Quick start of Titan for INF 3380

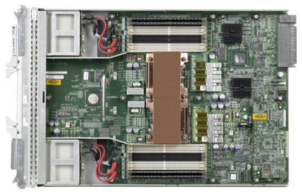
Overview (1)

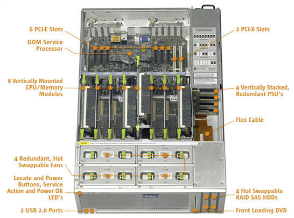
Titan is the computing facilities at UiO hosted by USIT by the VD (Research Computing Services) group. Titan is a powerful computing cluster composed of 650+ computers having over 5000 cores (CPUs). All nodes in the Titan cluster run Linux Operating system [1].

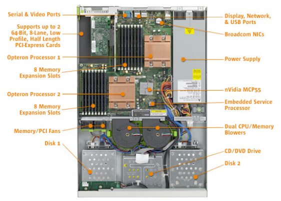
In this course, the Titan will be used for parallel programming exercises and projects using MPI on a certain number of nodes and OpenMP on one node. The whole system is shared by users simultaneously.

Titan

Overview (2)



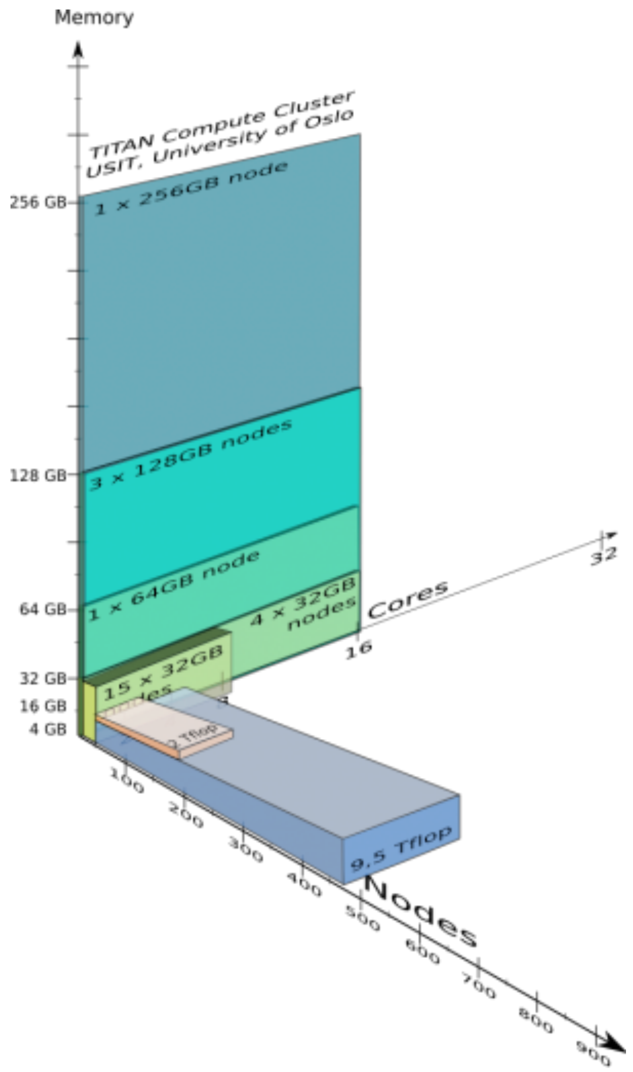

6 PCI-E Slots
ILOM Service Processor
8 Vertically Mounted CPU/Memory Modules
4 Redundant, Hot Swappable Fans
Locate and Power Buttons, Service Action and Power OK LED's
2 USB 2.0 Ports
2 PCI-X Slots
4 Vertically Stacked, Redundant PSU's
Flex Cable
4 Hot Swappable RAID SAS HDDs
Front Loading DVD



DELL M610 blade computing nodes | **HP BL2×220c blade computing nodes** | **SUN x6220 blade computing nodes** | **SUN x4600 4u computing nodes** | **SUN x2200 1u computing nodes**

Hardware Information	
Number of Cores	5004
Number of nodes	651
Max Floating point performance, double	40 Tflops/s
Total memory	11 TeraBytes
Total Interconnect Bandwidth	647 GigaBytes/s
Total local storage	271 TeraBytes
Total Bandwidth to local storage	39 GigaBytes/s
Software Information	
Nodes in titan run Linux, 64 bit Centos 5. GPFS parallel file system, OFED stack for Infiniband.	

Cores and memory



TITAN computing nodes (February 2008) [1]

Cores

Most of nodes of Titan have 8 cores while there are a very few nodes having 32 cores (8 quad-core processors) and huge memory between 64GB and 256 GB

Memory	Num nodes
8 GB	70
16 GB	530
24 GB	16
32 GB	36
64GB	2
128 GB	2
256 GB	1

The more memory you ask for the fewer available nodes you have.

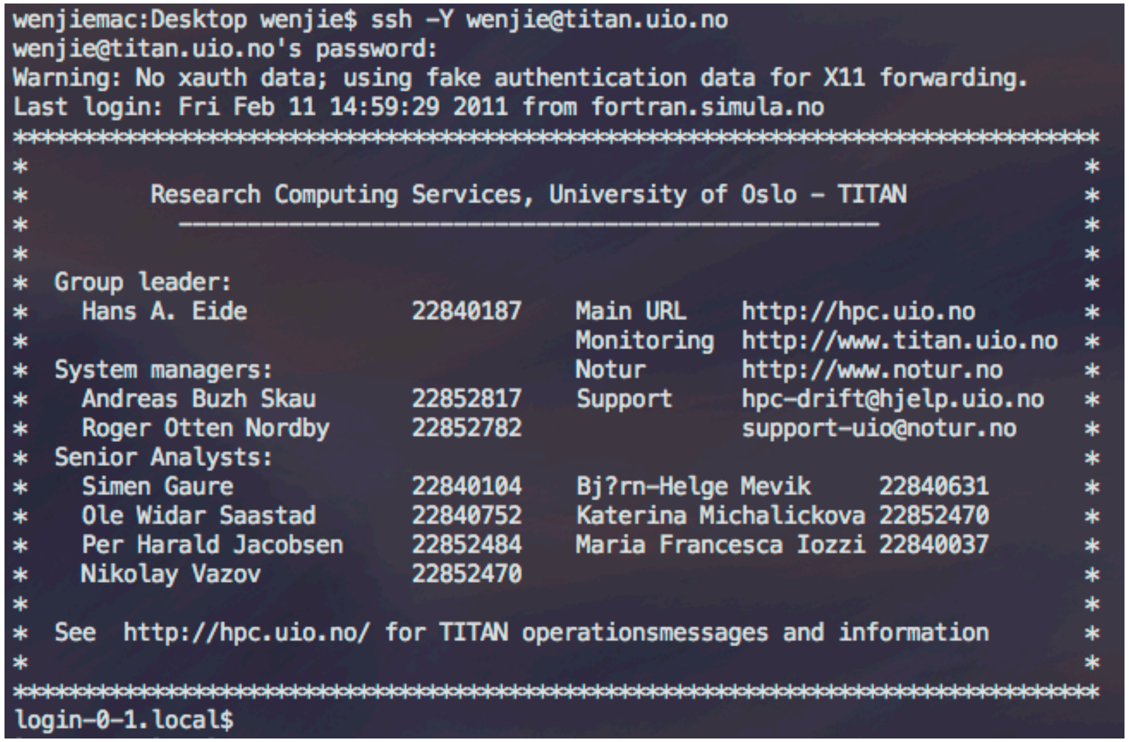
Unix systems

Use secure shell to connect to Titan:

```
ssh -Y titan.uio.no
```

To copy file, use secure copy command:

```
scp myfile.txt myusername@titan.uio.no:~
```



```
wenjiemac:Desktop wenjie$ ssh -Y wenjie@titan.uio.no
wenjie@titan.uio.no's password:
Warning: No xauth data; using fake authentication data for X11 forwarding.
Last login: Fri Feb 11 14:59:29 2011 from fortran.simula.no
******************************************************************************
*                                                                            *
*         Research Computing Services, University of Oslo - TITAN           *
*           --------------------------------------------------               *
*                                                                            *
*  Group leader:                                                             *
*    Hans A. Eide           22840187    Main URL     http://hpc.uio.no       *
*                                       Monitoring   http://www.titan.uio.no *
*  System managers:                     Notur        http://www.notur.no     *
*    Andreas Buzh Skau      22852817    Support      hpc-drift@hjelp.uio.no  *
*    Roger Otten Nordby     22852782                 support-uio@notur.no    *
*  Senior Analysts:                                                          *
*    Simen Gaure            22840104    Bj?rn-Helge Mevik      22840631      *
*    Ole Widar Saastad      22840752    Katerina Michalickova 22852470       *
*    Per Harald Jacobsen    22852484    Maria Francesca Iozzi 22840037       *
*    Nikolay Vazov          22852470                                         *
*                                                                            *
*  See  http://hpc.uio.no/ for TITAN operationsmessages and information      *
*                                                                            *
******************************************************************************
login-0-1.local$
```

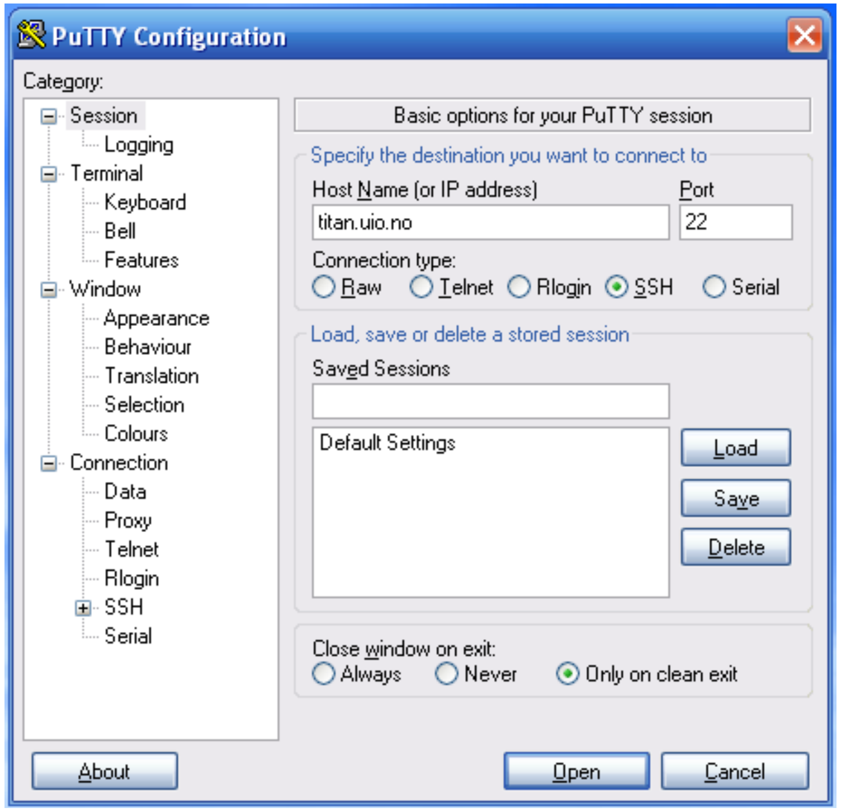
Accessing (2)

Windows systems

1. X emulators for Windows, such as cygwin.

Details are available at http://www.cygwin.com/

2. Remote terminal, which supports the ssh protocol (secure shell), such as putty.



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): titan.uio.no

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

When using putty, an application with GUI cannot be forwarded from Titan to your Windows system successfully if you don't have an X11 server installed.

Popular X11 servers:

Hummingbird Exceed X Server

Xming
http://www.straightrunning.com/XmingNotes/

Titan

After you login to Titan, you will have a connection to a special node called front-end node, where you can compile and edit source code and submit your job.

The front-end node is shared by a certain number of users. You should **NOT** run your computing code there. Heavy load on a front-end node will lead to a slow response to all users logining there.

Your code should be posted to compute nodes, i.e. back-end nodes, through a batch system.

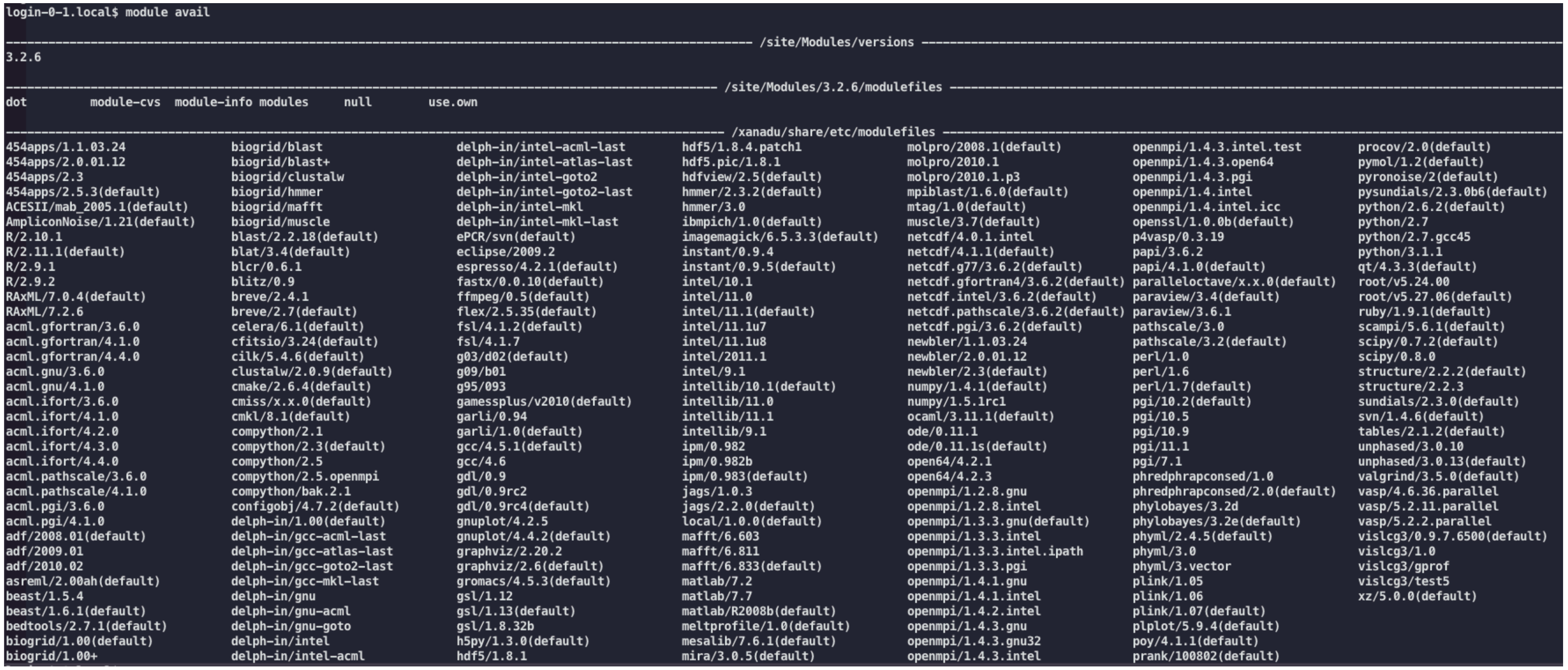
Environment Modules package (1)

The **Environment Modules package** provides for the dynamic modification of a user's environment via modulefiles [2], which has already been installed on Titan.

With the aid of this package, one can switch environments of compilers and precompiled packages easily and fast.

1. Showing how many modules can be used.

```
module avail
```



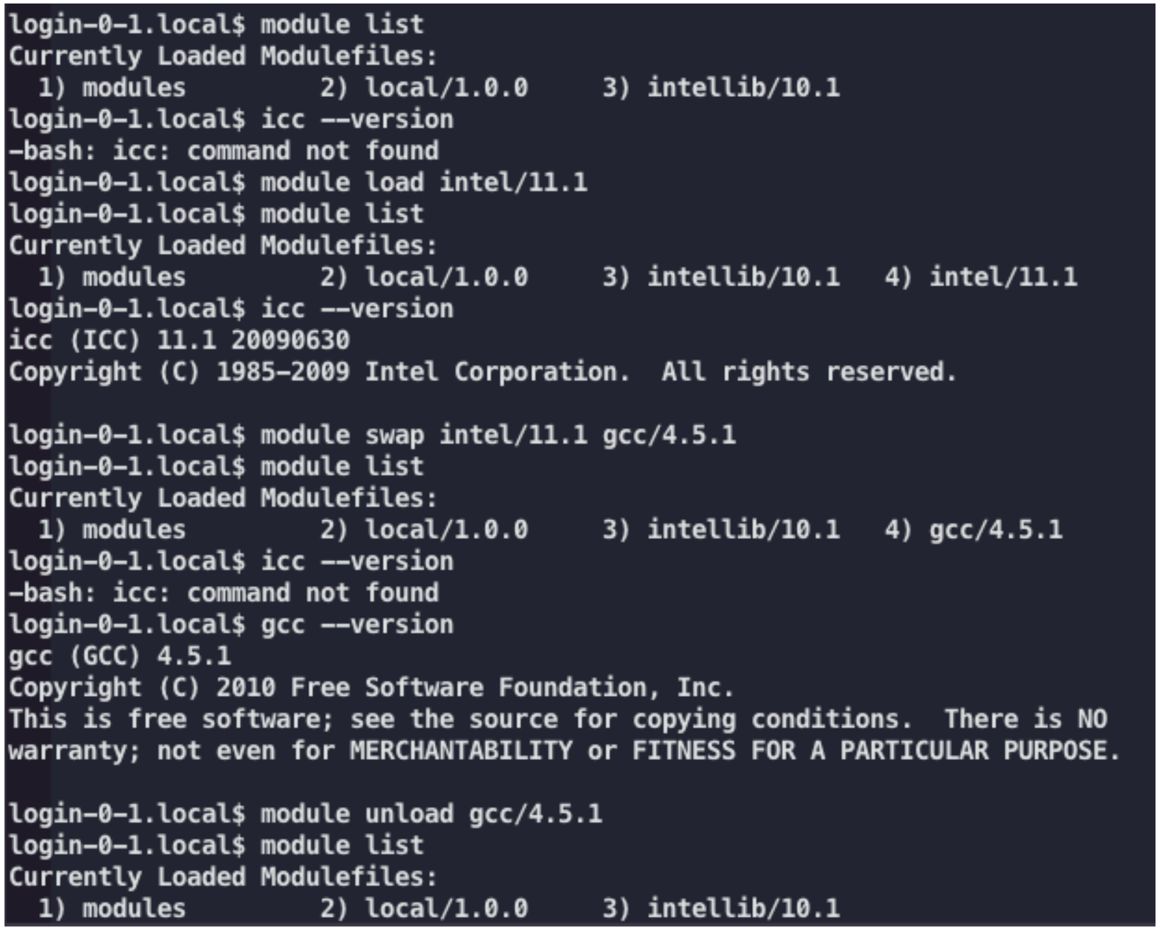
```
login-0-1.local$ module avail

---------------------------------------------------------------------------------------------------------- /site/Modules/versions ------------------------------------------------------------------------------------------
3.2.6

----------------------------------------------------------------------------------------------------- /site/Modules/3.2.6/modulefiles ----------------------------------------------------------------------------------------
dot         module-cvs  module-info modules     null        use.own

------------------------------------------------------------------------------------------------------ /xanadu/share/etc/modulefiles -----------------------------------------------------------------------------------------
454apps/1.1.03.24               biogrid/blast                   delph-in/intel-acml-last        hdf5/1.8.4.patch1               molpro/2008.1(default)          openmpi/1.4.3.intel.test        procov/2.0(default)
454apps/2.0.01.12               biogrid/blast+                  delph-in/intel-atlas-last       hdf5.pic/1.8.1                  molpro/2010.1                   openmpi/1.4.3.open64            pymol/1.2(default)
454apps/2.3                     biogrid/clustalw                delph-in/intel-goto2            hdfview/2.5(default)            molpro/2010.1.p3                openmpi/1.4.3.pgi               pyronoise/2(default)
454apps/2.5.3(default)          biogrid/hmmer                   delph-in/intel-goto2-last       hmmer/2.3.2(default)            mpiblast/1.6.0(default)         openmpi/1.4.intel               pysundials/2.3.0b6(default)
ACESII/mab_2005.1(default)      biogrid/mafft                   delph-in/intel-mkl              hmmer/3.0                       mtag/1.0(default)               openmpi/1.4.intel.icc           python/2.6.2(default)
AmpliconNoise/1.21(default)     biogrid/muscle                  delph-in/intel-mkl-last         ibmpich/1.0(default)            muscle/3.7(default)             openssl/1.0.0b(default)         python/2.7
R/2.10.1                        blast/2.2.18(default)           ePCR/svn(default)               imagemagick/6.5.3.3(default)    netcdf/4.0.1.intel              p4vasp/0.3.19                   python/2.7.gcc45
R/2.11.1(default)               blat/3.4(default)               eclipse/2009.2                  instant/0.9.4                   netcdf/4.1.1(default)           papi/3.6.2                      python/3.1.1
R/2.9.1                         blcr/0.6.1                      espresso/4.2.1(default)         instant/0.9.5(default)          netcdf.g77/3.6.2(default)       papi/4.1.0(default)             qt/4.3.3(default)
R/2.9.2                         blitz/0.9                       fastx/0.0.10(default)           intel/10.1                      netcdf.gfortran4/3.6.2(default) paralleloctave/x.x.0(default)   root/v5.24.00
RAxML/7.0.4(default)            breve/2.4.1                     ffmpeg/0.5(default)             intel/11.0                      netcdf.intel/3.6.2(default)     paraview/3.4(default)           root/v5.27.06(default)
RAxML/7.2.6                     breve/2.7(default)              flex/2.5.35(default)            intel/11.1(default)             netcdf.pathscale/3.6.2(default) paraview/3.6.1                  ruby/1.9.1(default)
acml.gfortran/3.6.0             celera/6.1(default)             fsl/4.1.2(default)              intel/11.1u7                    netcdf.pgi/3.6.2(default)       pathscale/3.0                   scampi/5.6.1(default)
acml.gfortran/4.1.0             cfitsio/3.24(default)           fsl/4.1.7                       intel/11.1u8                    newbler/1.1.03.24               pathscale/3.2(default)          scipy/0.7.2(default)
acml.gfortran/4.4.0             cilk/5.4.6(default)             g03/d02(default)                intel/2011.1                    newbler/2.0.01.12               perl/1.0                        scipy/0.8.0
acml.gnu/3.6.0                  clustalw/2.0.9(default)         g09/b01                         intel/9.1                       newbler/2.3(default)            perl/1.6                        structure/2.2.2(default)
acml.gnu/4.1.0                  cmake/2.6.4(default)            g95/093                         intellib/10.1(default)          numpy/1.4.1(default)            perl/1.7(default)               structure/2.2.3
acml.ifort/3.6.0                cmiss/x.x.0(default)            gamessplus/v2010(default)       intellib/11.0                   numpy/1.5.1rc1                  pgi/10.2(default)               sundials/2.3.0(default)
acml.ifort/4.1.0                cmkl/8.1(default)               garli/0.94                      intellib/11.1                   ocaml/3.11.1(default)           pgi/10.5                        svn/1.4.6(default)
acml.ifort/4.2.0                compython/2.1                   garli/1.0(default)              intellib/9.1                    ode/0.11.1                      pgi/10.9                        tables/2.1.2(default)
acml.ifort/4.3.0                compython/2.3(default)          gcc/4.5.1(default)              ipm/0.982                       ode/0.11.1s(default)            pgi/11.1                        unphased/3.0.10
acml.ifort/4.4.0                compython/2.5                   gcc/4.6                         ipm/0.982b                      open64/4.2.1                    pgi/7.1                         unphased/3.0.13(default)
acml.pathscale/3.6.0            compython/2.5.openmpi           gdl/0.9                         ipm/0.983(default)              open64/4.2.3                    phredphrapconsed/1.0            valgrind/3.5.0(default)
acml.pathscale/4.1.0            compython/bak.2.1               gdl/0.9rc2                      jags/1.0.3                      openmpi/1.2.8.gnu               phredphrapconsed/2.0(default)   vasp/4.6.36.parallel
acml.pgi/3.6.0                  configobj/4.7.2(default)        gdl/0.9rc4(default)             jags/2.2.0(default)             openmpi/1.2.8.intel             phylobayes/3.2d                 vasp/5.2.11.parallel
acml.pgi/4.1.0                  delph-in/1.00(default)          gnuplot/4.2.5                   local/1.0.0(default)            openmpi/1.3.3.gnu(default)      phylobayes/3.2e(default)        vasp/5.2.2.parallel
adf/2008.01(default)            delph-in/gcc-acml-last          gnuplot/4.4.2(default)          mafft/6.603                     openmpi/1.3.3.intel             phyml/2.4.5(default)            vislcg3/0.9.7.6500(default)
adf/2009.01                     delph-in/gcc-atlas-last         graphviz/2.20.2                 mafft/6.811                     openmpi/1.3.3.intel.ipath       phyml/3.0                       vislcg3/1.0
adf/2010.02                     delph-in/gcc-goto2-last         graphviz/2.6(default)           mafft/6.833(default)            openmpi/1.3.3.pgi               phyml/3.vector                  vislcg3/gprof
asreml/2.00ah(default)          delph-in/gcc-mkl-last           gromacs/4.5.3(default)          matlab/7.2                      openmpi/1.4.1.gnu               plink/1.05                      vislcg3/test5
beast/1.5.4                     delph-in/gnu                    gsl/1.12                        matlab/7.7                      openmpi/1.4.1.intel             plink/1.06                      xz/5.0.0(default)
beast/1.6.1(default)            delph-in/gnu-acml               gsl/1.13(default)               matlab/R2008b(default)          openmpi/1.4.2.intel             plink/1.07(default)
bedtools/2.7.1(default)         delph-in/gnu-goto               gsl/1.8.32b                     meltprofile/1.0(default)        openmpi/1.4.3.gnu               plplot/5.9.4(default)
biogrid/1.00(default)           delph-in/intel                  h5py/1.3.0(default)             mesalib/7.6.1(default)          openmpi/1.4.3.gnu32             poy/4.1.1(default)
biogrid/1.00+                   delph-in/intel-acml             hdf5/1.8.1                      mira/3.0.5(default)             openmpi/1.4.3.intel             prank/100802(default)
```

Environment Modules package (2)



```
login-0-1.local$ module list
Currently Loaded Modulefiles:
  1) modules         2) local/1.0.0     3) intellib/10.1
login-0-1.local$ icc --version
-bash: icc: command not found
login-0-1.local$ module load intel/11.1
login-0-1.local$ module list
Currently Loaded Modulefiles:
  1) modules         2) local/1.0.0     3) intellib/10.1   4) intel/11.1
login-0-1.local$ icc --version
icc (ICC) 11.1 20090630
Copyright (C) 1985-2009 Intel Corporation.  All rights reserved.

login-0-1.local$ module swap intel/11.1 gcc/4.5.1
login-0-1.local$ module list
Currently Loaded Modulefiles:
  1) modules         2) local/1.0.0     3) intellib/10.1   4) gcc/4.5.1
login-0-1.local$ icc --version
-bash: icc: command not found
login-0-1.local$ gcc --version
gcc (GCC) 4.5.1
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

login-0-1.local$ module unload gcc/4.5.1
login-0-1.local$ module list
Currently Loaded Modulefiles:
  1) modules         2) local/1.0.0     3) intellib/10.1
```

2. Showing how many modules have already loaded for you.

`module list`

3. Loading and unloading a specified module.

`module load` ***`xxxmodule`***

`module unload` ***`xxxmodule`***

4. Replacing a module with another one.

`module swap` ***`oldModule newModule`***

You may add loading of modules to your ***.bash_login*** file to make sure your favorite module is always loaded when logging in.

Titan

There are different C/C++ compiler systems, GNU, Intel, Pathscale, etc. , installed on Titan. Here we show examples with GNU compiler respectively for serial, OpenMP and MPI codes.

	Serial	OpenMP	MPI
Step 1 Loading module	`module load gcc/4.5.1`	`module load gcc/4.5.1`	`module load openmpi/1.4.3.gnu`
Step 2 Compiling	`gcc –o app main.c`	`gcc –fopenmp –o app main.c`	`mpicc –o app main.c`

mpicc is not a new compiler, which is just a wrapper of *gcc*. Type the command "`mpicc –show`" you will get the wrapper information.

```
gcc -I/site/VERSIONS/openmpi-1.4.3.gnu/include -pthread -L/
site/VERSIONS/openmpi-1.4.3.gnu/lib -lmpi -lopen-rte -lopen-pal
-ldl -Wl,--export-dynamic -lnsl -lutil -lm -ldl
```

Queue system (1)

On the Titan, one needs to use a script to submit a job into a queue that is managed by a scheduler and resource manager.

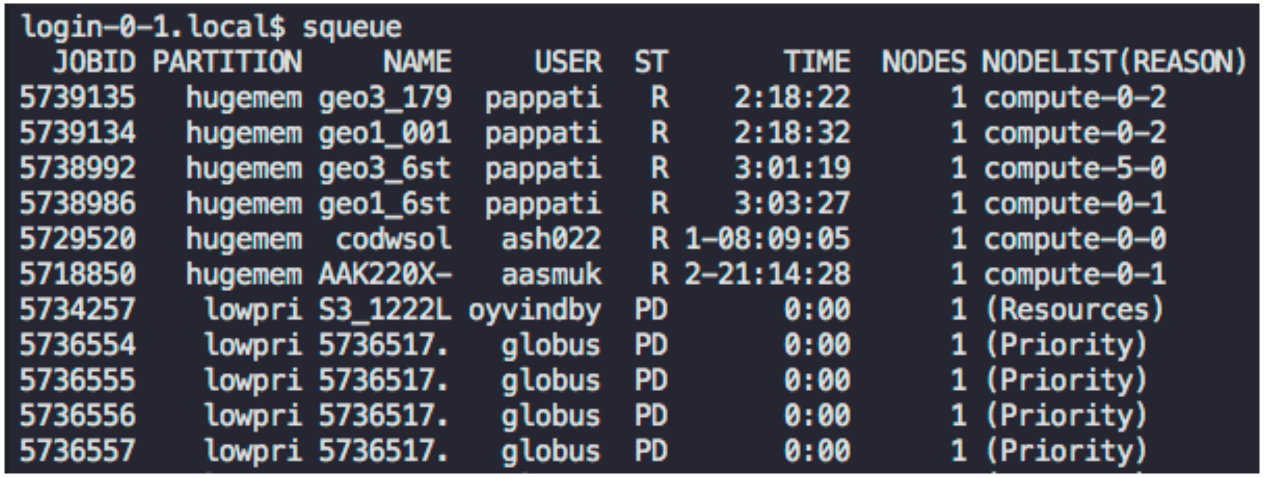
The Simple Linux Utility for Resource Management (SLURM) adpoted on Titan is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system.

Most useful commands of SLURM.

1. Inspecting jobs and the queue.

```
squeue

squeue –u username #To see jobs belong to a specified user
```



```
login-0-1.local$ squeue
  JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
5739135   hugemem geo3_179  pappati   R    2:18:22      1 compute-0-2
5739134   hugemem geo1_001  pappati   R    2:18:32      1 compute-0-2
5738992   hugemem geo3_6st  pappati   R    3:01:19      1 compute-5-0
5738986   hugemem geo1_6st  pappati   R    3:03:27      1 compute-0-1
5729520   hugemem  codwsol   ash022   R 1-08:09:05      1 compute-0-0
5718850   hugemem AAK220X-   aasmuk   R 2-21:14:28      1 compute-0-1
5734257    lowpri S3_1222L oyvindby  PD       0:00      1 (Resources)
5736554    lowpri 5736517.   globus  PD       0:00      1 (Priority)
5736555    lowpri 5736517.   globus  PD       0:00      1 (Priority)
5736556    lowpri 5736517.   globus  PD       0:00      1 (Priority)
5736557    lowpri 5736517.   globus  PD       0:00      1 (Priority)
```

Queue system (2)

2. Submitting a job described in a file "*job.scp*".

```
sbatch job.scp
```

3. Cancelling running or pending (waiting) jobs.

```
scancel jobid # Cancel job with id jobid (as returned from sbatch)

scancel --user=myusername # Cancel all your jobs
```

4. Showing information about a specified job with "*jobid* ".

```
scontrol show job jobid
```

5. Showing project information of a "*username*".

```
projects –u myusername
```

Job script (1)

Serial job

```
#!/bin/bash
# Job name:
#SBATCH --job-name=SerialTest

# Project:
#SBATCH –account=inf3380

# Wall clock limit:
#SBATCH --time=00:10:00

# Max memory usage:
#SBATCH --mem-per-cpu=1G

## Set up job environment
source /site/bin/jobsetup

## Do some work:
./app
```

A Job script is like an Unix-shell script, which can execute Unix commands before executing your "real" job.

Any line starting as "**#**" will be ignored by *bash* while "**#SBATCH**" will be used by the queue system.

You can specify a job name.

State the project name, which will be charged

The maximum time for running your code, the code will be terminated when it runs out the specified time. NB, the longer time you specify, the longer waiting time you may need.

'M' for megabytes or 'G' for gigabytes, e.g., 512M, 2G.

`jobsetup` will set up needed environment variables and shell functions.

Run your own code.

Job script (2)

Parallel job

OpenMP and Threading

```
#!/bin/bash
# Job name:
#SBATCH --job-name=OMPTest
#
# Project:
#SBATCH --account=inf3380
#
# Wall clock limit:
#SBATCH --time=00:10:00
#
# Max memory usage per core (MB):
#SBATCH --mem-per-cpu=500M
#
# Number of cores:
#SBATCH --ntasks-per-node=NumCores

## Set up job environment
source /site/bin/jobsetup

## Number of threads controlled by OpenMP is set by:
export OMP_NUM_THREADS=$SLURM_NTASKS_PER_NODE

## Run command
./app
```

Number of cores will be used for a OpenMP code, which should be same with the number of OpenMP threads specified in **$OMP_NUM_THREADS.**

Job script (3)

Parallel job

MPI

```
#!/bin/bash
# Job name:
#SBATCH --job-name=MPITest
#
# Project:
#SBATCH --account=inf3380
#
# Wall clock limit:
#SBATCH --time=00:10:00
#
# Max memory usage per task:
#SBATCH --mem-per-cpu=500M
#
# Number of tasks (cores):
#SBATCH --ntasks-per-node=8
#SBATCH --nodes=4
## Set up job environment:
. /site/bin/jobsetup
module load openmpi/1.4.3.gnu

mpirun ./app
```

Here we specify 8 cores per node and 4 nodes, thus 4x8=32 MPI processes will be used in total.

The combination of **`#SBATCH --ntasks-per-node=8`** and **`#SBATCH –nodes=4`** can get 4 complete nodes, which gives better performance than simply using **`#SBATCH --ntasks=32`** that however may start your job faster.

Specify the MPI library for your code, *openmpi* is compiled with *GCC*. If you code is compiled with Intel compiler, you should use the following one instead.

module load openmpi/1.2.8.intel

Work Directory

The `/site/bin/jobsetup` command creates a directory named `$SCRATCH` on local disc on each node.

1. If your job uses a lot of files, or does much random access on the files, you are strongly advised to use `$SCRATCH` as the work directory for the job.

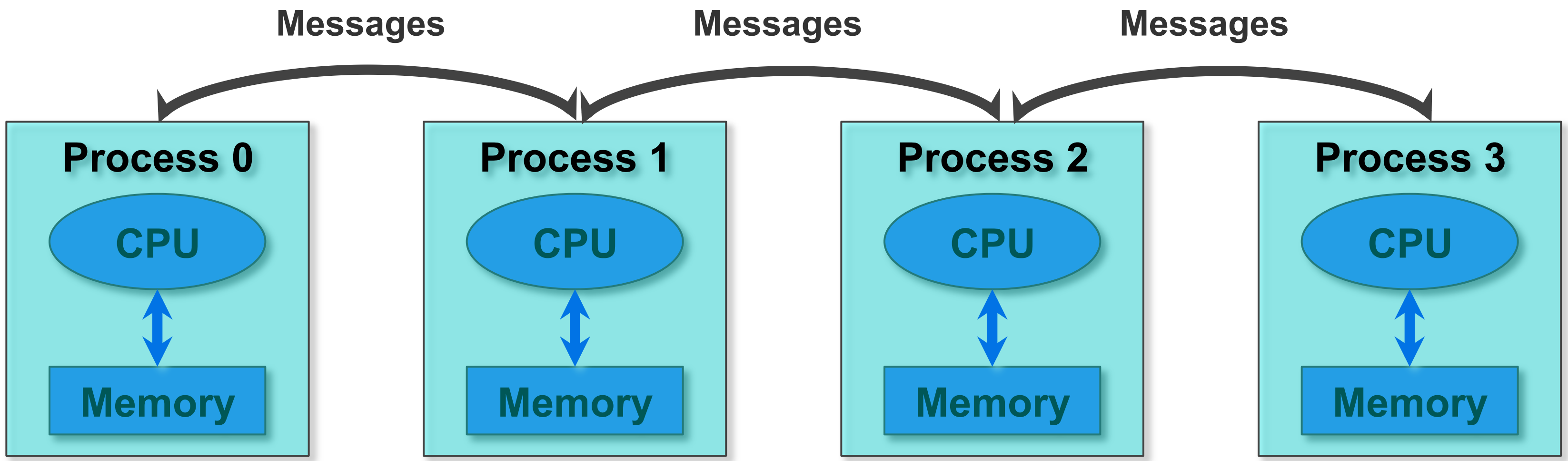
2. If your job uses only few files, and sequentially reads or writes them, there is no need to use `$SCRATCH`. You can just use your home directory (i.e. the directory where you ran `sbatch`). It is simpler [3]. The introduced job scripts are in this case.

More details can be found at https://wiki.uio.no/usit/suf/vd/hpc/index.php/Titan_User_Guide#Work_Directory

MPI (1)

What is Message Passing Interface (MPI)?

1. It is not a new progamming language but a library specification for message-passing.
2. It has become a *de facto* programming standard for distributed memory systems.
3. Most MPI implementations offer a set of APIs callable from *Fortran, C/C++, Java, Python, etc.*



MPI (2)

Here we show the simplest MPI code and the whole procedure of compiling and running it on Titan.

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

int main(int argc, char* argv[]){
  int rank;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  printf("Rank %d: Hello world!\n", rank);
  MPI_Finalize();
  return 0;
}
```

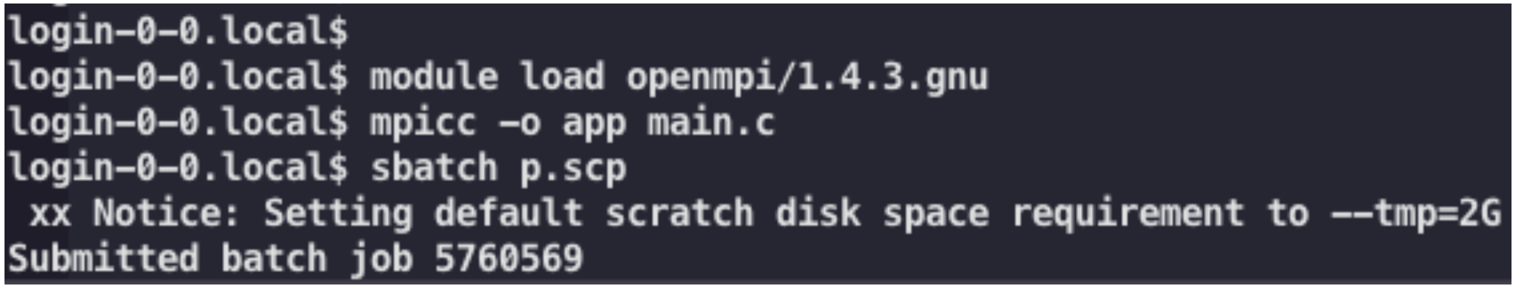
- `#include <mpi.h>` → The head file of MPI library.
- `MPI_Init(&argc, &argv);` → Compulsory initialization of MPI, which makes each process deal with parameters correctly. Call it as early as possible.
- `MPI_Comm_rank(MPI_COMM_WORLD,&rank);` → Get the current process ID.
- `MPI_Finalize();` → Compulsory finalization of MPI.

```
#!/bin/bash
#SBATCH --job-name=MPITest
#SBATCH --account=inf3380
#SBATCH --time=00:10:00
#SBATCH --mem-per-cpu=500M
#SBATCH --ntasks-per-node=8
#SBATCH --nodes=1
## Set up job environment:
. /site/bin/jobsetup
module load openmpi/1.4.3.gnu
mpirun ./app
```

- `#SBATCH --ntasks-per-node=8` → We run the code with 8 cores from the same node.

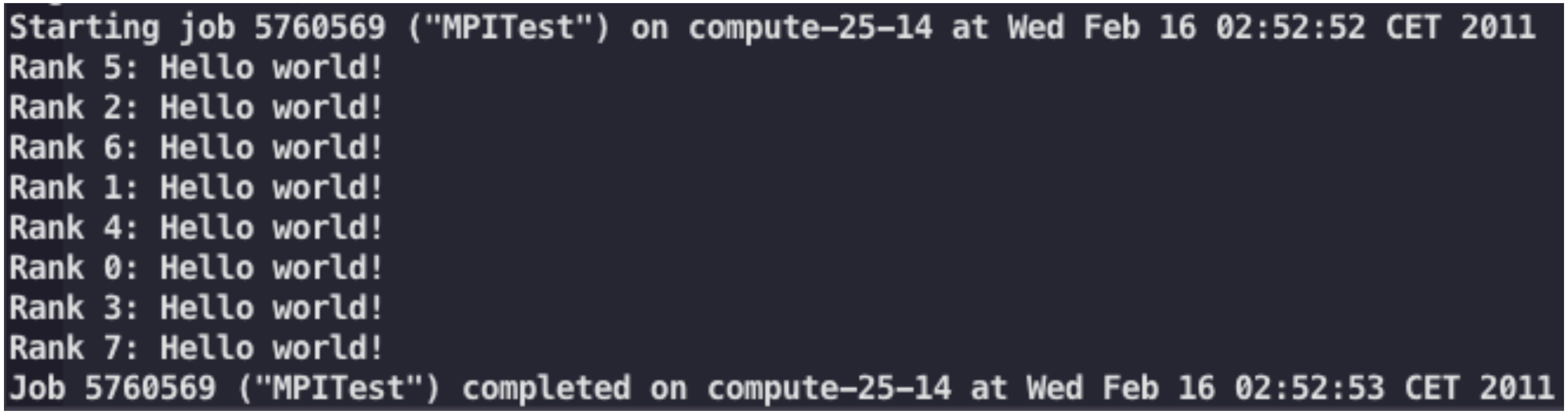
MPI (3)

Compiling and submitting the job



```
login-0-0.local$
login-0-0.local$ module load openmpi/1.4.3.gnu
login-0-0.local$ mpicc -o app main.c
login-0-0.local$ sbatch p.scp
 xx Notice: Setting default scratch disk space requirement to --tmp=2G
Submitted batch job 5760569
```

Output

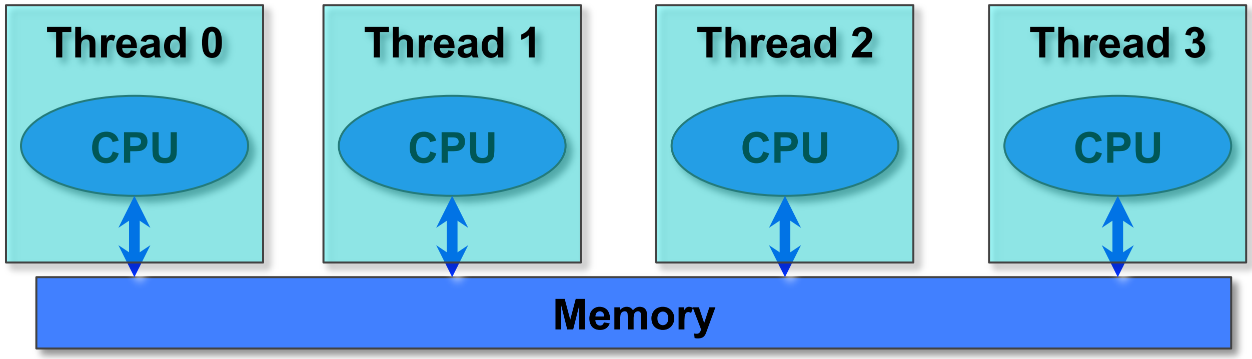


```
Starting job 5760569 ("MPITest") on compute-25-14 at Wed Feb 16 02:52:52 CET 2011
Rank 5: Hello world!
Rank 2: Hello world!
Rank 6: Hello world!
Rank 1: Hello world!
Rank 4: Hello world!
Rank 0: Hello world!
Rank 3: Hello world!
Rank 7: Hello world!
Job 5760569 ("MPITest") completed on compute-25-14 at Wed Feb 16 02:52:53 CET 2011
```

OpenMP (1)

What is OpenMP?

1. OpenMP is not a new programming language.
2. It is a set of runtime routines and compiler directives aiding OpenMP-aware compilers to create parallel applications on shared memory system.
3. OpenMP thus only can be used with certain languages. By far, it is supported by C/C++ and Fortran.



OpenMP (2)

A certain compiler option is necessary for dealing with directives of OpenMP.

GNU Compiler *gcc/g++/gfortran (compiler option -fopenmp)*

Intel Compiler *icc/icpc (compiler option -openmp)*

Portland Group Compiler *pgcc/pgCC (compiler option -mp)*

Pathscale Compiler *pathcc/pathCC (compiler option -mp)*

OpenMP (3)

Here we show a simplest OpenMP code and the whole procedure of compiling and running it on Titan.

```
#include <stdio.h>
#include <omp.h>
main(){
#pragma omp parallel
 {
   int myNum = omp_get_thread_num();
   printf("Thread %d: Hello, world!\n", myNum);
 }
}
```

→ The head file of OpenMP library.

→ Mark out a parallel region, where a group of threads will be spawned.

→ Each thread gets its thread ID.

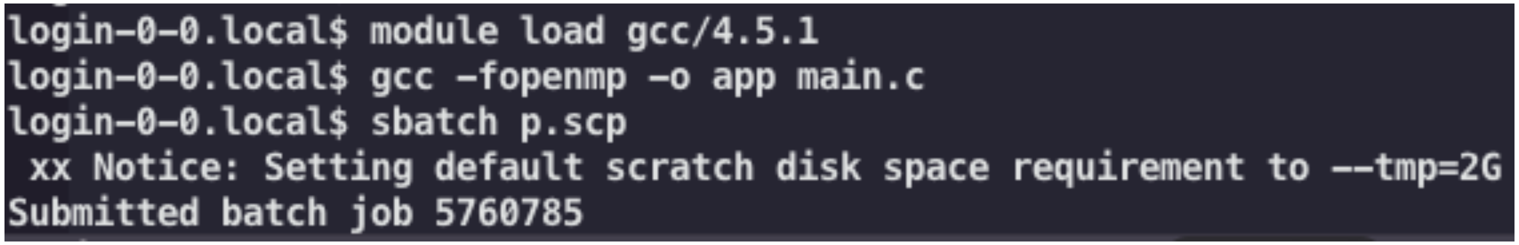
```
#!/bin/bash
#SBATCH --job-name=OMPTest
#SBATCH --account=inf3380
#SBATCH --time=00:10:00
#SBATCH --mem-per-cpu=100M
#SBATCH --ntasks-per-node=8
## Set up job environment
source /site/bin/jobsetup
## Number of threads controlled by OpenMP is set by:
export OMP_NUM_THREADS=$SLURM_NTASKS_PER_NODE

## Run command
./app
```

→ We run the code with 8 cores in one node.

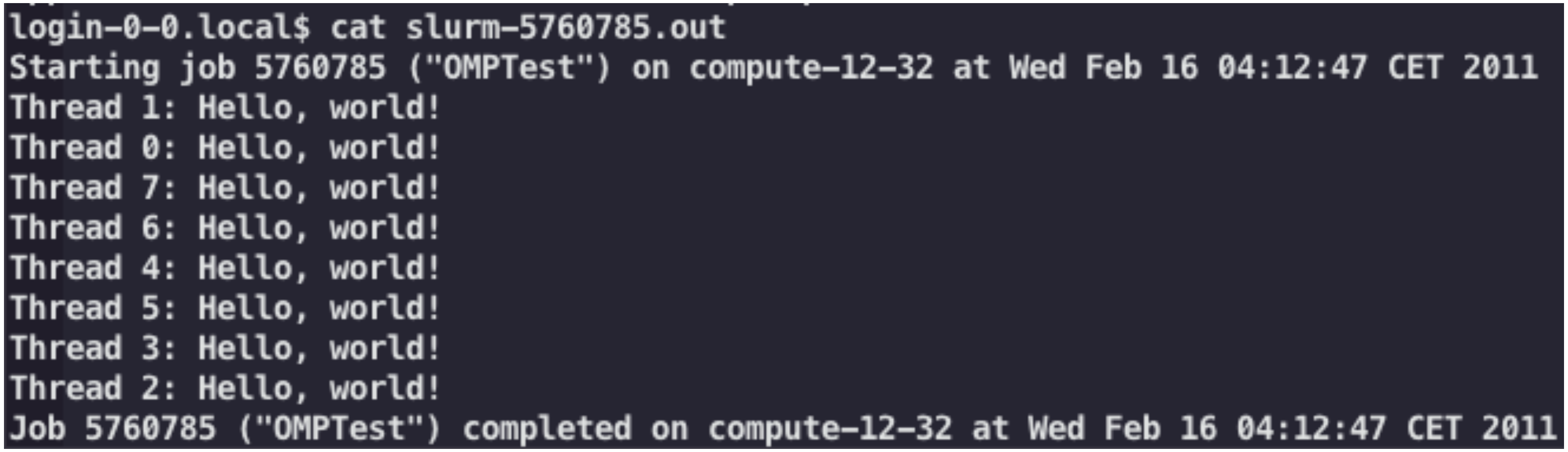
OpenMP (4)

Compiling and submitting the job



```
login-0-0.local$ module load gcc/4.5.1
login-0-0.local$ gcc -fopenmp -o app main.c
login-0-0.local$ sbatch p.scp
 xx Notice: Setting default scratch disk space requirement to --tmp=2G
Submitted batch job 5760785
```

Output



```
login-0-0.local$ cat slurm-5760785.out
Starting job 5760785 ("OMPTest") on compute-12-32 at Wed Feb 16 04:12:47 CET 2011
Thread 1: Hello, world!
Thread 0: Hello, world!
Thread 7: Hello, world!
Thread 6: Hello, world!
Thread 4: Hello, world!
Thread 5: Hello, world!
Thread 3: Hello, world!
Thread 2: Hello, world!
Job 5760785 ("OMPTest") completed on compute-12-32 at Wed Feb 16 04:12:47 CET 2011
```

Time measurements

Timing is an important part of performance study. Concrete timing subroutines for serial and parallel codes will be presented.

In a serail code, **`gettimeofday(struct timeval *timmer)`** is widely used to measure a wall time.

```
#include <stdio.h>
#include <sys/time.h>

double timing(){
        double time;
        struct timeval timmer;

        gettimeofday(&timmer,NULL);
        time = 1000000*timmer.tv_sec + timmer.tv_usec;
        time /= 1000000;
        return time;
}
```

A simple timing subroutine

We can use the following for timing

```
double t1, t2, t;
t1 = timing();
……
t2 = timing();
t = t2 - t1;
```

Time measurements

MPI:

Basically, all subroutines that can be used in serial codes also can be used in MPI codes, meanwhile MPI library also offers a subroutine **`double MPI_Wtime()`** for convenience.

OpenMP:

Thread safe is very important to OpenMP and other multi-threaded programming models, which determines if a subroutine can work correctly in a parallel region.

When you want to measure run time in a parallel region, a thread-safe subroutine **`double omp_get_wtime()`** from OpenMP runtime library is the best choice.

Using OpenMP on your laptop

Ubuntu:

The earliest version of gcc supporting the OpenMP is 4.2.0. You can check the version by using *gcc –version*.

- **Using** *sudo apt-get install libgomp1, sudo apt-get install gcc-4.2* **to install the OpenMP library and compiler.**
- **Downloading and recompiling the latest gcc source package.**
  - **Warning: It might damage your compiler system if you are not familiar with Linux system.**
- **Reinstall the latest Ubuntu distribution.**

Mac:

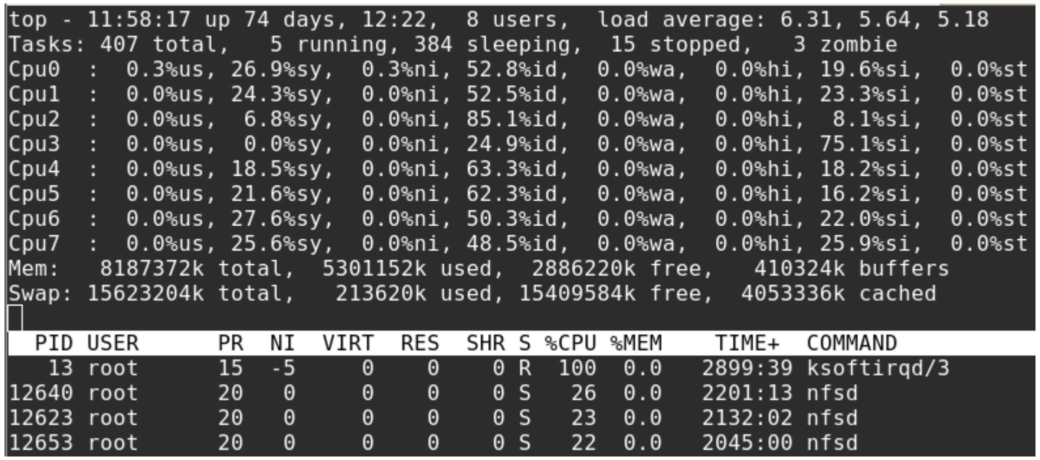
Install the Xcode Tools from http://developer.apple.com/technologies/xcode.html

Using OpenMP on your laptop

Useful commands in linux/unix:

```
wenjie@bigblue:~/temp$ top
```

After activating ***top***, press "***1***" then you have the following show for all the CPU cores.

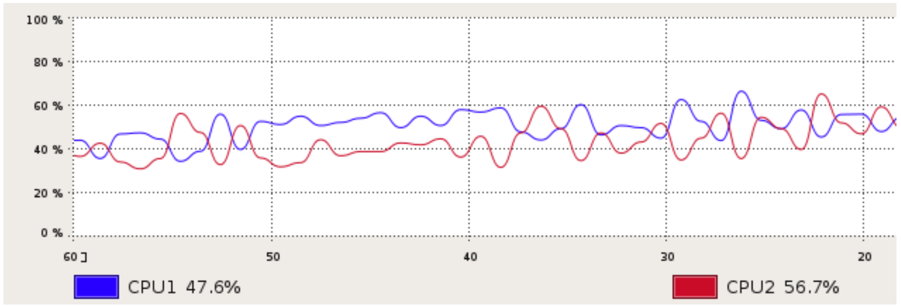


```
top - 11:58:17 up 74 days, 12:22,  8 users,  load average: 6.31, 5.64, 5.18
Tasks: 407 total,   5 running, 384 sleeping,  15 stopped,   3 zombie
Cpu0  :  0.3%us, 26.9%sy,  0.3%ni, 52.8%id,  0.0%wa,  0.0%hi, 19.6%si,  0.0%st
Cpu1  :  0.0%us, 24.3%sy,  0.0%ni, 52.5%id,  0.0%wa,  0.0%hi, 23.3%si,  0.0%st
Cpu2  :  0.0%us,  6.8%sy,  0.0%ni, 85.1%id,  0.0%wa,  0.0%hi,  8.1%si,  0.0%st
Cpu3  :  0.0%us,  0.0%sy,  0.0%ni, 24.9%id,  0.0%wa,  0.0%hi, 75.1%si,  0.0%st
Cpu4  :  0.0%us, 18.5%sy,  0.0%ni, 63.3%id,  0.0%wa,  0.0%hi, 18.2%si,  0.0%st
Cpu5  :  0.0%us, 21.6%sy,  0.0%ni, 62.3%id,  0.0%wa,  0.0%hi, 16.2%si,  0.0%st
Cpu6  :  0.0%us, 27.6%sy,  0.0%ni, 50.3%id,  0.0%wa,  0.0%hi, 22.0%si,  0.0%st
Cpu7  :  0.0%us, 25.6%sy,  0.0%ni, 48.5%id,  0.0%wa,  0.0%hi, 25.9%si,  0.0%st
Mem:   8187372k total,  5301152k used,  2886220k free,   410324k buffers
Swap: 15623204k total,   213620k used, 15409584k free,  4053336k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
   13 root      15  -5     0    0    0 R  100  0.0   2899:39 ksoftirqd/3
12640 root      20   0     0    0    0 S   26  0.0   2201:13 nfsd
12623 root      20   0     0    0    0 S   23  0.0   2132:02 nfsd
12653 root      20   0     0    0    0 S   22  0.0   2045:00 nfsd
```

GUI tools in GNOME.

```
wenjie@bigblue:~/temp$ gnome-system-monitor
```



CPU load will be displayed in different colors. It is mainly for your desktop/laptop system.

Useful links

Titan User Guide (queue system, job script and modules)
https://wiki.uio.no/usit/suf/vd/hpc/index.php/Titan_User_Guide#Modules

Titan software (compiler, MPI library and the other precompiled applications)
https://wiki.uio.no/usit/suf/vd/hpc/index.php/Titan_software

Environment Modules Project
http://modules.sourceforge.net/

Simple Linux Utility for Resource Management (SLURM)
https://computing.llnl.gov/linux/slurm/overview.html

The OpenMP specification
http://www.openmp.org/blog/

The GNU OpenMP Implementation
http://gcc.gnu.org/onlinedocs/libgomp.pdf

Open MPI
http://www.open-mpi.org/

References

[1] https://wiki.uio.no/usit/suf/vd/hpc/index.php/TITAN

[2] http://modules.sourceforge.net/

[3] https://wiki.uio.no/usit/suf/vd/hpc/index.php/Titan_User_Guide