# Introduction to PDC environement

Cristian Cira

PDC Center for High Performance Computing
KTH Royal Institute of Technology

PDC April 2016

# Outline

# SNIC

## Swedish National Infrastructure for Computing


National research infrastructure that provides a balanced and cost-efficient set of resources and user support for large scale computation and data storage to meet the needs of researchers from all scientific disciplines and from all over Sweden (universities, university colleges, research institutes, etc).

# Access to EU Facilities and Experts







# PDC and Industry

Working with industrial researchers and developers on major international projects that push high-performance computing to the next level.

Recently established a business development unit that provides consultancy and HPC services to industries.





# Broad Range of Training

**Summer School** Introduction to HPC held every year

**Specific Courses** Programming with GPGPU, Recent Advances in Distributed and Parallel Computing and/or Cloud Computing, Software Development Tools, etc

**PDC User Days** PDC Open House and Pub Afternoon

# First-Line Support and System Staff

## First-line support

Provides specific assistance to PDC users
Answers general questions from the public

## System staff

System managers/administrators ensure that computing and storage resoruces run smoothly and securely

# Application Experts

Hold PhD degrees in different scientific fields and are experts in HPC. Together with researchers, they optimize, scale and enhance scientific codes for the next generation supercomputers.

**Jaime Rosal Sandberg**
Computational Chemistry

**Cristian Cira**
Perfomance Analysis

**Henric Zazzi**
Bioinformatics/Genetics

**Michael Djurfeldt**
Computational Physics

**Jing Gong**
Scientific Computing

**Thor Wikfeldt**
Computational Chemistry

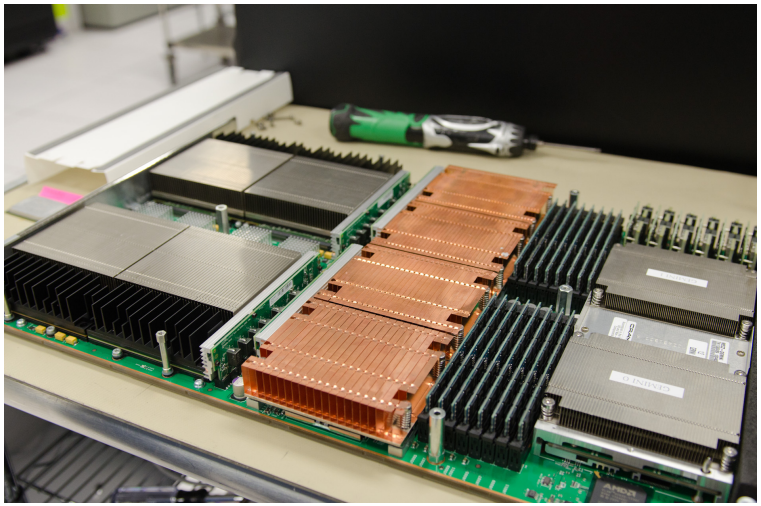
# Services

- Access to supercomputers
- HPC training
- Postgraduate degree projects
- Visualization
- Support
- Expertise in HPC software
- Access to international HPC facilities
- Data storage

# Outline



# What is a cluster?

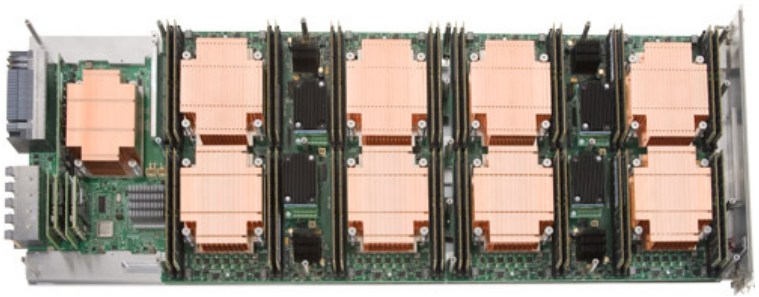



- Cluster
- Racks
- Blades
- Nodes
- Processors
- Cores
- Login nodes
- Compute nodes
- Dedicated nodes
- Transfer nodes
- Service nodes

# Beskow - Cray XC40 system


## Fastest machine in Scandinavia

- in production until Q4 2018
- 9 racks 1676 nodes
- Intel Xeon Processor E5-2698 v3 40M Cache, 2.30 GHz
- 53.632 cores - 32 cores/node
- Aries Dragonfly network topology
- 104.7 TB memory - 64 GB/node



# Tegner

## pre/post processing for Beskow

**5 x 2TB Fat nodes**

4 x 12 core Ivy Bridge
2TB RAM
2 x Nvidia Quadro K420

**5 x 1TB Fat nodes**

4x 12 core Ivy Bridge
1TB RAM
2 x Nvidia Quadro K420

**55 Thin Nodes**

2 x 12 core Haswell
512GB RAM
Nvidia Quadro K420 GPU

- Used for pre/post processing data
- Only for academia within the Stockholm area
- Has large RAM nodes
- Has nodes with GPUs
- Lifetime: Q4 2018

# Summary of PDC resources

| | Beskow | Tegner |
|---|---|---|
| Core/node | 32 | 48/24 |
| Nodes | 1.676 | 50 x 24 Haswell/GPU<br>10 x 48 Ivy bridge |
| RAM (GB) | 1.676 x 64 | 50 x 512<br>5 x 1000<br>5 x 2000 |
| Allocations (core hours)<br>Small<br>Medium<br>Large | <br>$< 5.000$<br>$< 200.000$<br>$\geq 200.000$ | <br>$< 5.000$<br>$< 80.000$ |
| Allocation via SNIC | yes | yes |
| AFS | login node only | yes |
| Lustre | yes | yes |

# File Systems

## Andrew File System (AFS)

- Distributed file system accessible to any running AFS client
- Home directory
  `/afs/pdc.kth.se/home/[initial]/[username]`
- Access via Kerberos tickets and AFS tokens
- Not accessible to compute nodes on Beskow

## Lustre File System (Klemming)

- Open-source massively parallel distributed file system
- Very high performance (5PB storage - 140GB/s bandwidth)
- NO backup (always move data when done) NO personal quota
- Home directory
  `/cfs/klemming/nobackup/[initial]/[username]`

# Outline



# Access requirements

User account either SUPR or PDC

Time allocation set the access limits

## Apply for SUPR account

- http://supr.snic.se
- SNIC database of persons, projects, project proposals and more
- Apply and link SUPR account to PDC

## Apply for PDC account

- http://www.pdc.kth.se/support/accounts/user
- Electronic copy of your passport
- Valid post address for password
- Membership of specific time allocation

# Time Allocations

## Small allocation

- Applicant can be a PhD student or higher
- Evaluated on a technical level only
- Limits is usually $5K$ corehours/month

## Medium allocation

- Applicant must be a senior scientist in Swedish academia
- Evaluated on a technical level only
- On large clusters: $200K$ corehours/month

## Large allocation

- Applicant must be a senior scientist in Swedish academia
- Need evidence of successful work at a medium level
- Evaluated on a technical and scientific level
- Proposal evaluated by SNAC twice a year

# Using resources

- All resources are free of charge for Swedish academia
- Acknowledgement are taken into consideration when applying
- Please acknowledge SNIC/PDC when using these resources:

## Acknowledge SNIC/PDC

The computations/simulations/[SIMILAR] were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at [CENTERNAME (CENTER-ACRONYM)]

## Acknowledge people

NN at [CENTER-ACRONYME] is acknowledged for assistance concerning technical and implementation aspects [OR SIMILAR] in making the code run on the [OR SIMILAR] [CENTER-ACRONYM] resources.

# Authentication

## **Kerberos** Authentication Protocol

### Ticket

- Proof of users identity
- Users use passwords to obtain tickets
- Tickets are cached on the user's computer for a specified duration
- Tickets should be created on your local computer
- No passwords are required during the ticket's lifetime

### Realm

Sets boundaries within which an authentication server has authority `NADA.KTH.SE`

### Principal

Refers to the entries in the authentication server database `username@NADA.KTH.SE`

# Kerberos commands

| | |
|---|---|
| kinit | proves identity |
| klist | lists kerberos tickets |
| kdestroy | destroys ticket file |
| kpasswd | changes password |

```
$ kinit --forwardable username@NADA.KTH.SE
$ klist -Tf

Credentials cache : FILE:/tmp/krb5cc_500
    Principal: username@NADA.KTH.SE
Issued       Expires     Flags  Principal
Mar 25 09:45 Mar 25 19:45 FI krbtgt/NADA.KTH.SE@NADA.KTH.SE
Mar 25 09:45 Mar 25 19:45 FA afs/pdc.kth.se@NADA.KTH.SE
```

# Login using Kerberos tickets

## Get a 7 days forwardable ticket on your local system

```
$ kinit -f -l 7d username@NADA.KTH.SE
```

## Forward your ticket via ssh and login

```
$ ssh
  -o GSSAPIDelegateCredential
  -o GSSAPIAuthentication
  -o GSSAPIKeyExchange
  username@clustername.pdc.kth.se
```

## OR, when using ~/.ssh/config

```
$ ssh username@clustername.pdc.kth.se
```

Always create a kerberos ticket on your local system

# Outline

# Compiling, Linking and Running Applications

## on HPC clusters

| | |
|---|---|
| source code | C / C++ / Fortran ( `.c`, `.cpp`, `.f90`, `.h` ) |
| compiled | Cray/Intel/GNU compilers<br>include headers, expand macros (`.i`,`.ii` ) |
| assembled | into machine code (`.o`, `.obj` ) |
| linked | Static Libraries (`.lib`, `.a` )<br>Shared Library (`.dll`, `.so` ) |

| | |
|---|---|
| request allocation | submit job request to SLURM queuing system<br>`salloc/sbatch` |
| run | application on scheduled resources<br>`aprun/mpirun` |

# Compiling serial and/or parallel code

specific to Tegner

## GNU Compiler Collection (**gcc**)

```
$ module load  gcc openmpi
$ gcc    -fopenmp source.c
$ g++    -fopenmp source.cpp
$ gfortran -fopenmp source.F90
$ mpicc    -fopenmp source.c
$ mpicxx   -fopenmp source.cpp
$ mpif90   -fopenmp source.F90
```

## Portland Group Compilers (**pgi**)

```
$ module load  pgi
$ pgcc  -mp source.c
$ pgcpp -mp source.cpp
$ pgf90 -mp source.F90
```

## Intel compilers (**i-compilers**)

```
$ module load i-compilers
$ icc   -openmp source.c
$ icpc  -openmp source.cpp
$ ifort -openmp source.F90

$ module add i-compilers intelmpi
$ mpiicc  -openmp source.c
$ mpiicpcp  -openmp source.cpp
$ mpiifort  -openmp source.F90
```

# Modules

The *modules package* allow for dynamic add/remove of installed software packages to the running environment

## Loading modules

```
module load  <software_name>
module add  <software_name>
module use  <software_name>
```

## Unloading modules

```
module unload <software_name>
```

# Modules

## Displaying modules

### $ module list

```
Currently Loaded Modulefiles:
  1) modules/3.2.6.7
  ...
  20) PrgEnv-cray/5.2.56
```

### $ module avail [*software_name*]

```
--------------------------- /opt/modulefiles ----------------------------
gcc/4.8.1     gcc/4.9.1(default)     gcc/4.9.2     gcc/4.9.3     gcc/5.1.0
```

### $ module show *software_name*

```
------------------------ /opt/modulefiles/gcc/4.9.1 --------------------
conflict  gcc
prepend-path  PATH /opt/gcc/4.9.1/bin
prepend-path  MANPATH /opt/gcc/4.9.1/snos/share/man
prepend-path  LD_LIBRARY_PATH /opt/gcc/4.9.1/snos/lib64
setenv  GCC_PATH /opt/gcc/4.9.1
-------------------------------------------------------------------
```

# Programming Environment Modules

specific to Beskow only

| | | |
|---|---|---|
| Cray | `$ module load PrgEnv-cray` | `$ cc source.c` |
| Intel | `$ module load PrgEnv-intel` | `$ CC source.cpp` |
| GNU | `$ module load PrgEnv-gnu` | `$ ftn source.F90` |

## Compiler wrappers : **cc CC ftn**

### Advantages

Compiler wrappers will automatically

- link to BLAS, LAPACK, BLACS, SCALAPACK, FFTW
- use MPI wrappers

### Disadvantage

Sometimes you need to edit Makefiles which are not designed for Cray

# Outline

# How to run programs

- After login we are on a *login node* used only for:
  - submitting jobs,
  - editing files,
  - compiling small programs,
  - other computationally light tasks.
- Never run calculations interactively on the login node
- We want balanced load of the resources
- Fair share according to time allocations
- Only persons/groups belonging to a time allocation can run

- We use a queuing/batch system

# SLURM queueing system

## Simple Linux Utility for Resource Management

- Open source, fault-tolerant, and highly scalable cluster management and job scheduling system
  - Allocates exclusive and/or non-exclusive access to resources for some duration of time
  - Provides a framework for starting, executing, and monitoring work on the set of allocated nodes
  - Arbitrates contention for resources by managing a queue
- Job Prority computed based on

| | |
|---|---|
| Age | the length of time a job has been waiting |
| Fair-share | the difference between the portion of the computing resource that has been promised and the amount of resources that has been consumed |
| Job size | the number of nodes or CPUs a job is allocated |
| Partition | a factor associated with each node partition |
| QOS | a factor associated with each Quality Of Service |

# Interactive session — salloc

## Request for an interactive allocation or resources

```
$ salloc -A <account> -t <d-hh:mm:ss> -N <nodes>
salloc: Granted job allocation 123456
```

## Run application on Beskow

```
$ aprun -n <PEs> -d <depth> ./binary.x
#PEs   - number of processing elements
#depth  - number of threads (depth) per PE
```

## Run application on Tegner

```
$ mpirun -np <cores> ./binary.x
```

# Launch jobs in the backgound **sbatch**

## Submit the job to SLURM queue

```
$ sbatch <script>
Submitted batch job 958287
```

The script should contain all necessary data to identify the account and requested resources

## Example of request to run myexe for 1 hour on 4 nodes

```
# set !/bin/bash -l

#SBATCH -A 201X-X-XX
#SBATCH -J myjob
#SBATCH -t 1:00:00
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=32
#SBATCH -e error_file.e
#SBATCH -o output_file.o

aprun -n 128 ./myexe > my_output_file 2>&1
```

# Monintoring and/or canceling running jobs

## **squeue** -u $USER

Displays all queue and/or running jobs that belong to the user

```
cira@beskow-login2:~> squeue -u cira
 JOBID     USER ACCOUNT           NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES  CPUS
957519   cira pdc.staff      VASP-test   R None      2016-08-15T08:15:24    6:09:42   17:49:18    16  1024
957757   cira pdc.staff      VASP-run    R None      2016-08-15T11:14:20    3:10:46   20:48:14   128 8192
```

## **scancel** [job]

Stops a runing job or removes a pending one from the queue

```
cira@beskow-login2:~> scancel 957519
salloc: Job allocation 957891 has been revoked.

cira@beskow-login2:~> squeue -u cira
JOBID     USER ACCOUNT           NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES  CPUS
957757   cira pdc.staff      VASP-run    R None      2016-08-15T11:14:20    3:10:46   20:48:14   128 8192
```

# Outline



# How to start your project

- Proposal for a small allocation
- Develop and test your code
- Run and evaluate scaling
- Proposal for a medium (large) allocation

# 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
  - you get a ticket number.
  - always include the ticket number in follow-ups/replies they look like this: [SNIC support #12345]
- Or by phone: +46 (0)8 790 7800
- You can also make an appointment to come and visit.

# How to report problems support@pdc.kth.se

- 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.
- 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.

# Questions...?

# Introducing the unix shell

## login into Beskow

```
$ ssh beskow.pdc.kth.se
Last login: Fri Feb 13 20:20:06 2016 from example.com
bast@beskow-login2:~$ _
```

- Command Line Interface often more efficient than GUI
- High action-to-keystroke ratio
- 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 work-flows

# Bash: Files and directories

## pwd command returns current directory

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

## Change the directory with cd

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

## 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
```

# Bash: Creating and editing files and directories

## Command **mkdir** creates a new directory

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

## File editors

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

# Bash: Copying, moving, renaming, and deleting

```
$ cp draft.txt backup.txt
$ cp -r results backup
$ mv draft.txt draft_2.txt
$ mv results backup
$ mv results ..
$ rm draft.txt
$ rm -r results
```

```
# copy file
# recursively copy directory
# move/rename file
# move/rename directory
# move directory one level up
# remove file
# remove directory and contents
```

# Bash: Finding things

## Extract lines which contain an expression with **grep**

```
$ grep fixme draft.txt
$ man grep
$ grep energy results.out | sort | uniq
```

## Redirecting output

```
$ grep energy results.out | sort | uniq > energies.txt
$ grep dipole results.out | sort | uniq >> energies.txt
$ cat results2.txt
$ cat results2.txt >> results_all.txt
```

# Bash: Writing shell scripts

## Edit script in preferred editor

```
#!/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
```

## Change permissions **chmod** to make the script executable

```
# make it executable
$ chmod u+x my_script
# run it
$ ./my_script
```