Compute Cluster on a Pi

by
Walter Anderson

Beowulf Cluster

- Collection of normally identical, commodity grade computers networked together
- Invented at NASA in 1994
- No requirement for any specific OS or software

Beowulf on the Pi

- Pi released in February 2012
- First Pi clusters started to appear as soon as Pi's could be purchased in quantity, sometime in early 2013

Parallel Computing is DIY

- Very little commercial software available to run on compute clusters
- Using one requires programming

Common languages

- FORTRAN
- C
- C++
- Other languages can be used but examples are harder to find

Message Passing Interface

- Starting in the 80's as supercomputers were evolving into massively parallel machines a number of message passing environments developed
- In the early 90's an effort was started to develop a standard.

MPI-1

- First standard released in 1994
- Most popular, and one of the earliest implementations was MPICH produced by Argonne labs

MPICH

- Adheres to MPI-1,MPI-2, and MPI-3
- Distributed as source
- Tested on Linux (ia32, x86-64), Mac OS/X (Power PC and Intel), Solaris, and Windows

MPICH

- Cluster can be made of any combination of CPU architectures and operating systems that are running the same version of MPICH
- https://www.mpich.org

MPICH

- We will be using the latest stable release MPICH 3.1.4
- Because distributed as source you need to compile, which takes a fair bit of time on the Raspberry Pi
- Providing Raspian Image with all software already installed, just need to modify configuration for each node.

Why you need parallel processing

- If you have a multi-core machine (like the Pi2) and you only program in a traditional manner, you are only utilizing a fraction of the power of the machine.
- You can run MPICH on a single computer if it has multiple cores!

Setting up MPICH on your own

- Download the latest version from: *http://www.mpich.org/downloads/*
- *Create a ~/mpich directory*
- *Create a ~/mpich/build directory*
- *Create a ~/mpich/install directory*

Setting up MPICH on your own

- Unarchive the downloaded mpich source to ~/mpich/mpich-3.1.4 (or whatever version your using)

```
pi@PiClstr01 ~/mpich $ ls -l
total 12
drwxr-xr-x   7 pi pi 4096 Aug  5 19:16 build
drwxr-xr-x   6 pi pi 4096 Aug  5 21:10 install
drwxr-xr-x  11 pi pi 4096 Feb 20 15:06 mpich-3.1.4
pi@PiClstr01 ~/mpich $
```

Setting up MPICH on your own

- Run configure from your build directory (must have gFortran installed first).

```
pi@PiClstr01 ~/mpich $ ../mpich-3.1.4/configure \
–prefix=/home/pi/mpich/install
```

Setting up MPICH on your own

- Build the application

```
pi@PiClstr01 ~/mpich $ make
```

```
pi@PiClstr01 ~/mpich $ make install
```

Setting up MPICH on your own

- Add the ~/mpich/install/bin to your path

Setting up MPICH on your own

- Set up SSH on your primary node

```
pi@PiClstr01 ~/ $ ssh-keygen -t rsa -b 4096 -C "pi@PiClstr01"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pi/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pi/.ssh/id_rsa.
Your public key has been save in /home/pi/.ssh/id_rsa.pub.
The key fingerprint is:
25:ad:d0:95:42:25:b3:cc:ca:a0:4c:a8:c2:7b:f0:ca pi@PiClstr01
The key's randomart image is:
+--[ RSA 4096]----+
|        .+.o.    |
|        ..*.     |
|       . *.o     |
|        * *      |
|         S       |
| o      .        |
|. * . .          |
|=E*    .         |
+-----------------+
```

Setting up MPICH on your own

- Create duplicate set-up, including user and directory structure for all MPICH files
- Copy primary node SSH credentials to each of the secondary nodes

```
pi@PiClstr01 ~/ $ ssh-copy-id 192.168.0.#
The authenticity of host '192.168.0.# (192.168.0.#)' can't be established.
ECDSA key fingerprint is 25:ad:', and checkd0:95:42:25:b3:cc:ca:a0:4c:a8:c2:7b:f0:ca.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.#' (ECDSA) to the list of known hosts.
Now try logging into the machine, with 'ssh 192.168.0.#', and check in:

  ~/.ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

Setting up MPICH on your own

- All nodes must have either a static IP or always receive the same IP from your DHCP server.
- List all nodes:cores in a text file that you provide to MPIEXEC to tell it what machines to run your application on.

Setting up MPICH on your own

- Now you just need to write some software
- Provided image includes what is needed for Fortran 95, C, C++
- Other languages that have bindings for MPICH include Python and R

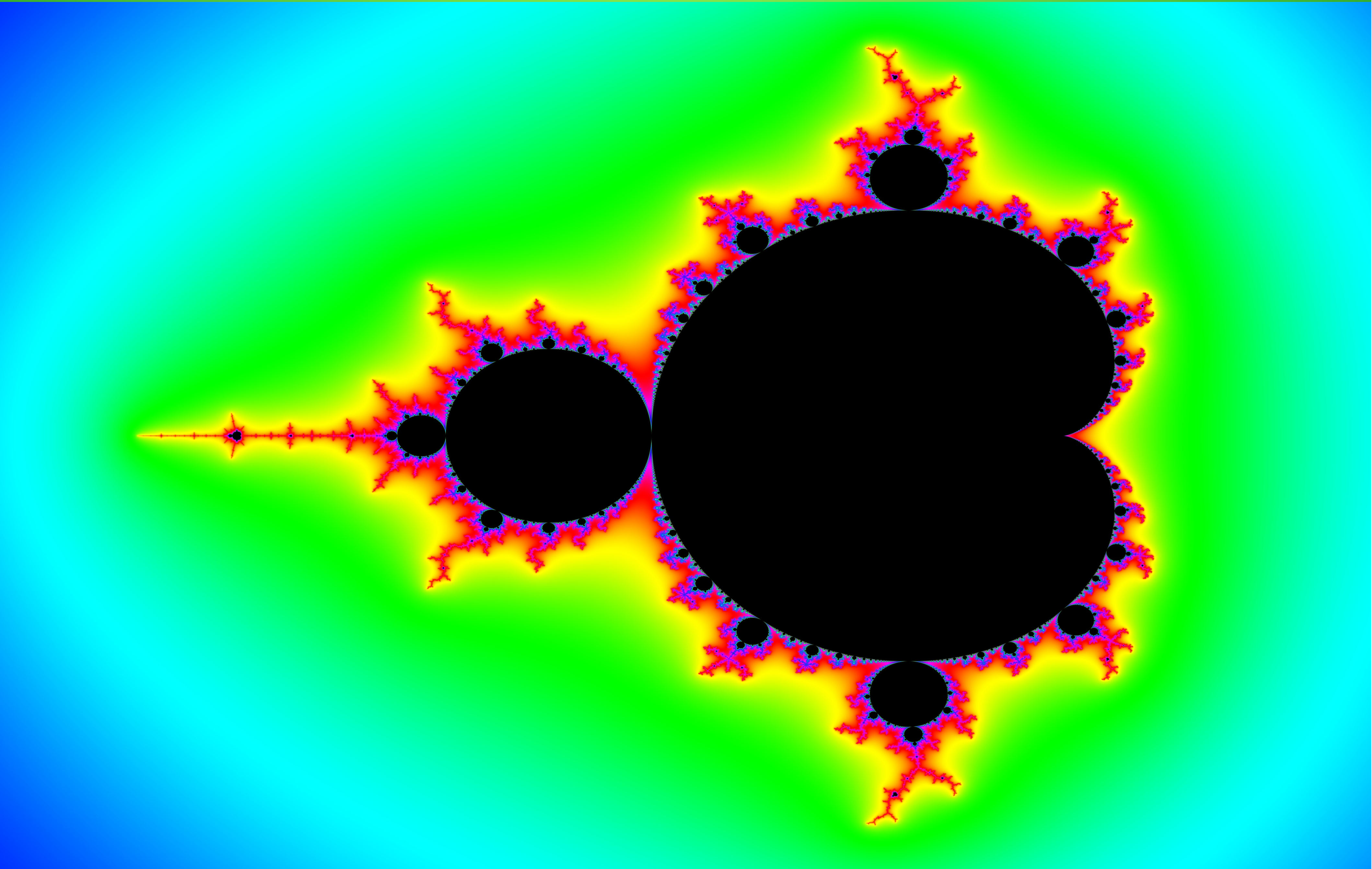
Modify your copy for your node

- Change name in the /etc/hosts
- Change ip address in the /etc/network/interfaces
- Use the information on the card I handed out

Our Example

- Needs a low network bandwidth for communication between processes
- Needs to have ability for highly parallel, independent computations
- We will use classic Mandelbrot set calculation as our example

Mandelbrot Set



Mandelbrot Set

- Equation $z_{n+1} = z_n^2 + c$
- The formal definition can remind you of a university math class you hated; however, the idea is pretty simple.
- If you iterate the equation and after some number of iterations the value is still less than 2, then you can assume point is within set, otherwise assign the point a color to indicate its proximity to the set.
- The colors are assigned to the points outside of the set based upon the number of iterations it takes to determine the point is not within the set

Running pmandel

- Mpiexec -n 64 bin/pmandel -xscale 2000 -yscale 2000 -i

```
Welcome to the Mandelbrot/Julia set explorer.
input xmin ymin xmax ymax max_iter, (0 0 0 0 0 to quit):
-2.0 -1.0 1.0 1.0 1000
read <-2.0 01.0 1.0 1.0 1000
>from stdin
x0,y0 = (-2.0000000, -1.000000) x1,y1 = (1.000000,1.000000) max_iter = 1000
input xmin ymin xmax ymax max_iter, (0 0 0 0 0 to quit):
0 0 0 0 0
read <0 0 0 0 0
> from stdin
x0,y0 = (0.000000, 0.000000) x1,y1 = (0.000000,0.000000) max_iter = 0
Done calculating mandelbrot, now creating file
pmandel.ppm
width: 2000
height: 2000
colors: 100
str: Mandelbrot over (0.000000-0.000000,0.000000-0.000000), size 2000 x 2000
```