

PPCES2010 - MPI Introduction

Lab Exercises
March 22, 2010

{anmey|bierbaum|iwainsky|kapinos|reichstein|schmidl|wienke}@rz.rwth-aachen.de

Part 1: MPI Basic Exercises on the Virtual Machine

To start the VM on the laptop, double-click on the **VMware** icon on the desktop. Click on the *SUS_HPC_Kurs_1* tab and then on the green arrow to power on this virtual machine. Log in with “**hpclab**” as user name (the password will be announced during the lab).

You can exchange data between the Laptop’s Windows 7 host system and the VM with a shared folder, which is mounted in the VM at **/mnt/hgfs/shared/PPCES2010**.

You can also access the cluster from within the VM and exchange data with **ssh** and **scp** commands. The examples are prepared to work with the Sun compiler and the Sun MPI library on the VM. You don’t need to use any module commands there.

Click on the desktop icon labeled with *Terminal* to open a console.

Extract the MPI exercises into your VM home directory from either the shared folder

```
cd /home/hpclab
gtar xf /mnt/hgfs/SharedFolder/PPCES2010/MPI/mpi_intro.tar
```

or from the cluster

```
cd /home/hpclab
scp hpclabXZ@cluster-linux:/home/hpclab/PPCES2010/MPI/mpi_intro.tar .
gtar xf mpi_intro.tar
```

Exercise 1

Change into the directory `simple/c` or `simple/f` depending on your preference for C or Fortran

```
cd ~/mpi_intro/simple/X
```

Compile, link and execute program “simple” with 4 MPI processes

```
gmake clean
gmake build
gmake go
```

Repeat the execution several times. What do you observe?

Exercise 2

Modify this program “simple”, such that the master (rank 0) reads a positive integer number *n* from standard input and sends it to all slaves (rank 1 ... rank *np*-1)

Exercise 3

Change into the directory ring/c or ring/f depending on your preference for C or Fortran

```
cd ~/mpi_intro/ring/X
```

Modify this program, such that the master (rank 0)

initializes an array of n 8-byte real numbers with values 1.0, 2.0, ..., n .

Then this array shall be passed around all processes from rank 0 to rank 1 to rank 2, ... to rank $np-1$, and then back to rank 0 like in a ring.

Exercise 4

Modify this program, such that the total sum of all these numbers of all process is printed out by the master.

Check the result: $np * (1.0 + 2.0 + \dots + np) = np * n * (n + 1) / 2$

Exercise 5

Change into the directory fix_me/c or fix_me/f depending on your preference for C or Fortran

```
cd ~/mpi_intro/fix_me/X
```

The program contains a bug. What is the symptom? What is the problem?

```
gmake clean
```

```
gmake build
```

```
gmake go NPROCS=2
```

or use the TotalView debugger (does not currently not work for the C version in the VM)

```
gmake tv
```

Exercise 6

Change into the directory pi/c or pi/f depending on your preference for C or Fortran

```
cd ~/mpi_intro/pi/X
```

Improve the program pi, which contains a very stupid parallel approach of parallelizing a numerical integration.

Exercise 7a

Change into the directory jacobi/c or jacobi/f depending on your preference for C or Fortran

```
cd ~/mpi_intro/jacobi/X
```

The program solves a finite difference discretization of the Helmholtz equation:

$$(d^2/dx^2)u + (d^2/dy^2)u - \alpha u = f$$

using the Jacobi iterative method.

The parallelization of the subroutine jacobi is very dangerous. Why?

Change the exchange of the ghost cells using

a) **MPI_Sendrecv**

b) asynchronous receive operations **MPI_Irecv**

c) asynchronous receive and asynchronous send operations **MPI_Irecv**, **MPI_Isend**

In order to continue your work on the cluster under Linux, you can pack and transfer your solutions

```
cd ~
```

```
tar cf my_mpi_lab.tar mpi_intro
```

```
scp my_mpi_lab.tar hpclabXZ@cluster-linux
```

Part 2: MPI Exercises on the Linux-Cluster

(find the MPI Exercises for Windows on the Windows HPC and Visual Studio Lab Sheet)

Login into the Linux-Cluster with your lab account **hpclabXZ** using the NX-frontend machines:

```
cluster-x.rz.rwth-aachen.de  
cluster-x2.rz.rwth-aachen.de
```

Open a new connection by double-click on the NX icon on the Desktop and enter the hostname, user name (login) and password.

For your first login with NX you need to click on *Configure* to perform some advanced tuning. You should specify the expected bandwidth of your network connection (*ADSL*). We recommend to disable the direct draw option: ***Configure --> Advanced --> Disable direct draw for screen rendering***

After you started an NX session, you should open a terminal and log in to the lab machine (where Z is the rightmost digit on your badge):

```
ssh -l hpclabXZ labZ
```

Either extract the original MPI exercises into your cluster home directory

```
cd ~  
gtar xf /home/hpclab/PPCES2010/MPI/mpi_intro.tar .
```

Or continue work on the files which you transferred from the VM previously

```
cd ~  
gtar xf /home/hpclab/PPCES2010/MPI/my_mpi_lab.tar .
```

The examples are prepared for the Sun Studio compilers and Sun MPI library.

Use the **module** command to check and prepare your environment.

Using the Batch System

Long-running computations and computations which must run on empty machine to minimize side effects (e.g. benchmarks) should be submitted to the batch system (Oracle Grid Engine, formerly known as SGE). A batch script must be submitted using the **qsub** command (or alternatively using the **qmon** GUI which is not covered here). A batch script is basically a shell script containing all commands which must be executed, and all (recommended) necessary options for the batch system. It is also possible to provide all options as qsub command line parameters. An example batch script is the *submit.sh* file. Note, that the batch script can (and should!) be tested interactively before submitting. The interactive test lets you directly find eventual errors omitting the batch system waiting time. To let the batch file run interactively, it must be marked as “executable” by the command:

```
chmod 755 submit.sh
```

and then execute it by

```
./submit.sh
```

After you have tested your script you can submit it to the batch system:

```
qsub submit.sh
```

The batch system tells you the job ID of the submitted job:

```
Your job xxxxxx ("submit.sh") has been submitted
```

The status of all your jobs can be controlled by the **qstat** command. Only waiting jobs and jobs in an error state are shown; finished jobs are not listed.

You can cancel a job with the **qdel** command:

```
qdel xxxxxx
```

where xxxxxx is the job ID.

Exercise 7b

Which of the solutions of exercise 7a runs best on the HPC cluster?

Read the input data from file **input.big**

Exercise 8

Change into the directory hybrid/c or hybrid/f depending on your preference for C or Fortran

cd ~/mpi_intro/hybrid/X

The program jacobi solves a finite difference discretization of the Helmholtz equation:

$$(d^2/dx^2)u + (d^2/dy^2)u - \alpha u = f$$

using Jacobi iterative method. The parallelization employs a combination of MPI and OpenMP.

Compile and start the program interactively and in batch mode.

Do first tests with the small input data set in file **input** and then with the larger data set in file **input.big**.

What is the optimal number of MPI processes and OpenMP threads per MPI process when running the program on two nodes?

Exercise 9

Like Exercise 8. Use the Intel compiler and compare the performance.