Tobias Dreher 11.09.2012

TDT 4200 Problem set 2

Task 1

a)

SISD:

Single Instruction, Single Data
Executes one instruction at a time and can fetch or store one item of data at a time
Example: Sequential working algorithm like Backtracking

SIMD:

Single Instruction, Multiple Data
Applying one instruction at a time to more than one item of data at a time
Example: Parallelized loops or vector processing

MISD:

Multiple Instruction, Single Data
Applying different instructions to one item of data
Example: Search the shortest/best path in an unknown graph

MIMD:

Multiple Instruction, Multiple Data
Different instructions works with different data
Example: (Independent) processes in an operation system

b)

- You can split large data in smaller pieces to send them
- You can override the system buffer policy because outgoing messages are buffered in user defined spaces

c)

Less code means less errors. In this case you avoid the danger of a dead lock.

d)

Deadlock:

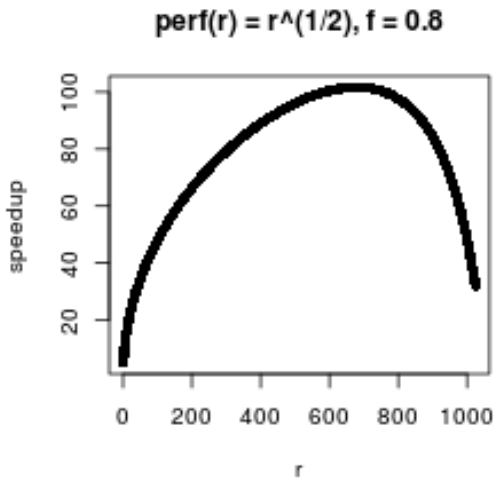
```
// rank 0
MPI_Ssend(&dataA, 1, MPI_INT, 1, 99, MPI_COMM_WORLD;
MPI_Recv(&dataB, 1, MPI_INT, 1, 99, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
// rank 1
MPI_Ssend(&dataB, 1, MPI_INT, 0, 99, MPI_COMM_WORLD);
MPI_Recv(&dataA, 1, MPI_INT, 0, 99, MPI_COMM_WORLD MPI_STATUS_IGNORE);
```

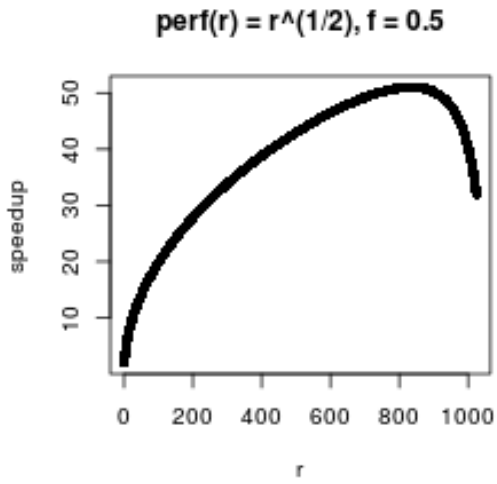
Correct:

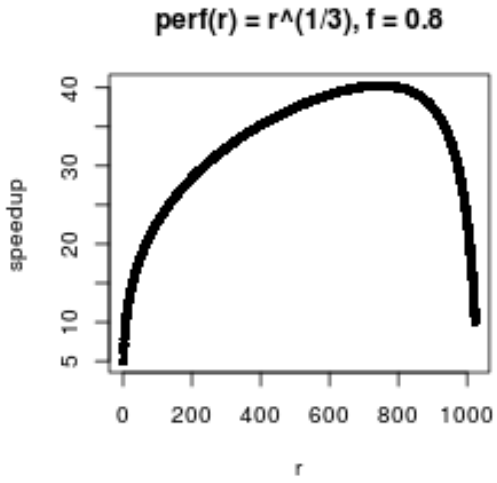
```
MPI_Request req[2];
MPI_Irecv(&dataB, 1, MPI_INT, 1, 99, MPI_COMM_WORLD, req);
MPI_Irecv(&dataA, 1, MPI_INT, 0, 99, MPI_COMM_WORLD, req + 1);
MPI_Send(&dataA, 1, MPI_INT, 1, 99, MPI_COMM_WORLD);
MPI_Send(&dataB, 1, MPI_INT, 0, 99, MPI_COMM_WORLD);
MPI_Waitall(2, req, MPI_STATUSES_IGNORE);
```

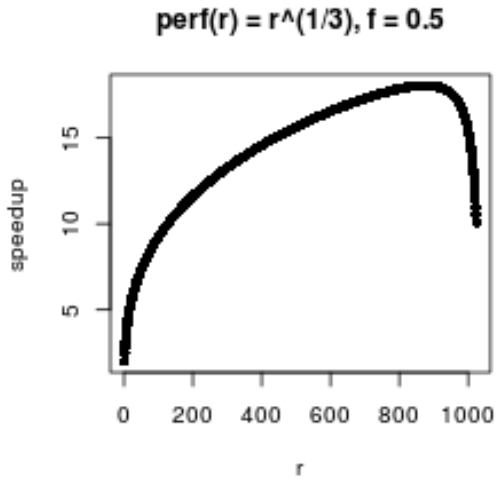
Task 2

a) r = 678, speedup = 101.72
b) r = 835, speedup = 51.03
c) r = 748, speedup = 40.26
d) r = 875, speedup = 18.04









Tobias Dreher 11.09.2012

Task 3

a)

First step: Processors with odd ranks communicate with the processors right of them in full duplex.

$$t_1 = 50\,ms + \frac{1024\,bytes}{1\,\frac{bytes}{ms}} = 1074\,ms$$

Second step: Processors with odd ranks communicate with the processors left of them in full duplex.

$$t_2 = 50\,ms + \frac{1024\,bytes}{1\,\frac{bytes}{ms}} = 1074\,ms$$

$$t = t_1 + t_2 = 2148\,ms$$

b)

First step: Processors with odd ranks communicate with the processors right of them in full duplex.

$$t_1 = 50\,ms + \frac{1024\,bytes}{2 * 1\,\frac{bytes}{ms}} = 562\,ms$$

Second step: Processors with odd ranks communicate with the processors left of them in full duplex.

$$t_2 = 50\,ms + \frac{1024\,bytes}{2\,\frac{bytes}{ms}} = 562\,ms$$

Third step: Processors in the upper half communicate with the processors under them in full duplex.

$$t_3 = 50\,ms + \frac{1024\,bytes}{8 * 1\,\frac{bytes}{ms}} = 178\,ms$$

$$t = t_1 + t_2 + t_3 = 1302\,ms$$

c)

First step: Processors with odd ranks communicate with the processors right of them in full duplex.

$$t_1 = 50\,ms + \frac{1024\,bytes}{4 * 1\,\frac{bytes}{ms}} = 306\,ms$$

Second step: Processors with odd ranks communicate with the processors left of them in full duplex.

$$t_2 = 50\,ms + \frac{1024\,bytes}{4 * 1\,\frac{bytes}{ms}} = 306\,ms$$

Third step: Processors in the top row and in the third row communicate with the processors under them in full duplex.

$$t_3 = 50\,ms + \frac{1024\,bytes}{4 * 1\,\frac{bytes}{ms}} = 306\,ms$$

Fourth step: Processors in the second row and in the fourth row communicate with the processors under them in full duplex (fourth row communicate with the first row).

$$t_4 = 50\,ms + \frac{1024\,bytes}{4 * 1\,\frac{bytes}{ms}} = 306\,ms$$

$$t = t_1 + t_2 + t_3 + t_4 = 1224\,ms$$