



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Summer School 2015

Programming Environment

<http://github.com/eth-cscs/SummerSchool2015/wiki> - DAY1

20th July 2015

Summary

- Accessing CSCS
- Remote display
- Compiling my code
- Running my code
- Editing my code
- Transferring files from/to CSCS
- Debugging tools



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Accessing CSCS

Accessing CSCS: SSH

Secure shell: 2 steps

- Piz Daint is hidden: ela / daint10x / daint0x / nidxxx
- frontend first Ela: `ssh -X courseNN@ela.cscs.ch`
- then login node Piz Daint: `ssh -X daint`

```
ssh -X course51@ela.cscs.ch
```

```
=====
                IMPORTANT REMINDER FOR USERS of CSCS facilities
                help@cscs.ch - +41 91 610 82 10 - http://user.cscs.ch
=====
CSCS next scheduled preventive maintenance is scheduled for:
Wednesday September 02, 2015
Center-Wide      07:30 - 18:00

course51@ela1:~
```

```
ssh -X daint
```

```
course51@ela1:~ ssh -X daint
course51@daint101:~ xclock
```

Accessing CSCS: VNC (1/2)

Virtual Network Computing: 2 steps

- Start the VNC server on `Piz Daint`: `vncserver`
- Start the VNC client on your laptop: `vncviewer` ...

vnc server

```
course51@daint103:~ $ vncpasswd
Password: *****
Verify:      *****
Would you like to enter a view-only password (y/n)? n

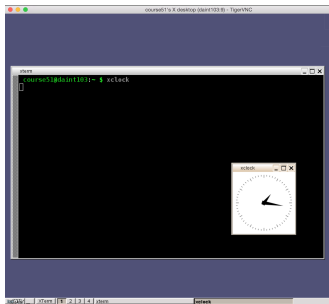
course51@daint103:~ $ vncserver
New 'X' desktop is daint103:9
```

vnc client

```
jg@mylaptop:~ ssh -f \
    -l course51 \
    -L 5909:daint103.login.cscs.ch:5909 \
    -C ela.cscs.ch \
    sleep 60

jg@mylaptop:~ vncviewer localhost:9
TigerVNC Viewer 64-bit v1.4.0
CConn:      connected to host localhost port 5909
```

Accessing CSCS: VNC (2/2)



- More infos:
<https://github.com/eth-cscs/SummerSchool2015/wiki>

Don't forget to kill the server when done

```
course51@daint103:~$ vncserver -kill :9
```



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Compiling the code

Compiling the code: setup your PE

Cray Programming Environment

- 4 compilers available: `CCE*`, `GNU`, `INTEL`, `PGI`
- 4 predefined Programming Environment:
 - `PrgEnv-cray`, `PrgEnv-gnu`, `PrgEnv-intel`, `PrgEnv-pgi`
 - `echo $PE_ENV` to get the current PrgEnv
- 3 wrappers available: `ftn` (Fortran), `cc` (C), `CC` (C++)
 - You `must` use the wrappers to compile for the compute node,
 - The wrappers support serial, OpenMP, MPI and Cuda codes.
- The wrappers are based on the `module` command
 - dynamic modification of the `programming environment` via modulefiles,
 - the wrappers will detect the loaded `modulefiles`,
 - and will automatically add the needed flags and libraries.

Compiling the code: module list

module list

```
*course51@daint103:~ $ module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.3
  2) eswrap/1.1.0-1.020200.1231.0
  3) switch/1.0-1.0502.54233.2.96.ari
  4) craype-network-aries
  5) craype/2.4.0
* 6) cce/8.3.12
  7) totalview-support/1.1.4
  8) totalview/8.11.0
* 9) cray-libsci/13.0.4
 10) udreg/2.3.2-1.0502.9275.1.12.ari
 11) ugni/5.0-1.0502.9685.4.24.ari
 12) pmi/5.0.7-1.0000.10678.155.25.ari
 13) dmapp/7.0.1-1.0502.9501.5.219.ari
 14) gni-headers/3.0-1.0502.9684.5.2.ari
 15) xpmem/0.1-2.0502.55507.3.2.ari
 16) job/1.5.5-0.1_2.0502.54585.3.66.ari
 17) dvs/2.5_0.9.0-1.0502.1873.1.145.ari
 18) alps/5.2.1-2.0502.9041.11.6.ari
 19) rca/1.0.0-2.0502.53711.3.127.ari
 20) atp/1.8.2
*21) PrgEnv-cray/5.2.40
*22) craype-sandybridge
 23) slurm
*24) cray-mpich/7.2.2
 25) ddt/5.0
```

Compiling the code: module swap

module swap from CCE to GNU

```
course51@daint103:~ module swap PrgEnv-cray PrgEnv-gnu
Currently Loaded Modulefiles:
 4) craype-network-aries
 5) craype-sandybridge
 7) cray-mpich/7.2.2
 9) craype/2.4.0
*10) gcc/4.8.2
13) cray-libsci/13.0.4
*25) PrgEnv-gnu/5.2.40

course51@daint103:~ ftn --version
GNU Fortran (GCC) 4.8.2 20131016 (Cray Inc.)
```

module swap from GNU to CCE

```
course51@daint103:~ module swap PrgEnv-gnu PrgEnv-cray
Currently Loaded Modulefiles:
 4) craype-network-aries
 5) craype-sandybridge
 7) cray-mpich/7.2.2
 9) craype/2.4.0
*10) cce/8.3.12
13) cray-libsci/13.0.4
*25) PrgEnv-cray/5.2.40

course51@daint103:~ ftn -V
Cray Fortran : Version 8.3.12
```

Compiling the code: module avail

module avail

```
course51@daint103:~ module avail
```

```
# --- COMPILERS ---
```

```
PrgEnv-cray/5.2.40 PrgEnv-gnu/5.2.40  
PrgEnv-intel/5.2.40 PrgEnv-pgi/5.2.40  
cce/8.3.12 gcc/4.8.2 intel/15.0.1.133 pgi/15.3.0  
cray-mpich/7.2.2 python/3.4.0
```

```
# --- TOOLS ---
```

```
ddt/5.0  
perftools/6.2.4  
scalasca/1.4.1 scorep/1.4.1 vampir/1.4.1  
visit/2.9 paraview/4.3
```

```
# --- LIBS ---
```

```
cray-libsci/13.0.4  
cray-hdf5-parallel/1.8.14 cray-netcdf-hdf5parallel/4.3.3.1  
cray-petsc-64/3.5.3.1 cray-tpsl-64/1.5.0 cray-trilinos/11.12.1.3  
fftw/3.3.4.3
```

```
# --- GPU ---
```

```
craype-accel-nvidia35  
cudatoolkit/6.5.14-1.0502.9613.6.1  
cray-libsci_acc/3.1.1
```

```
# --- APPS ---
```

```
cp2k/2.6 espresso/5.1.2 gromacs/5.0.4  
lammps/10Feb15 namd/2.9 nwchem/6.3r2 vasp/5.3
```

Compiling the code: module show/help

module avail cray-hdf5-parallel

```
course51@daint103:~ module avail cray-hdf5

----- /opt/cray/modulefiles ---
cray-hdf5/1.8.11      cray-hdf5-parallel/1.8.11
cray-hdf5/1.8.12      cray-hdf5-parallel/1.8.12
cray-hdf5/1.8.13      cray-hdf5-parallel/1.8.13
cray-hdf5/1.8.14(default)  cray-hdf5-parallel/1.8.14(default)
cray-hdf5/1.8.9       cray-hdf5-parallel/1.8.9
```

module show cray-hdf5-parallel

```
module show cray-hdf5-parallel # CCE
      setenv HDF5_DIR /opt/cray/hdf5-parallel/1.8.14/CRAY/83

module swap PrgEnv-cray PrgEnv-gnu # GNU
module show cray-hdf5-parallel # GNU
      setenv HDF5_DIR /opt/cray/hdf5-parallel/1.8.14/GNU/48
```

module help cray-hdf5-parallel

```
course51@daint103:~ module help cray-hdf5-parallel
# Module Specific Help for 'cray-hdf5-parallel/1.8.14'
# Doc: http://www.hdfgroup.org/HDF5/doc/index.html
# ...
```

Compiling the code: module load/rm

module load cray-hdf5-parallel

```
course51@daint103:~ module load cray- hdf5-parallel
course51@daint103:~ module list
course51@daint103:~ which h5dump
# /opt/cray/hdf5/1.8.14/bin/h5dump
```

module rm cray-hdf5-parallel

```
course51@daint103:~ module rm cray- hdf5-parallel
course51@daint103:~ which h5dump
# not set
```

Compiling the code: mini-app

Get the src

```
course51@daint103:~ git clone \  
https://github.com/eth-cscs/SummerSchool2015.git \  
SummerSchool2015.git \  
# Cloning into 'SummerSchool2015.git'...
```

Compile the Fortran version

```
course51@daint103:~ cd SummerSchool2015.git/miniapp/serial/fortran/  
course51@daint103:~ make clean  
# rm -f main *.o *.i *.mod output.*  
  
course51@daint103:~ make  
# ftn -O3 -fopenmp -c stats.f90 -o stats.o  
# ftn -O3 -fopenmp -c data.f90 -o data.o  
# ftn -O3 -fopenmp -c operators.f90 -o operators.o  
# ftn -O3 -fopenmp -c linalg.f90 -o linalg.o  
# ftn -O3 -fopenmp *.o main.f90 -o main
```

Compile the C++ version

```
course51@daint103:~ cd SummerSchool2015.git/miniapp/serial/cxx/  
course51@daint103:~ make clean; make  
# CC -O3 -fopenmp -c stats.cpp -o stats.o  
# CC -O3 -fopenmp -c data.cpp -o data.o  
# CC -O3 -fopenmp -c operators.cpp -o operators.o  
# CC -O3 -fopenmp -c linalg.cpp -o linalg.o  
# CC -O3 -fopenmp *.o main.cpp -o main
```



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Running the code

Running the code: interactive session (1)

salloc

- The job submission system used at CSCS is `SLURM`.
- `salloc` allows to connect to the compute node (`man salloc`)
- `salloc --res=summerschool`: our reservation for the course
- `salloc -N`: number of compute nodes (8 cores max per node)
- `salloc -t`: duration of the session in minutes (default is 1h)
- http://user.cscs.ch/getting_started/running_jobs/

salloc

```
course51@daint102:~ salloc --res=summerschool -t120 -N2
salloc: Pending job allocation 1145157
salloc: job 1145157 queued and waiting for resources
salloc: job 1145157 has been allocated resources
salloc: Granted job allocation 1145157
course51@daint102:~ echo "Hourra :-)"
```


Running the code: interactive session (2)

Other useful Slurm commands

- `squeue -u $USER` : what is the status of my salloc session ?
- `scontrol show job $JOBID` : more details about my session ?
- `scancel $JOBID` : cancel my session

squeue/scontrol/scancel

```
course51@daint102:~ squeue -u course51
```

JOBID	USER	ACCOUNT	ST	REASON	NODES	PRIORITY
1145201	course51	courses	PD	None	2	15044

```
course51@daint102:~ scontrol show job 1145201
```

```
JobId=1145201 Name=bash
```

```
UserId=course51(22854) GroupId=courses(30340)
```

```
Priority=15044 Nice=0 Account=courses QOS=normal
```

```
JobState=RUNNING Reason=None Dependency=(null)
```

```
...
```

```
NodeList=nid000[68-69]
```

```
BatchHost=daint03
```

```
NumNodes=2 NumCPUs=16 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
```

```
Socks/Node=* NtasksPerN:B:S:C=0:0:*:1 CoreSpec=0
```

```
MinCPUsNode=1 MinMemoryCPU=2G MinTmpDiskNode=0
```

```
WorkDir=/users/course51
```

```
course51@daint102:~ scancel 1145201
```

Running the code: interactive session (3)

```
aprun [options] myexecutable
```

- To run an application on the compute nodes, use the aprun command.
- Adapt the command to your needs:
 - `-n`: Total number of MPI tasks
 - `-N`: Number of MPI tasks per compute node (8 max per node)
 - `-d`: Number of OpenMP threads

Running the miniapp (serial version)

```
course51@daint102:~ aprun -n1 ./main 256 256 200 0.01
=====
                        Welcome to mini-stencil!
version  :: Fortran90 serial
mesh     :: 256 * 256      dx = 3.9215688593685627E-3
time     :: 200 time steps from 0 .. 1.00000000000000002E-2
=====
-----
simulation took 5.2622885461896658 seconds
13761 conjugate gradient iterations
2615.0219394495398 CG iterations per second
1328 nonlinear newton iterations
-----
Summer School 2015 18 Goodbye!
```



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Editing the code

Editing the code

Many text editors are available

- `vim` filename (X version: gvim)
- `emacs` -nw filename (X version: emacs)
- X only:
 - `gedit`
 - `/apps/pilatus/nedit/5.5/Linux-x86/bin/nedit`



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Moving data from/to CSCS

Moving data: scp

SCP

- Getting a file: `scp USER@FROM:remotefile localfile`
- Sending a file: `scp localfile USER@TO:remotefile`
- Add the `-rp` flag to `scp` to copy an entire directory

Getting 1 (or more) file from CSCS

```
jg@mylaptop:
scp course51@ela.cscs.ch:~/SummerSchool2015.git/miniapp/*.pdf .
# miniapp.pdf          30%  576KB  140.9KB/s   00:09  ETA
# miniapp.pdf          52%  992KB  128.1KB/s   00:06  ETA
# miniapp.pdf          85% 1600KB  116.1KB/s   00:02  ETA
# miniapp.pdf         100% 1875KB  104.2KB/s   00:18
jg@mylaptop: evince miniapp.pdf &
```

Sending 1 (or more) file to CSCS

```
jg@mylaptop: scp          mycode.c          course51@ela.cscs.ch:~
```



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Debugging tools

Why a debugger ?


- Complexity can be so great that it appears chaotic :
 - Many threads, no synchronisation between them
 - Use printf or the command line debugger (cuda-gdb) within accelerator regions ?
 - Not optimal from a user perspective
- Key debugging features must include:
 - Line by line execution of functions & kernels (cpu & gpu)
 - Pause execution on the host & device (breakpoints)
 - Inspect data on the host & device (variables and arrays)
 - Navigate between MPI tasks, OpenMP & CUDA threads
 - Detect memory errors on the host & device
 - Allinea DDT (Distributed Debugging Tool)
 - Designed for debugging multi-threaded (OpenMP), multi-process (MPI), and accelerated (Cuda, OpenACC) codes
 - written with (Fortran, C, C++)
 - DDT is supported on all CSCS systems (including whole PizDaint)

DDT: a simple hello world example

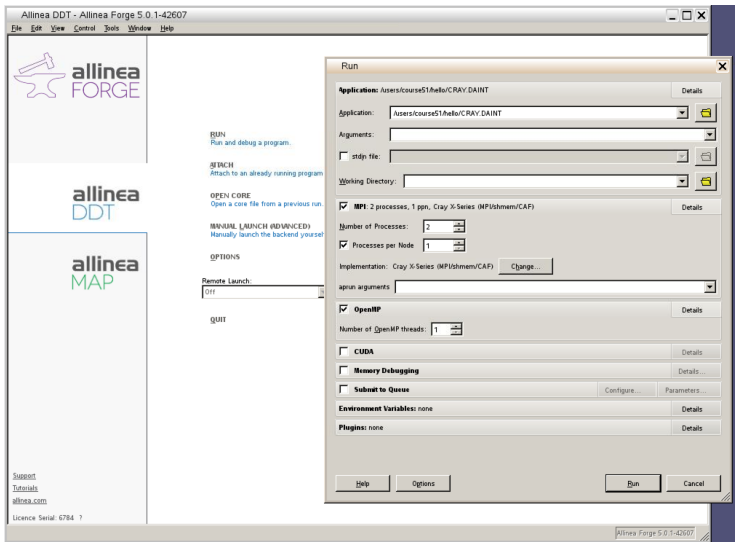
```
xterm
course51@daint101:~/hello $ salloc --res=summerschool -N2
salloc: Pending job allocation 1145487
salloc: job 1145487 queued and waiting for resources
salloc: job 1145487 has been allocated resources
salloc: Granted job allocation 1145487
salloc: Waiting for resource configuration
salloc: Nodes nid0000[8-9] are ready for job
course51@daint02:~ $
course51@daint02:~ $ aprun
course51@daint02:~ $ cd hello/
course51@daint02:~/hello $ aprun -n2 -N1 ./CRAY.DAINT
hello in f90 mpi/3.0= 0 / 2 omp/201107= 0 / 1 n=nid00008
hello in f90 mpi/3.0= 1 / 2 omp/201107= 0 / 1 n=nid00009
```

DDT: launching the debugger

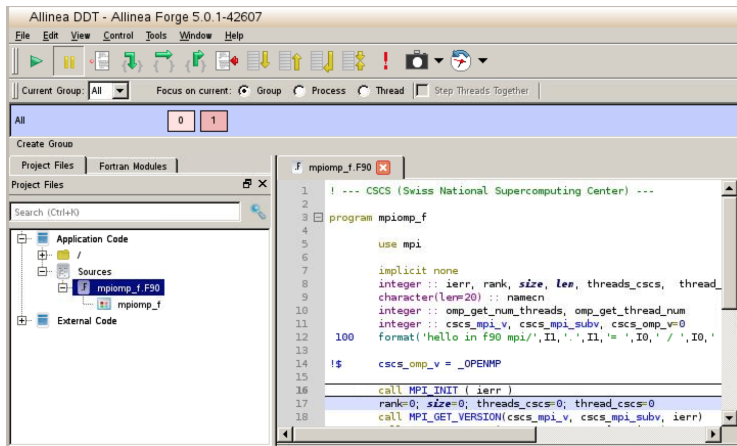
```
xterm
course51@daint02:~/hello $
course51@daint02:~/hello $
course51@daint02:~/hello $
course51@daint02:~/hello $ ddt ./CRAY.DAINT
```

The logo for allinea FORGE is displayed. It features a stylized purple anvil with a hammer resting on it. Below the graphic, the word "allinea" is written in a bold, black, sans-serif font, and the word "FORGE" is written in a purple, all-caps, sans-serif font.

DDT: setting up the job



DDT: starting the job



DDT: setting a breakpoint

The screenshot shows the Allinea DDT - Allinea Forge 5.0.1-42607 interface. The main window displays a Fortran program named `mpimp_f.F90`. A breakpoint is set at line 26, which is `write (6,100) cscs_mpi_v, cscs_mpi_subv, rank, size, cscs_mpi_v`. The program is stopped at this breakpoint. The 'Locals' panel on the right shows the current state of variables:

Variable Name	Value
cscs_mpi_subv	0
cscs_mpi_v	3
cscs_mpi_v	201107
namecn	76800000
rank	0
size	2
thread_cscs	0
threads_cscs	1

A 'Program Stopped' dialog box is displayed in the foreground, indicating that the program has stopped at a breakpoint. The dialog box contains the following information:

- Processes 0-1:
- Process stopped at breakpoint in `mpimp_f_craymt0001 (mpimp_f.F90:26)`.
- ☒ Always show this window for user-defined breakpoints.
- Buttons: Continue, Pause.

DDT: inspecting data

The screenshot displays the Allinea DDT - Allinea Forge 5.0.1-42607 application. The main window shows a Fortran code file named `J: mpiomp_f.F90`. The code includes MPI initialization, a parallel loop, and finalization. The project tree on the left shows the file structure. The locals window on the right displays the current state of variables.

Code Snippet:

```
16 call MPI_INIT ( ierr )
17 rank=0; size=0; threads_cscs=0; thread_cscs=0
18 call MPI_GET_VERSION(cscs_mpi_v, cscs_mpi_subv, ierr)
19 call MPI_COMM_SIZE (MPI_COMM_WORLD, size, ierr)
20 call MPI_COMM_RANK (MPI_COMM_WORLD, rank, ierr)
21 call MPI_Get_processor_name(namecn, len, ierr)
22
23 !$omp parallel private(thread_cscs)
24 threads_cscs = omp_get_num_threads()
25 thread_cscs = omp_get_thread_num()
26 write (6,100) cscs_mpi_v, cscs_mpi_subv, rank, size, csc
27 !$omp end parallel
28
29 call MPI_FINALIZE ( ierr )
30
31
32 end program mpiomp_f
```

Locals Window:

Variable Name	Value
cscs_mpi_subv	0
cscs_mpi_v	3
cscs_omp_v	201107
namecn	node00008
rank	0
size	2
thread_cscs	0
threads_cscs	1

Input/Output Window:

```
hello in T90 mpi/3.0 0 / 2 omp/201107 0 / 1 n=node00008
hello in T90 mpi/3.0 1 / 2 omp/201107 0 / 1 n=node00009
```