Lab 1

Programming Environment, File Systems, Modules, Batch System

TACC

LAB I: Program Environments

- login to the Sun Constellation system (ranger) using your train## login:

  ssh -X train##@ranger.tacc.utexas.edu

- Untar the file lab1.tar file (in ~train00) into your directory:

  tar xvf ~train00/lab1.tar

- Change into the lab1 directory:

  cd lab1

Programming Environment

1.) Look at your Environment Variables (e.g. $WORK, $ARCHIVE,…).
2.) Discover which file systems are Lustre file system and local file systems.
3.) Take a look at the default files in your home directory.
4.) Determine the version of the operating system you are using
5-7.) Cd to $WORK using the cdw alias; Determine who owns the directory; determine the value of the $WORK variable (execute “pwd”).

```
1  env | more
2      df -h
3      ls –la
4      uname –a
5      cdw
6      ls -ld
7      echo $WORK
```

Programming Environment

1.) Check your quota:

```
lfs quota -u <username> /share
lfs quota -u <username> /work
lfs quota -u <username> /scratch
```

2.) What do you think all the output means?

File Systems – scp vs rcp

Cd to the copy directory: **cd $HOME/lab1/copy**.
Time the amount of time it takes to copy a 50MB file from your home directory to ranch.tacc.utexas.edu using “scp” and “rpc”.

rcp <file> ${ARCHIVER}:${ARCHIVE}
scp <file> ${ARCHIVER}:${ARCHIVE}

Use “/usr/bin/time” on Linux machines.
Use the “do_cp” command to perform these operations.

- **Create an executable that will generate a large file:**

  make mkfile

- **Execute the mkfile command to create a 50MB file:**

  mkfile

File Systems – scp vs rcp

- Linux (ranger)
  - lslogin2$ /usr/bin/time –p /usr/bin/rcp fort.7 ${ARCHIVER}:$ARCHIVE
  - lslogin2$ /usr/bin/time –p scp fort.7 ${ARCHIVER}:$ARCHIVE
- You should observe about a 1:5 ratio in wall clock times

- You can use the do_cp script to run both cases:
  - do_cp
- Remove the executables, object files, and fort.7 when finished:
  - make clean

Modules

- List the arguments available in the module command [execute “module”]
- List the modules that are presently loaded [module list].
- List the modules that are available [module avail].
- Determine which mpicc is being used
  (switch to mvapich-devel MPI and Intel Compiler)

```
login3%   module
login3%   module list
login3%   module avail
login3%   which mpicc
```

```
login3%   which mpicc
login3%   module unload   mvapich2
login3%   module   swap   pgi intel
login3%   module    load   mvapich-devel
```

SGE Batch (ranger)

- Cd back to **cd $HOME/lab1/batch**.
- Compile the simple “hello world” fortran MPI code

```
mpif90 –O3  mpihello.f90
mpicc  –O3  mpihello.c
```

- Look over the “job” script, and submit the program to LSF batch:

```
qsub  job
```

- Watch the status of your job and see the details (-f):

```
qstat
qstat –f –j <jobid>
showq
```

- Now, put a “sleep 60” statement in the jobs script, resubmit it, & delete the job:

```
vi job …
qsub  job  {observe the returned jobid or determine it from bjobs}
qdel  jobid
```

Precision

Look over the precision.f program in the precision dir.;
cd $HOME/lab1/precision.

Note: The sin of pi should be identically zero. The pi constant uses “E” format in one case and “D” in the other.It makes a difference! Compile precision.f and compare the results of the two sin(pi) calculations.

- **module unload mvapich2**
  module swap pgi intel
  module load mvapich2
- **Ranger**

  login3% ifort –FR precision .f (or)

  login3% ifort precision.f90

  login3% ./a.out

(The ifc compiler regards “.f” files as F77 fixed format programs. The –FR option specifies that the file is free format.)

Makefiles

- Cd down to the **using_makefiles** directory. Read over the Makefile file. The include file automatically defines the compilers and loader.

  Flags used for the system are defined by the make macro ${SYSTEM}. The compilers are Intel “ifort” or PGI “pgif77/90”; the –FR flag is used for the Intel compiler.

- Create a new a.out [make].
- Put a new access time on suba.f (Or edit the file, and save results).
- Now execute make again. Note: Only suba.f is recompiled. Unmodified .f files are not compiled. A new a.out is generated.

  Example:

  cd using_makefiles
  make
  touch suba.f (or “vi suba.f*”, change a statement and save it.)
  make

Library Generation

Generate a library from the subroutine with the ar command
[ar –ruv <lib_name> <object_files>].

Now use the library to make a new a.out.
Compile the main program and load the library [<compiler> prog.f libsub.a].

- **Linux Cluster**
  login3% ar -rv libsub.a suba.o subb.o
  login3% ifort -FR –O3 prog.f libsub.a
  login3% ./a.out

{execute “ifort --help” to discover what the “-FR” option means.}