Assignment No. 3
Parallel Computing, DM8XX (Fall 2008)

Department of Mathematics and Computer Science
University of Southern Denmark
Daniel Merkle

Due on: Wednesday 5. November, 12:00 p.m. (Department secretaries office (Lone Seidler Petterson) or my office).

Exercises or parts of exercises marked with * are voluntary exercises.

Exercise 1

MPI Ping Pong (0+10+5+5=20 points)

In this exercise you will familiarize with basic MPI send/receive operations. In the directory `/home/daniel/assignment3/` you will find a program `helloworld.c` that you have to extend for this exercise. You can compile the program with the command `mpicc helloworld.c -o helloworld`. The program can be started with the command `mpirun`. For example `mpirun -np 2 --host logon1,logon2 ./helloworld` starts 2 copies of the program `helloworld`, one on node `logon1` and one one node `logon2`.

The operations for sending and receiving messages that we will use in this exercise are:

```
int MPI_Send(void *buf, int count, MPI_Datatype dtype, int dest, int tag, MPI_Comm comm)
int MPI_Recv(void *buf, int count, MPI_Datatype dtype, int src, int tag, MPI_Comm comm, MPI_Status *stat)
```

`MPI_Send` will send a message to a process specified with the identifier `dest`. The initial address of the buffer to be send is `buf`, `count` is the number of elements in the send buffer, and the datatype of each send buffer element is `dtype`, (for example `MPI_DOUBLE`). The message is received by the corresponding `MPI_Recv` command. With the variable `tag` a message receives a tag, that has to be identical for the `MPI_Send` and `MPI_Recv` command, in order that the message is received. As *communicator* `MPI_COMM_WORLD` will be used. (We will discuss communicators when we cover Chapter 6 of the course book).

For measuring execution times you should use `MPI_Wtime()`. The usage of `MPI_Wtime()` is straightforward (see the source code `helloworld.c`). Furthermore, the function `double pruned_average(double *time, int n, double alpha)` (to be found in the files `timing.c` and `timing.h`) can be used to compute a pruned average of values. More precisely, the n `double` values stored at `*time` are used to compute an average that discards the "smallest" and "largest" values in the following way. When `alpha` is set to 0.25, then the 25% smallest and 25% largest `double` values of are discarded and with all other values the average is computed. Use `alpha`=0.25 in this exercise to eliminate outliers.

a) *Extend the program `helloworld.c` such that you are able to measure the execution time of a communication operation. For this process 0 should send a message to process 1, and then process 1 should send a message back to process 0. Repeat this ping-pong operation several times and compute the average runtime of a communication operation using `pruned_average`. Use two physically different machines for the experiments. What bandwidth corresponds to the communication time you measured?

b) Measure the execution time for sending and receiving for messages of size 2^0, $2^1, \ldots, 2^{20}$ `double` numbers. Use two physically different machines for the experiments. For each message size repeat this ping-pong operation, such that the overall data that is transfered is approximately similar. Use `pruned_average` to compute the average communication time and the average bandwidth. Plot the communication time and the bandwidth (in Mbit/s) in a figure which has the message size as the x-axis.

c) Use your results to estimate the parameters t_s (startup time) and t_w (per word transfer time).

d) Repeat b) and c) using the two cores of one machine. (`mpirun -np 2 --host logon1,logon1 ./pingpong` will start two processes on the machine logon1.)

Exercise 2

Circular q-shift (5 points)

Show that in a p-node hypercube, all the p data paths in a circular q-shift are congestion-free if E-cube routing (Section 4.5 of the course book) is used.

Hint: (1) If $q > p/2$, then a q-shift is isomorphic to a $(p - q)$-shift on a p-node hypercube. (2) Prove by induction on hypercube dimension. If all paths are congestion-free for a q-shift $(1 \leq q < p)$ on a p-node hypercube, then all these paths are congestion-free on a $2p$-node hypercube also.

Exercise 3*

Circular q-shift

Show that the length of the longest path of any message in a circular qshift on a p-node hypercube is $\log p - \gamma(q)$, where $\gamma(q)$ is the highest integer j such that q is divisible by 2^j.

Hint: (1) If $q = p/2$, then $\gamma(q) = \log p - 1$ on a p-node hypercube. (2) Prove by induction on hypercube dimension. For a given q, $\gamma(q)$ increases by one each time the number of nodes is doubled.

Exercise 4

All-to-all Broadcast on a tree (5 points)

Given is a balanced binary tree (cmp. Figure 4.7., page 155 of the course book), where only the leaves of the tree contain computing nodes.

a) One possibility to perform an all-to-all broadcast on the tree is the embedding of the all-to-all broadcast algorithm for a ring onto a corresponding tree. Is this possible without congestion (i.e. no two messages travelling in the same direction share one communication link), and — if possible — what is the runtime of this all-to-all broadcast algorithm?

b) Describe a procedure to perform an all-to-all broadcast that takes time $(t_s + t_w \cdot m \cdot p/2) \log p$ for m-word messages on p nodes. Assume that an exchange of two m-word messages between any two nodes connected by bidirectional channels takes time $t_s + t_w \cdot m \cdot k$ if the communication channel (or a part of it) is shared by k simultaneous messages.

Exercise 5

Amdahl's law (5 points)

If a problem of size W has a serial component W_S (i.e. the part of a program that can to be parallelized), prove that W/W_S is an upper bound on its speedup, no matter how many processing elements are used.

* Read on Gustafson's law (check for example wikipedia), that addresses shortcomings of Amdahls's law.

Exercise 6

Scalability (4+5+5+2+2+2=20 points)

Assume the following hypothetical overhead function for an algorithm (as usual W denotes the problem size).

$$T_O = p^2 \cdot \sqrt{W} + p \cdot \sqrt{W}$$

Assume that maximal degree of concurrency of the algorithm is $2^{\sqrt{W}}$.

a) Determine the parallel computation time T_P (as a function in W and p).

b) Determine the isoefficiency function due to the overheads T_O.

c) Determine the isoefficiency function due to the maximal concurrency.

d1) Determine the number of processes p', for which the parallel runtime is minimal.

d2) Determine the runtime when using p' processes (cmp. d1).

d3) Determine the asymptotic efficiency (as a function in W) when using p' processes. What is the efficiency for arbitrary large problem sizes W when using p' processes?