

Beginner's Guide for UK IBM systems

This document is intended to provide some basic guidelines for those who already had certain programming knowledge with high level computer languages (e.g. Fortran, C, C++, etc) but are new to high performance computing(HPC), or have trouble to make their parallel application(s) run, or have concerns about the performance of their application(s) on the IBM systems.

The idea for that is to give the beginners a whole picture about what skills are required and where to get started. Since there are many aspects involved in HPC, it is expected that the users should obtain the detailed information from other documents for any specific aspect. I will keep revising or updating this document in the future. If you have any concerns, comments or suggestions about anything in this document, please feel free to contact me (Qiyong Jiang) at qiyongjiang1@uky.edu.

Any documents will be useful to users to some extent, but the most useful thing for HPC users is experience for which each user has to gain it by himself. It is for sure that users will still have lots of issues or need help in figuring out some error messages especially at the beginning . In any cases, please don't hesitate to seek help from the CCS staff especially the CI specialists.

1. “What I need to know before I start my work on the IBM systems?”

(1) Obtain an account and access to the IBM systems at UK

First of all, you will need an account on the systems. You can apply for an account by sending mail to sysadmin@ccs.uky.edu or go to <http://hpc.uky.edu/Accounts/> and apply for an account. Once your account is established, you will receive a username and password for you to access to the systems.

Currently there are two IBM clusters at UK. One is IBM p5-575 cluster and the other IBM HS21 BladeCenter cluster (at UK, it is also called as BCX cluster, or Intel commodity cluster).

The p5-575 cluster has 8 p575 nodes with 16 processor cores per node (see Table 1 for more information). The IBM POWER5 processor core supports both Simultaneous Multi-Threading (SMT) and Single Threaded (ST) operation modes. In the SMT mode, one physical processor core may have two processes running. In other words, users can treat the 16-core p575 node as 32-core node for your applications in the SMT mode, which is currently the default mode. To find out the basic hardware configuration, you can go to /proc:

e.g.:

```
cd /proc
```

```
cat meminfo for memory/cache information
```

```
cat cpuinfo for cpu information
```

The HS21 BladeCenter cluster has 340 blades with 4 processor cores per blade (see Table 1 for more information).

Once you get an account, you can access to the HS21 BladeCenter through the login node (bcx.uky.edu) with ssh. There are many SSH Client available on internet for free. Download and install one for the system from where you are going to login bcx.uky.edu. If your system is WINDOWS, you can install PuTTY (www.putty.org), or SSH Client Shell (www.ssh.com). If your system is Linux, you can install OpenSSH (www.openssh.com).

Some basic SSH Client commands that you may find useful:

(a) To identify the ssh client that you are currently running and its corresponding version number:

```
ssh -V
```

(b) To login to remote host (as an example, bcx.uky.edu):

```
ssh -l username bcx.uky.edu
```

(Note: For the first time login, you may need to create a host key. You can do that by running the following command:

```
ssh-keygen -F publickey.pub
```

The second time when you do login, it will prompt only for the password.)

(c) File transfer to/from remote host:

```
scp source destination
```

e.g. copy a file from the localhost to your remote bcx.uky.edu account:

```
scp localhostfile.txt username@bcx.uky.edu:/home/username
```

You can also ssh from the BCX cluster to the p575 cluster.

```
ssh plogin
```

Table 1. The hardware and software configurations for the two IBM clusters

System	p575 cluster	HS21 BladeCenter
Number of nodes	8	340
Host names:	pnode1, pnode2,..., pnode8	node1, node2,..., node340
Processor	IBM Dual Core POWER5+ @ 1.9 GHz	Intel Dual Core Xeon 5160 @ 3.0 GHz
Number of processor cores	16 per node	4 per blade
Interconnect	IBx4 SDR, Gigbit Ethernet	IBx4 SDR, Gigbit Ethernet
Memory	64 GB per node	8 GB per blade
Operating System	SLES 9	SLES 9
Kernel	2.6.5-7.244-pseries64	2.6.5-7.244-smp
MPI	poe 4.3.0.2	OpenMPI 1.2.7 (default), IntelMPI 3.0, 3.1
Compilers	IBM Fortran compiler – XLF10.1 IBM C/C++ compiler – VACPP 8.1	Intel Fortran compiler 10.1.015 Intel C/C++ compiler 10.1.021

(2) Linux and Shell commands

There are different flavors of Linux. SLES (SUSE Linux) is just one of them. Most commands are the same for different Linux systems. If you had experience with Unix system before, that also will help.

A shell enables the user to interact with resources of the computer such as programs, files and devices. The default shell on the systems is Bourne shell.

There are a lot of useful tools and commands provided by shells. I suggest you to get familiar with (if you haven't) the following ones first. If you need detailed information about any tool or command, please refer to the specific manuals.

vi or emacs - built-in text editors. You will need them to edit your files. So, that's very important. You can find a list of basic vi commands here:

<http://www.cec.mtu.edu/newuserdoc/node9.html>

cd - change a directory

mkdir - create a directory

rm - delete a file

ls - list files

cp - copy files

mv - mv files

cat

more

pwd

find

grep

tar

chmod

which

export

env

kill

...

(3) Compilers

On the p575 cluster, you will need to use IBM compilers for Fortran, C and C++:

xlf - for Fortran77

xlf90 - for Fortran90

xlc - for C

xlc - for C++

There are some variations for the compilers. You can get a lot of information especially the available options from the compiler manuals or simply type the following commands:

```
xlf -help  
xlc -help
```

e.g.:

```
xlf -O3 -o MyApplication MyApplication.f
```

where -O3 is for level 3 optimization and -o specifies the executable name.

On the HS21 BladeCenter cluster, the default is Intel compiler:

ifort – for Fortran77/90

icc – for C

icpc - for C++

It will be very helpful for you to get familiar with those options if you just try some of them with simple applications.

(4) Parallel Computing and MPI

The purpose of parallelization is to reduce the time spent for computation. Message-passing is the tool to consolidate what parallelization has separated.

If you are new to MPI, you can easily find some tutorial material from the internet for you to get familiar with (for example, <https://computing.llnl.gov/tutorials/mpi/>).

There are different flavors of MPI implementations, such as OpenMPI, IntelMPI, poe (IBM MPI), HP MPI, MVAPICH, etc.

On the p575 cluster, it is IBM MPI, which is usually called poe (Parallel Operating Environment). The compiler wrappers are:

mpfort for Fortran (specifically, mpfort -qfixed for F77; mpfort -qfree for F90)

mpcc for C

mpCC for C++

poe for running the executables

On the HS21 BladeCenter cluster, the default is OpenMPI, but you can also use IntelMPI as another option. The compiler wrappers for OpenMPI are:

mpif77 for F77

mpif90 for F90

mpicc for C

mpiCC for C++

mpirun for running the binary

After you login to the systems, you can find all options for compilers and MPI implementations with the following commands:
module avail

You can load anyone from the list:
module load XXX

You can also unload the one already loaded:
module unload XXX

(5) SLURM

It is required by our system administrators that all jobs have to be submitted through a job scheduler. SLURM (Simple Linux Utility for Resource Management) is the one for the systems. Please check this out regarding SLURM and other software available at UK systems: <http://hpc.uky.edu/Software/>. You can use:

srun for submitting a job (tip: run "srun --h" to get a list of options)
scancel for terminating a pending or running job (tip: run "scancel --h" to get a list of options)
squeue for monitoring job queues (tip: run "squeue --h" to get a list of options)
sinfo for monitoring partition and overall system state (tip: run "sinfo --h" to get a list of options)

The following is a brief demo with a very simple MPI example to show the processes of creating, compiling and submitting a job.

(a) Creating a Fortran file with vi

The command you will use is: vi trivial.f

Then type in the content for the file:

```
PROGRAM MAIN
IMPLICIT NONE
INCLUDE 'mpif.h'
INTEGER TASK_COUNT, TASK_RANK
INTEGER IRC
CALL MPI_INIT (IRC)
CALL MPI_COMM_SIZE (MPI_COMM_WORLD, TASK_COUNT, IRC)
CALL MPI_COMM_RANK (MPI_COMM_WORLD, TASK_RANK, IRC)
PRINT *, 'Task ',TASK_RANK,' of ',TASK_COUNT,' tasks started'
CALL MPI_BARRIER (MPI_COMM_WORLD, IRC)
CALL MPI_FINALIZE (IRC)
STOP
END
```

(b) Compiling a Fortran file

Assuming you are using the default openmpi-1.2.7 compiled with Intel compiler, the command you will use is:

```
mpif77 -O3 -o trivial_f trivial.f
```

where mpif77 is the openmpi Fortran compiler wrapper. The file trivial.f is compiled with -O3 optimization level and the binary with name trivial_f will be created (if the compiling is successful.)

(c) Submitting a job to BCX

First create a file -- run_mpirun with only one line in this file:

```
mpirun -np 2 trivial_f
```

which will run trivial_f on two cpus.

To make this file executable: `chmod +x run_mpirun`

To submit run_mpirun to BCX with SLURM"

```
srun -b -N 1 ./run_mpirun
```

where -b means it's a batch job and -N specifies the number of nodes.

(d) Checking the results

If the running is successful, the program should return with the following results:

```
Task 0 of tasks started
```

```
Task 1 of tasks started
```

WARNING: ONLY LAUNCH JOBS USING SLURM AND THE BATCH SCHEDULER. NO RUNNING JOBS ON LOGIN NODES. OTHERWISE, YOUR ACCOUNT MAY BE SUSPENDED!!!

2. "I see a lot of error messages in compiling/linking/running my application!"

It is very rare (or almost impossible) that someone can develop a real application from scratch and make it run correctly without encountering any problems. So, when you see error messages during any stage, don't panic! Sometimes, hundreds of error messages just have one root cause. The rule of thumb is to dig out the root cause of the first error message first. If you are lucky enough, all the rest of errors may be gone after you fix the first one. There are different kinds of errors based on different stages:

(1) Compiling errors

It is very common that users write codes with syntax errors. Most compilers will point out the lines and sometimes even the types of syntax errors during compiling. So, these are relatively easy errors for users to fix. Of course, some errors like "Internal compiler error" absolutely leave no clue for users to track down.

(2) Linking errors

Linking errors are often caused by missing subroutines or functions which are supposed to be linked with. If an error message says “ERROR: Undefined symbol: XXX”, first of all you need to find out whether XXX is supposed to be from the application package or be provided by an external library. For the former case, the user should check whether some files were not correctly compiled or some symbols (names of functions or subroutines) were changed by compilers especially for mixed-language (e.g. C, Fortran) applications. For the latter case, you should check whether the path to the library is missing or incorrectly set.

(3) Runtime errors

There are many different types of runtime errors. A very common runtime error is “Cannot find `***.so`”. That usually means some shared runtime library is not available. Check the environment variable `LD_LIBRARY_PATH` to make sure the path to the shared library is correctly set. The runtime library requirements of an executable can be checked with the `ldd` utility.

(4) Wrong results

If you are sure that the correct input file is used and the application finishes running normally but the results are not correct, the first thing you can try is to lower your optimization level and recompile and rerun the application. If the problem persists, you may need to apply some debug tools (e.g. TotalView) to analyze the cause or even report the problem to the application developer.

(5) System problems

The systems or any part of the systems can go wrong for some reasons. If you suspect any system problem, please report it to Help at help-hpc@uky.edu. Just make sure that the problem you are experiencing is not from your application side before you report.

3. “How do I make my application run faster?”

This is the most sophisticated part in HPC practice and is an endless effort. If you are just at the very beginning of parallel programming and already feel overwhelmed to get your application work, you probably should skip this part for the time being.

If you are really concerned about your application performance, here are the steps you can consider:

- (1) Try different optimization level or different option sets.
- (2) Tune runtime environment variables available.
- (3) Compare the performance among different compilers or MPIs or the combinations.
- (4) Link with tuned or optimized performance libraries if possible.

e.g.

If your application uses BLAS routines, you can link your application with IBM ESSL library (on p575 cluster) or Intel MKL (on HS21 BladeCenter cluster).

(5) Do profiling and identify “hot spots” and apply appropriate hand-tune strategies.