Concurrency & Parallel Programming
DAS4 instructions

1 Accessing DAS-4

For the practical assignments you will be working on the DAS-4. During the first practical session you should have received an account per team. The DAS-4 consist of several clusters. For example, DAS-4/VU and DAS-4/UvA. You will be using the UvA cluster, which head node is named `fs2.das4.science.uva.nl`.

To access the cluster, you have to use SSH:

```
$ ssh -X USERNAME@fs2.das4.science.uva.nl
```

Where `USERNAME` should be the user account you received. The `-X` flag is optional, but enables X11-forwarding. This way, you open graphical applications through ssh.

Accessing DAS-4 directly from the lab machines should be possible.

To transfer files from your own machine to DAS-4, you can use `scp`.

```
$ scp file USERNAME@fs2.das4.science.uva.nl:
```

To synchronize files you can use `rsync`. `rsync` is a file synchronization program that minimizes network data transfer. To synchronize folder A on your machine with folder A on DAS you can run:

```
rsync -avz -ssh /home/$USER/A USERNAME@fs2.das4.science.uva.nl:/home/$USER
```

Or mount your home directory with `sshfs`.

```
$ mkdir ~/das4
$ sshfs USERNAME@fs2.das4.science.uva.nl: ~/das4
```

Now all files in your DAS-4 home directory are accessible in the directory `/das4`.

Lastly, you can also use a version control system like git, and pull your files on DAS-4.

2 Running jobs

Once on the head node, you can edit and compile your program. In order to use `prun` you should first load the `prun` module.

To see the available modules you can type:

```
$  module avail
```

To load the `prun` module type:

```
$ module  load prun/default
```

When you are ready to test your program, you can schedule your job in the queue using `prun`.

```
$ prun -v -np 2 <programname>
```

For example you can try:

```
$ prun -v -np 2 date
```

Here, `-np 2` means you want two node. Each node contains two cpu's with each 4 cores. Sometimes all nodes are busy, and your job will have to wait in the queue. You can check the queue using the command `qstat`. There are 18 nodes in the UvA cluster.

Note: If you name you program 'test' and want to execute it you should specify it's absolute path. That is because there is a command in `/usr/bin/test`. So assuming your program is in your home directory instead of typing:

```
$ ./test
```

you should type:

```
$ $HOME/test
```

3 MPI

To add MPI to your environment type:

```
$module load openmpi/gcc
```

If everything is ok you should be able to run:

```
$which mpicc
```

Using this sample named cpi.c:

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>

static double
f(double a)
{
    return (4.0 / (1.0 + a*a));
}

int
main(int argc, char *argv[])
```

```
{
    int done = 0, n, myid, numprocs, i;
    double PI25DT = 3.141592653589793238462643;
    double mypi, pi, h, sum, x;
    double startwtime = 0.0, endwtime;
    int  namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Get_processor_name(processor_name, &namelen);

    fprintf(stderr, "Process %d on %s\n", myid, processor_name);

    n = 0;
    while (!done) {
        if (myid == 0) {
    if (n == 0) {
n = 100; /* precision for first iteration */
    } else {
n = 0;   /* second iteration: force stop */
    }

    startwtime = MPI_Wtime();
        }

        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
        if (n == 0) {
            done = 1;
        } else {
            h   = 1.0 / (double) n;
            sum = 0.0;
            for (i = myid + 1; i <= n; i += numprocs) {
                x = h * ((double) i - 0.5);
                sum += f(x);
            }
            mypi = h * sum;

            MPI_Reduce(&mypi, & pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

            if (myid == 0) {
                printf("pi is approximately %.16f, error is %.16f\n",
                       pi, fabs(pi - PI25DT));
endwtime = MPI_Wtime();
printf("wall clock time = %f\n", endwtime - startwtime);
    }
```

```
        }
    }

    MPI_Finalize();

    return 0;
}
```

you can compile the code by running:

```
$mpicc -O2 -o cpi cpi.c
```

and run it using:

```
$mpirun -np 2 cpi
```

Here, `-np 2` means you want two processes to run.

you can also submit with `prun`, which is the preferred way by typing:

```
$prun -np 1 -v -2 -sge-script $PRUN_ETC/prun-openmpi ‘pwd‘/cpi
```

Here, `-np 1` means you want one node and `-V -2`, two processes to run. If you have `-np 2` and `-V -2` this means that on each node you'll get 2 processes.

4 GPU programming

GPUs on DAS-4 can be programmed using CUDA and OpenCL.

4.1 CUDA

To add CUDA 5.5 to your environment type:

```
$module load cuda55/toolkit
```

If all is ok, you should be able to run the CUDA compiler try:

```
$nvcc --version
```

For running CUDA jobs, you can use:

```
$prun -v -np 1 -native ’-l gpu=GTX480’ <programname>
```

For example you can try:

```
$prun -v -np 1 -native ’-l gpu=GTX480’ $CUDA_SDK/bin/x86_64/linux/release/deviceQuery
```

5 Remote access

If you want to work on the assignment from home, you can't directly connect to the DAS-4. In order to work from home, you have to use UvA-VPN. Instructions for all platforms can be found on the UvA site: `http://student.uva.nl/en/az/content/uvavpn/uvavpn.html`

6 More information

A more comprehensive explanation can be found on the DAS-4 website: `http://www.cs.vu.nl/das4/home.shtml`. Here you will find more information about both the machines itself, and how to use them.