Introduction to PDC environment

Xin Li

PDC Center for High Performance Computing
KTH Royal Institute of Technology

PDC Summer School August 2018

Outline

History of PDC

Year	rank	procs.	peak TFlops	vendor	name
2014	32	53632	1973.7	Cray	Beskow[1]
2011	31	36384	305.63	Cray	Lindgren[2]
2010	76	11016	92.534	Cray	Lindgren[3]
2010	89	9800	86.024	Dell	Ekman[4]
2005	65	886	5.6704	Dell	Lenngren[5]
2003	196	180	0.6480	HP	Lucidor[6]
1998	60	146	0.0934	IBM	Strindberg[7]
1996	64	96	0.0172	IBM	Strindberg[8]
1994	341	256	0.0025	Thinking Machines	Bellman[9]

[1] XC40 16-core 2.3GHz
[2] XE6 12-core 2.1 GHz
[3] XT6m 12-core 2.1 GHz
[4] PowerEdge SC1435 Dual core Opteron 2.2GHz, Infiniband
[5] PowerEdge 1850 3.2 GHz, Infiniband
[6] Cluster Platform 6000 rx2600 Itanium2 900 MHz Cluster, Myrinet
[7] SP P2SC 160 MHz
[8] SP2/96
[9] CM-200/8k

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).

Broad Range of Training

Summer School Introduction to HPC held every year

Specific Courses Programming with GPGPU, Distributed and Parallel Computing and/or Cloud Computing, Software Development Tools, CodeRefinery workshops, etc

PDC User Days PDC Pub and Open House

Support and System Staff

First-line support

Provide specific assistance to PDC users related to accounts, login, allocations etc.

System staff

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

Application Experts

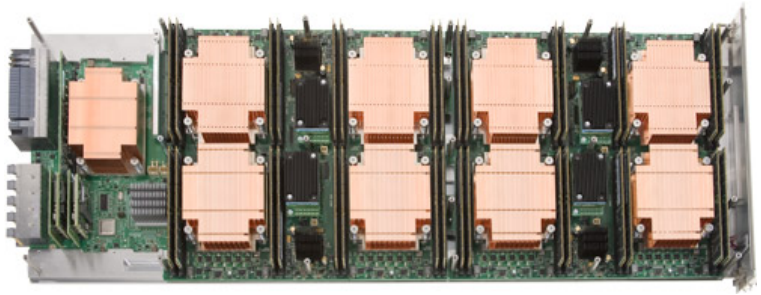
Hold PhD degrees in various fields and specialize in HPC. Assist researchers in optimizing, scaling and enhancing scientific codes for current and next generation supercomputers.

Outline

Beskow - Cray XC40 system

Fastest machine in Scandinavia

- Lifetime: Q4 2019
- 11 racks, 2060 nodes
- Intel Haswell processor 2.3 GHz
  Intel Broadwell processor 2.1 GHz
- 67,456 cores - 32(36) cores/node
- Aries Dragonfly network topology
- 156.4 TB memory - 64(128) GB/node



1 XC compute blade
1 Aries Network Chip (4 NICs)
4 Dual-socket Xeon nodes
4 Memory DIMM / Xeon 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

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

46 Thin Nodes

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

9 K80 Nodes

2 x 12 core Haswell, 512GB RAM
Nvidia Tesla K80 GPU

- Used for pre/post processing data
- Has large RAM nodes
- Has nodes with GPUs
- Has two transfer nodes
- Lifetime: Q4 2019

Summary of PDC resources

	Beskow	Tegner
Cores in each node	32/36	48/24
Nodes	1676 Haswell 384 Broadwell	55 x 24 Haswell/GPU 10 x 48 Ivy bridge
RAM (GB)	1676 x 64GB 384 x 128GB	55 x 512GB 5 x 1TB 5 x 2TB
Allocations (core hours per month) Small Medium Large	 $< 5k$ $< 200k$ $\geq 200k$	 $< 5k$ $< 80k$
Availability via SNIC	yes	with Beskow
AFS Lustre	login node only yes	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 - 130GB/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 PDC account via SUPR

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

Apply for PDC account via PDC

- https://www.pdc.kth.se/support → "Getting Access"
- 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 more senior
- Evaluated on a technical level only
- Limits is usually $5K$ corehours each month

Medium allocation

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

Normal commands:	On KTH-Ubuntu machines:
kinit generates ticket	pdc-kinit
klist lists kerberos tickets	pdc-klist
kdestroy destroys ticket file	pdc-kdestroy
kpasswd changes password	pdc-kpasswd

```
$ 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=yes
  -o GSSAPIAuthentication=yes
  -o GSSAPIKeyExchange=yes
  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
https://www.pdc.kth.se/support/documents/login/login.html

File transfer

Scp/Rsync: copy files between hosts on a network

AFS client: drag-and-drop or use a cp command

Using scp

- scp localFile user@t04n28.pdc.kth.se:/afs/pdc.kth.se/home/u/user
- scp -r localDir user@t04n28.pdc.kth.se:/afs/pdc.kth.se/home/u/user
- scp user@t04n28.pdc.kth.se:/cfs/klemming/scratch/u/user/pdcFile .

Using AFS client

- AFS client can be installed on Linux, Windows, and MacOS
- Linux: start with "sudo /etc/init.d/openafs-client start"
- MacOS: start with "aklog"

Note: You cannot access /cfs/klemming files via AFS client.

Outline

Compiling, Linking and Running Applications

on HPC clusters

source code C / C++ / Fortran (`.c`, `.cpp`, `.f90`, `.h`)

compile Cray/Intel/GNU compilers

assemble into machine code (object files: `.o`, `.obj`)

link Static Libraries (`.lib`, `.a`)
Shared Library (`.dll`, `.so`)
Executables (`.exe`, `.x`)

request allocation submit job request to SLURM queuing system
`salloc/sbatch`

run application on scheduled resources
`aprun/mpirun`

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>
```

Swapping modules

```
module swap  <software_name_1> <software_name_2>
```

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

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

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
```

Intel compilers (**i-compilers**)

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

Portland Group Compilers (**pgi**)

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

CUDA compilers (**cuda**)

```
$ module load cuda
$ nvcc source.cu
$ nvcc -arch=sm_37  source.cu
```

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
- Instead, request compute resources *interactively* or via *batch script*

- All jobs must be connected to a time allocation
- For courses, PDC sets up a *reservation* for resources

- To manage the workload on the clusters, PDC uses a queueing/batch system

SLURM workload manager

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 Priority 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

Interactive session — salloc

Request an interactive allocation of 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> -N <PEs_per_node> ./binary.x
#PEs  - number of processing elements
#depth  - number of threads (depth) per PE
#PEs_per_node  - PEs per node
```

Run application on Tegner

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

Launch batch jobs — 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

```
#!/bin/bash -l

#SBATCH -A summer-2017
#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
```

Monitoring and/or cancelling 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 running 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...?

Hands-on exercise

Login

- Some configuration steps are needed to log in to PDC
- Depends on OS: https://www.pdc.kth.se/support/documents/login/login.html
- In short, Kerberos and SSH supporting GSSAPI key exchange must be installed
- If needed, you will receive help to connect from your own laptops

Exercises

- You will now practice key steps in using PDC resources
- The example source code and batch scripts can be found at `/afs/pdc.kth.se/home/k/kthw/Public/pdc_test.tar.gz`, or in the GitHub repository `https://github.com/PDC-support/introduction-to-pdc` (in the intro-course branch)

Hands-on exercise

Exercises

The exercises below will demonstrate:

1. Kerberos - creating and listing tickets
2. Login - via command line or PuTTY, and where you end up on PDC
3. AFS home, information on your projects, `/cfs/klemming`
4. Working with modules
5. Compiling code
6. Submitting jobs
7. Running interactively

Further information

- https://www.pdc.kth.se/support
- https://hpc-carpentry.github.io/hpc-intro/
- https://software-carpentry.org/lessons/

Kerberos

- Two files should be created/edited to enable login to PDC, `krb5.conf` and (linux and mac) `.ssh/config`. For Windows, SSH configuration is done in PuTTY. Further information here: https://www.pdc.kth.se/support/documents/login/configuration.html
- Assuming that you have now configured Kerberos and SSH, create a forwardable Kerberos ticket
- List the ticket. Can you see if it is forwardable? What does all the jargon mean?

Kerberos

Answer

- To create a forwardable Kerberos ticket, type:

```
$ kinit -f <username>@NADA.KTH.SE
```

- To list your Kerberos tickets, type:

```
$ klist -f
```

The output will hopefully look similar to

```
Credentials cache: FILE:/tmp/krb5cc_H26527
    Principal: kthw@NADA.KTH.SE
Issued                Expires               Flags    Principal
Aug  3 16:41:51 2017  Aug  4 16:39:50 2017  FfA      krbtgt/NADA.KTH.SE@
Aug  3 16:41:52 2017  Aug  4 16:39:50 2017  fA       afs/pdc.kth.se@NADA
```

The first line is your Kerberos ticket, the second line your AFS token.

Login

- Start by logging in to Beskow
- Where are you after login?
  - What is your current directory?
  - What's the name of the login node?
- Have a look at the currently running processes on the login node

Login

Answer

- On Linux/MacOS, type one of these (use PuTTY on Windows):

```
$ ssh username@beskow.pdc.kth.se
$ ssh -o GSSAPIDelegateCredentials=yes -o GSSAPIKeyExchange=yes
  -o GSSAPIAuthentication=yes username@beskow.pdc.kth.se
```

- After logging in, `pwd` shows your current directory:

```
$ pwd
/afs/pdc.kth.se/home/u/username
```

- The `hostname` command shows the hostname of the login node:

```
$ hostname
beskow-login2.pdc.kth.se
```

- The `top` command shows a snapshot of currently running processes:

```
$ top
```

AFS home, listing projects, /cfs/klemming

- Check your AFS disk quota (hint: you will need the `fs` command, type `fs help` to see available subcommands)
- Go to your klemming nobackup directory. Can you run `fs` there?
- Check which allocation(s) you belong to using the `projinfo` command
  - Check all options of `projinfo` using the `-h` flag
  - How much have your allocations been used, and for how long are they valid?

AFS home, listing projects, /cfs/klemming

Answer

- The `fs lq` command shows the name of the AFS volume of your home directory, your disk quota and usage:

```
$ fs lq
```

- `fs lq` only works on AFS. You can type `df -h .` instead, but there's no quota on your klemming usage:

```
$ df -h .
Filesystem      Size  Used Avail Use% Mounted on
...:/klemming  5.2P  4.4P  730T  86% /cfs/klemming
```

- The `projinfo` command accesses a database that contains logs of all allocations

```
$ projinfo
```

Working with modules

- List the modules that are currently loaded (hint: type `module help`)
- List all the modules that are available on Beskow
- List all available modules named `PrgEnv`
- Swap from the `PrgEnv-cray` to the `PrgEnv-gnu` module

Working with modules

Answer

- Listing all loaded modules and all available modules is done like this:

```
$ module list
$ module avail
```

- To list all modules matching a pattern (like PrgEnv), type

```
$ module avail PrgEnv
```

- To swap programming environments (i.e. compiler environments), type

```
$ module swap PrgEnv-cray PrgEnv-gnu
```

  After swapping these modules, the compiler wrappers `CC`, `cc` and `ftn` point to the GNU compilers (instead of the Cray compilers)

Compiling code

- Copy the tarball `/afs/pdc.kth.se/home/k/kthw/Public/pdc_test.tar.gz` to your nobackup directory, and unpack it
- Now compile the MPI example code `hello_world_mpi.c`
  - Do you need to load any MPI libraries?
  - What compiler will be used when using the `cc` compiler wrapper?

Compiling code

Answer

- Copying and extracting the tarball to your klemming directory:

```
$ cd /cfs/klemming/nobackup/u/username # replace this!
$ cp /afs/pdc.kth.se/home/k/kthw/Public/pdc_test.tar.gz .
$ tar zxf pdc_test.tar.gz
$ cd pdc_test
```

- When using the compiler wrappers CC, cc and ftn, no MPI or numerical libraries need to be loaded or linked to explicitly!

```
$ cc -o hello_world_mpi hello_world_mpi.c
```

- Which compiler did we just compile with?

```
$ cc --version
gcc (GCC) 4.9.1 20140716 (Cray Inc.)
```

Submitting jobs

- Open the batch script `sbatch_beskow.sh` in your favorite editor (emacs or vim)
  - Set the allocation ID to `edu18.intropdc`
  - Set that the job should run on two nodes, with 16 processes running on each node
  - Set the requested time of the job to 2 minutes
- Submit the job to the SLURM queue!
- Monitor the queue to see if your job is running
- What output did you get?

Submitting jobs

Answer

- The allocation, number of nodes and time are set with these flags

```
#SBATCH -A edu18.intropdc
#SBATCH -t 0:02:00
#SBATCH --nodes=2
```

- If you want to run 32 MPI processes in total, with 16 processes on each node, both the `-n` and `-N` flags to `aprun` must be used:

```
aprun -n 32 -N 16 ./hello_world_mpi
```

- The job is submitted and monitored like this:

```
$ sbatch sbatch_beskow.sh
$ squeue -u <username>
```

- The output gets written to a default filename:

```
$ cat slurm-2559495.out
```

Running interactively

- For this example, use the OpenMP code `hello_world_openmp.c`.
  - First compile it. Does it work to compile with `cc -o hello_world_openmp hello_world_openmp.c`?
  - Spoiler: it won't work, since you need to add an OpenMP compiler flag. Have a look in the Makefile to find the flag
- Now allocate an interactive node. Hint: you need to specify allocation, number of nodes and time.
- Run the compiled `hello_world_openmp` on the interactive node. Important: don't forget to use `aprun`
- How many OpenMP threads do you see? How do you set the number of threads?

Running interactively

Answer

- To compile the code with OpenMP support, do:

```
$ cc -fopenmp -o hello_world_openmp hello_world_openmp.c
```

- To allocate an interactive node for 10 minutes:

```
$ salloc -A edu18.intropdc -t 0:05:0 -N 1
```

- To run the code using 32 threads, first set this environment variable and then run with `aprun`

```
$ export OMP_NUM_THREADS=32
$ aprun -n 1 -d 32 ./hello_world_openmp
# the -d (depth) isn't actually needed here
```

NOTE: `aprun` sends the job to the interactive compute node. Omitting `aprun` means the calculation will run on the login node.

Questions...?