High-Performance Computing: Assignment 2

Adrian F. Clark ⟨alien@essex.ac.uk⟩

2016–17

when	before noon on Fri 16 Dec
what	the source code of your program
	the output from your program
	a graph in PDF format
weighting	20% of module mark
return	start of spring term

Introduction

This is the second and final assignment for *CE816: High-Performance Computing*. Having built a compute cluster and installed MPI on it, this assignment asks you to write some MPI-based software that works on a cluster. However, you will not use the cluster you built in the first assignment, you will run your code on our brambles, our dedicated clusters of Raspberry Pis.

You *do not* have to carry out the practical part of this assignment during the scheduled laboratory sessions for this module; however, someone will always be on hand at this time (in the same laboratory as the first assignment was performed) to answer questions or help you if you are having difficulties.

The cluster

As described in lectures, each cluster comprises 47 Raspberry Pi computers, all connected to a single 100 Mb/s Ethernet switch that forms a private network. That network, and hence the Raspberry Pis, are accessed via a single host on the campus network equipped with two network cards — the details are in your lecture notes. Remember that `cseepiman1` and `cseepiman2` are PCs, not Raspberry Pis, so you should not compile or execute MPI jobs on them unless you know how to perform mixed-architecture computation.

Although these machines are a specialist resource available only to students on this module, you should still be able to use your campus login name and password. However, you will not have your normal filesystem mounted: you have to copy files from normal campus machines onto or from the cluster controller using `scp`, a process you may have become familiar with from the first assignment. You will also have to use `scp` to copy files around the cluster.

It is very tedious to have to enter your password on every `pinode` every time you run a program. To avoid this, `ssh` is able to use a public/private key pair to authenticate you; if you do not already have these from the first assignment, you should set one up straight away. Once you have `ssh` set up correctly on `cseepiman1`, you can use the short Python script

```
/home/alien/distribute-key
```

to copy your public key to the relevant place on every `pinode`.

When you have set things up so that you do not have to enter your password on every node, the script

```
/home/alien/distribute
```

should help you copy files from the cluster controller to all nodes. Do take a copy of it and adapt it if it does not do precisely what you require.

The task

1. Your starting point is a conventional serial program for calculating a quantity called *coverage* from data files; it is provided as both Python and C. You are asked to convert one version of it into an MPI program that distributes computation efficiently across the cluster. The program should be able to work with any number of nodes and should yield the same results as the serial code.

   You are welcome to write the program in any language that works with MPI on the cluster. Your program should be well laid out, structured and commented — see

   ```
   http://orb.essex.ac.uk/ce/ce316/assessment-criteria.html
   ```

   for the assessment criteria that will be used for marking your code.

   The program and the data files it requires are provided on the module website for you to download and use as a starting point. However, because the amount of backing store ('disk' space) on each `pinode` is severely restricted, by default the program uses the data files stored in `/home/alien/data/`; you can modify `load_locations` to change this.

2. You should execute your program on an unloaded cluster for several different numbers of nodes and record the time taken in each case, then draw a graph of execution time against the number of nodes. Note that you do not have to run the program for *every* number of nodes, you should select enough values to show the trend. You should save this graph as a PDF file and upload it along with your program when you submit it. You should discuss your graph in the comments of your program — marks for this will be awarded under the *Results* section of the feedback you receive.

3. You should save the output from your program using shell re-direction (`./coverage >results.txt` say) and upload that with your program and graph.

The coverage program and data files are available from the module website in a single zip-file.