# 2014 HPC Winter School

University of Stellenbosch

Practical 4

Surname

Name

University

Login number

student

**Important:** Follow these instructions carefully.

1. Login into the CHPC cluster using ssh to connect to `lengau.chpc.ac.za`
2. Make a new sub-directory called `PRAC4` and include **your surname** in its name:

   ```
   cd
   mkdir PRAC4-surname
   ```

3. Make sure I can read it:

   ```
   chmod 0755 .
   chmod 0755 PRAC4-surname
   ```

   Don't forget the dot!

4. Replace *surname* above with **your surname.**

   For example, if I was writing the prac I would use:

   ```
   mkdir PRAC4-colville
   ```

5. Copy the files from `~/../student00/PRAC4` into `PRAC4-surname`

## 1. MPI — Scaling Benchmark

The purpose of this problem is to demonstrate the scaling behaviour of the `stencil_mpi` program's domain decomposition approach to parallelising the solution to Laplace's equation.

In this question you will run the `stencil_mpi` program on the CHPC cluster using one to 48 processors, as given in the table below, and report the time to complete each run. You will use values for *n* and *niters* that are large enough to show the effects of timing; and make sure that *n* is divisible by the various choices of *px* and *py*. Note that *n* and *niters* must be the same for **all** runs in a set.

| ***p*** | ***px* × *py*** |
|---|---|
| 1 | 1 × 1 |
| 4 | 2 × 2 |
| 8 | 2 × 4 |
| 16 | 4 × 4 |
| 24 | 2 × 12 |
| 24 | 3 × 8 |
| 24 | 4 × 6 |
| 36 | 6 × 6 |
| 48 | 2 × 24 |
| 48 | 4 × 12 |
| 48 | 8 × 6 |

You *must* use a *different* PBS job script file for each run. Name them appropriately. (For example, `stencil1x1.pbs`, `stencil2x2.pbs`, `stencil2x3.pbs`, etc.) And name the output log files using the same pattern (eg `stencil2x3.log`, etc.).

Using a text editor create a plain text file called `report.txt` in the `PRAC4-`*`surname`* directory and write all your observations and comments in it.

After listing the parameters you used for *n* and *niters* write down the values of *px* and *py* you used for each run and the run time.

Answer the following about your results:

1. What type of speed-up do you have? It is linear or sub-linear?
2. Why do the run times show different values for different choices of *px* and *py* given the same value of *p*?
3. How does the run time vary with *p* the number of processors? Explain.