on Milner

- **Intel**

```
$ ftn -openmp source.F90
$ cc  -openmp source.c
$ CC  -openmp source.cpp
```

- Cray

```
$ ftn -h omp source.F90
$ cc  -h omp source.c
$ CC  -h omp source.cpp
```

- GNU

```
$ ftn -fopenmp source.F90
$ cc  -fopenmp source.c
$ CC  -fopenmp source.cpp
```

How about math libraries?

- Cray will automatically link to BLAS, LAPACK, BLACS, and SCALAPACK
- No need to worry about that

Compiling CUDA code on Zorn

- Zorn uses the module system as well

```
# check available modules
$ module avail

# load the CUDA kit
$ module load cuda/5.5

# compile the source
$ nvcc mysource.cc
```

- No compiler wrappers for Fortran/C/C++. Sources are compiled in the "traditional" way.

How to run programs

How to run programs

- After login we are on the login node
- A login node is for submitting jobs, editing files, and compiling small programs
- **We never run calculations interactively on the login node**
- We want balanced load of the resources
- Fair share according to time allocations
- Rather we use a queuing/batch system

- Only persons/groups belonging to a Charge Account Category (CAC; in other words time allocation) can submit
- You belong to the CAC "summer-2014" with 8000 node hours

Queuing systems at PDC

Milner

- Slurm
- Either submit a batch script
- Or book an interactive node just for you

Zorn

- TORQUE/Moab/PBS
- Can only submit a batch script
- Cannot book an interactive node

Slurm (Milner)

```
# submit the job
$ sbatch script.slurm

# see information about all your jobs
$ squeue -u $USER

# remove or stop a job
$ scancel [jobid]
```

PBS (Zorn)

```
# submit the job
$ qsub script.pbs

# see information about all your jobs
$ qstat -u $USER

# remove or stop a job
$ qdel [jobid]
```

Example Slurm job script (Milner)

```
#!/bin/bash --login

# name of the job
#SBATCH -J my_job

# wall-clock time given to this job (20 minutes)
#SBATCH -t 00:20:00

# number of nodes
#SBATCH -N 2

# number of MPI processes per node (the following is actually the default)
#SBATCH --ntasks-per-node=40

# number of MPI processes
#SBATCH -n 80

#SBATCH -o stdout.txt
#SBATCH -e stderr.txt

# run the executable named my_exe
# and write the output to my_output
cd $SLURM_SUBMIT_DIR
aprun -n 80 ./my_exe > my_output 2>&1
```

- In this case the job script is a Bash script but it can be Python or Perl instead

Running with or without hyperthreading

- Hyperthreading is on by default on Milner
- System sees 40 "logical" cores per node but only 20 physical cores are present
- You can turn hyperthreading off
- This is important if you study performance

```
# no hyperthreading
# a node appears as 20 cores
$ aprun -j 1 [other flags] [program]

# hyperthreading enabled (this is the default)
# a node appears as 40 cores
$ aprun -j 2 [other flags] [program]
```

OpenMP usage

- With hyperthreading

```
#!/bin/bash --login

#SBATCH -J my_job
#SBATCH -t 00:20:00
#SBATCH -N 1
#SBATCH -n 40
#SBATCH -o stdout.txt
#SBATCH -e stderr.txt

export OMP_NUM_THREADS=40

cd $SLURM_SUBMIT_DIR
aprun -j 2 -n 1 -N 1 -d $OMP_NUM_THREADS ./my_exe > my_output
```

OpenMP usage

- Without hyperthreading

```
#!/bin/bash --login

#SBATCH -J my_job
#SBATCH -t 00:20:00
#SBATCH -N 1
#SBATCH -n 20
#SBATCH -o stdout.txt
#SBATCH -e stderr.txt

export OMP_NUM_THREADS=20

cd $SLURM_SUBMIT_DIR
aprun -j 1 -n 1 -N 1 -d $OMP_NUM_THREADS ./my_exe > my_output
```

Booking an interactive node (Milner only)

- You can request an interactive node with `salloc`:

```
$ salloc

salloc: Granted job allocation 21223
```

- Then you get a node for interactive use.
- You can log out with `exit`:

```
$ exit

salloc: Relinquishing job allocation 21223
salloc: Job allocation 21223 has been revoked.
```

Example PBS job script (Zorn)

```
#!/bin/bash --login

# name of the job
#PBS -N my_job

# wall-clock time given to this job (20 minutes)
#PBS -l walltime=00:20:00

# number of nodes
#PBS -l nodes=1

#PBS -o stdout.txt
#PBS -e stderr.txt

# run the executable named my_exe
cd $PBS_O_WORKDIR
./my_exe > my_output
```

Important note

Submit all jobs from `/cfs`

```
# submit from here
/cfs/milner/scratch/u/user
```

and **not** from your home directory on `/afs`

```
# not here
/afs/pdc.kth.se/home/u/user
```

Performance analysis

- CrayPAT
- Allinea Performance Reports

Debugging

- DDT
- TotalView
- Cray ATP
- gdb

How to get help

PDC support

- Many questions can be answered by reading the web documentation: https://www.pdc.kth.se/support
- **Preferably contact PDC support by email**: support@pdc.kth.se
- Or by phone: +46 (0)8 790 7800
- You can also make an appointment to come and visit.
- Your email support request will be tracked - you get a ticket number.
- For follow-ups/replies always include the ticket number - they look like this: `[SNIC support #12345]`

How to report problems

- Do not report new problems by replying to old/unrelated tickets.
- Split unrelated problems into separate email requests.
- Use a descriptive subject in your email (unhelpful subject line: "problem").
- Give your PDC user name.
- Be as specific as possible.
- For problems with scripts/jobs, give an example. Either send the example or make it accessible to PDC support.
- Make the problem example as small/short as possible.
- Provide all necessary information to reproduce the problem.
- If you want the PDC support to inspect some files, make sure that the files are readable.
- Do not assume that PDC support personnel have admin rights to see all your files or change permissions.

That's all folks - enjoy the summer school!

Slideshow created using remark.