Technische Universität Berlin

School IV – Electrical Engineering and Computer Science
Chair Communication and Operating Systems (KBS)
http://www.kbs.tu-berlin.de

Integrated Course

Parallel Programming

in Summer Term 2013

J. Schönherr

Assignment 1

Issued: Wednesday, 17th April 2013

Due: Wednesday, 24th April 2013

Information

Please upload your solution via the ISIS page of this course until 23:55 of the due date. Your upload should consist of a PDF document covering the theoretical parts and a zip/tar.gz archive with your source files. Please list the names and matriculation numbers of all group members inside your uploaded documents.

Exercise 1 – Getting past technical problems

Get familiar with our infrastructure. Some documentation is available in the Parallel Programming Wiki hosted on ISIS.

Ultimately, log on to the TuSCI front-end and execute the following script:

```
$ i_am_logged_in
```

(Everyone should execute this script.)

Exercise 2 – MPI warm-up

Some MPI related resources are linked from within the Parallel Programming Wiki.

(a) (optional) Hello World

(Exercise 2a is optional. It helps you getting started, if you do not like to read MPI tutorials.)

Write, compile and run a very simple MPI program:

Every participating process should print a message, displaying its rank and the total number of processes.

MPI functions, that you will probably need:

`MPI_Init`, `MPI_Comm_rank`, `MPI_Comm_size`, `MPI_Finalize`

(b) A game of Telephone

Write, compile and run an MPI program, that performs a little game of Telephone.

The program should take a string as argument from the command line. The first process (with rank 0) sends this string to the second process, the second to the third and so on. The last process displays the string.

You may use a fixed message size. But still, pay attention to corner cases!

MPI functions, that you will probably need:

`MPI_Send`, `MPI_Recv`

Exercise 3 – Collective communications

In Exercise 2b you have implemented a variant of a broadcast operation: before the (series of) communication only one node knows the data, after the communication all nodes are aware of it.

(a) Influence of the interconnection network

Assume that each MPI process is executed on its own node and that the interconnection network has a certain node to node latency and a certain bandwidth between a node and the interconnection network. How long does it take (ignoring computation) until all processes of Exercise 2b are aware of the message?

p	number of processes
n	size of message in bytes
α	latency in seconds
β	bandwidth/datarate in bytes per second

(b) Other ways to broadcast

The broadcast in Exercise 2b is rather inefficient as most nodes do nothing most of the time.

Describe multiple ideas to realize the broadcast more efficiently (using only send/receive). Analyze each idea as in Exercise 3a and compare them. Is there a "best" variant to broadcast, or do we need different algorithms for, e. g., different message sizes?

(c) MPI collectives

Luckily, MPI already defines a set of collective communications so that you do not have to care about the concrete realization that much.

Reimplement the application using exactly one call to `MPI_Bcast` instead of send and receive!

Exercise 4 – Measurements (optional)

Try to actually measure the latency and bandwidth of TuSCI's interconnection network. Try out NMPI as well as OpenMPI. (The pool workstations are also interesting targets.)

Discuss ways to do this as well as your results in the discussion forum in ISIS.

Functions, that you might need:

`MPI_Wtime` (or `clock_gettime` to be independent of MPI)