

Parallel Numerics

Exercise 1: MPI (recapitulation)

During the summer term 2004 you attended the lecture on “Parallel Programming”. Part of the course dealt with MPI (Message-Passing-Interface) that can be used to parallelize programs on supercomputers or workstation clusters. This term we will use MPI for programming a couple of numerical codes. Therefore, a recapitulation of the material about MPI is subject of this exercise.

Have a short repeat of the slides (MPI1.pdf) on the site
<http://wwwbode.cs.tum.edu/gerndt/home/Teaching/SS2004/ParallelProgramming/MPI1.pdf>

and of the exercises you have done last term on this subject. The basic functions are most important (MPI_Init, MPI_Finalize, MPI_Comm_size, MPI_Comm_rank, MPI_Send, MPI_Recv). Recall their meanings!

On the site <http://www-unix.mcs.anl.gov/mpi/> you find helpful informations about all questions concerning MPI. By clicking on *MPI Standard 1.1* or *MPI Standard 2.0* you find (at the bottom of the page) a link to the *MPI 1.1 Standard Index* and *MPI 2.0 Standard Index*. There are good descriptions of all MPI functions. Have a look at this information material!

On the lecture site <http://www5.in.tum.de/lehre/vorlesungen/parnum/WS02/> you find the program `communication.c`. Download, compile and run this program on the computers you can find in room 02.05.036 on which MPI is installed. You should be able to log on to these machines; if you can't, send an email to lauter@in.tum.de, specifying precisely the problem you have. Do the following steps:

1. Be sure that you can run the MPI-compiler `mpicc` and runtime environment `mpirun`. Verify this by launching the two programs with `--help` as an argument.
2. MPI uses `ssh` to launch parallel processes on other computers. Therefore it is necessary that it can log on to these machines without the need of you typing a password. In order to get this feature working, execute the following commands on one of the computers:

```
cd
cd .ssh
ssh-keygen -t dsa
```

Just press enter on any question asked by ssh-keygen.

```
cp id_dsa.pub authorized_keys
```

Now log on to the following machines once using `ssh`:

```
ssh atzenger42
```

```
ssh atzenger27
```

```
ssh atzenger28
```

```
ssh atzenger16
```

```
ssh atzenger22
```

If you encounter any problem with this step you can't fix yourself, send me an email explaining your problem. If you merely fail to log on to one or two machines, don't worry. Just don't try to use MPI on these computers during the further steps.

3. Enter the names of the computer you were able to log on using `ssh` in a file named `machines.dat` using one line per machine, only specifying the name of the machine. Example:

```
atzenger42
```

```
atzenger27
```

```
atzenger22
```

```
atzenger16
```

```
atzenger28
```

You can use `emacs` to do so.

4. Compile and link the program you downloaded:

```
mpicc -o communication communication.c
```

5. Run the program using `mpirun` on for example 2 processors:

```
mpirun -np 2 -machinefile machines.dat communication
```

6. Try other values for the number of processes. You can use as many processes as you have entries of computers in the file `machines.dat`.¹ How does the timing of the short program you are executing change when you increase the number of processes from 1 to the maximum? You can use the utility `time` to measure the time needed. Ask yourself why you can observe such a behaviour.

Have a look at the program itself: What does the program do? Understand the program and the used functions!

¹Although MPI allows you to run even more than one process per machine, ask you why this is not worthful on our single-CPU machines!