Working doc. Just getting started.

**Why Orange Pi? Why not? Reasons Pro and Con**

What is an Orange Pi? If you are interested in this tutorial, you should already know that the Raspberry Pi Foundation popularized the full featured SOC (System on a Chip) based “credit card” sized single boards that only require an SD card, mouse, keyboard, ethernet connection, power supply (usually a cell phone charger) and monitor to serve as a decent linux desktop. Sometimes, just the SD card, power and ethernet cable are all you need if you are going to use ssh in a server configuration. In a Beowulf cluster, each node or computer on a local area network is just a “commodity” computer with software that allows the nodes to cooperatively process data. So why use Orange Pi? For one, the Orange Pi (which includes a number of different models) is about as “commodity” as it gets with prices as low as $9.99. (The “Orange Pi One” with four ARM cores and 500mg RAM). While they don’t have the computer power that an Raspberry Pi 3 has, they are close enough and ⅓ to ½ the cost. But more importantly, what we want to convey with this tutorial is not so much an endorsement of the Orange Pi line of Single Board Computers, so much as making a more neutral endorsement on the Raspberry Pi line. Don’t get it wrong, we love the Raspberry Pi! It is proving to be world changing; a truly historic computer.

But it is not the only single board computer out there, and we should anticipate the obsolescence of all of today’s hardware and even brands in fairly rapid succession. (This mid-career author’s first computer was a screaming 16K memory Radio Shack Color Computer.) What seems likely to remain much more stable for at least a long time is \*NIX variant operating systems and especially Linux - which is for now the global OS that runs on by far the most computers. (Android, for example is based on linux.) Further, open source Linux does now and should continue to dominate in high performance computing, and probably the Internet and Internet of Things will be a dominantly Linux world as well. Boards and hardware will change rapidly, but Linux is here to stay. So rather than endorse the Raspberry clones that come in flavors of Orange, Banana, or other single board computers of the moment such as C.H.I.P, Udoo, Odriod, BeagleBone, Pine A64 or whathaveyou, we wanted to give the sense of the alternatives and show that a cluster can be made from any of them.

Single board systems have actually been around for some time (we used to call them embedded systems boards or “matchbox servers”), and even before that, the Apple 1 computer was an ingenious single board system engineered by Steve Wozniak in 1976. The options available today are numerous, ranging from simple boards for electronics projects, such as the indispensable Arduino family, to full Windows capable PC platforms sometimes running near the $100 price as “bare bones” or “hdmi dongle” systems. The System on a Chip configuration of adding memory directly to or very near the processor (and integrating a video processor) to make a “single chip” computer is an evolution in SBCs of course, and this has led to the large number of contemporary single board computers, many of which do the same kinds things, or a few more more things (say, sensors), or less things (totally “headless” or without video output), with many different combinations of features. Avoiding the details of this great diversity, we recommend you have a look at the Wikipedia page for single board computers to mavel at the current diversity, <https://en.wikipedia.org/wiki/Comparison_of_single-board_computers>.

There are SBCs out there that might be better fits for particular applications. There are very few reasons that would prevent most of these boards from being used to create an educational cluster, although certainly some might be more convenient or appropriate than others. In particular, you are likely to want ethernet running on a proper RJ-45(8P8C) port for networking. You might choose a multi-core processor as these are as common in production systems as they are everywhere, same for the choice of 64 bit over 32 bit. Features such as HD output or even 4K output might be nice if you want to try your hand at data visualization. Gigabyte ethernet (10/100/1000) is really nice if available, as opposed to “fast” or 10/100 mbps ethernet. In general, faster is better, but for the purposes of learning cluster compute techniques, performance is an afterthought. Outside of the hardcore hobbyist, most clusters created from inexpensive SBCs will not need to worry about actual high performance computing. The important matter is that the concepts, setup, softwares, and programming techniques are very much the same as would be found on most very high performance, professional, production “supercomputers”. SBCs are an ideal platform for learning, if not for actual supercomputing in which nodes are normally counted in the thousands or tens of thousands.

On the wiki list you will find Raspberry Pi and Orange Pi, the OPi being among the fruit of the moment clones of the RPi coming from Shenzhen, China, arguably the greatest of the current capitol’s of electronics manufacturing in the world. These instructions for using the Orange Pi PC in place of the Raspberry Pi is just one example of many different options you have. But there are a few things that - at least at this current moment - called our attention to the Orange Pi. The Orange Pi PC is one of many Orange Pi models, others being the original Orange Pi, Orange Pi mini, Orange Pi Plus, Orange Pi2, Orange Pi2 Mini, and Orange Pi Plus2! That is a lot of juice. The Orange Pi PC is the model of Orange Pi that is most similar in features to the Raspberry Pi 2 in terms of hardware features, so it makes for a comparable alternative. But instead of $35, it is currently only $15 dollars for the Orange Pi “PC” and $9.99 for the “One”, that is a lot of value when you consider how generally comparable the boards actually. And for 5 shekels more than the Raspberry Pi, the Orange Pi Plus adds 8GB of emmc flash memory (so you don’t necessarily need a dedicated SD card), onboard wireless (no need to add a WiFi dongle if that is your desire), gigabyte ethernet and even SATA connector that would let you hook your cluster up to an SSD drive or even a SAN to create a rig a bit more like the high performance storage systems that “real” or production clusters use.  
  
There are also issues that make the Orange Pi line less advantageous. While the power requirements are essentially the same, the Orange Pi line of single board computers uses a 40mm/17mm barrel connector for power (don’t put a mobile charger into the OPi’s micro USB On-The-Go connector and expect anything good to happen), and of course the Raspberry Pi has a tremendous support community that makes it easier to solve technical problems and answer your questions, in particular for Raspbian. The Orange Pi line in particular tends to ship much more slowly, at least this is our experience through the primary manufacturer and its prefered sales channel which seems to be the only way to enjoy that bottom shelf $15 price. (<http://www.aliexpress.com/item/Orange-Pi-PC-linux-and-android-mini-PC-Beyond-Raspberry-Pi-2/32448079125.html>) And we have found the components such as USB ports to be a little lower quality (sometimes requiring a little wiggle to connect USB, for example.) But the bottom line is, with the Orange Pi PC or other alternative boards, the process and configuration necessary to build your own cluster may fit different needs, such as lower cost or higher performance or both, even given that there are likely other kinds of “cost” and “performance” scenarios where the Raspberry Pi and its ready availability and strong support community makes it more advantageous. (Again, in education the Raspberry Pi ecosystem obviously has so many great advantages in its ecosytem that it is a natural choice for schools even at more than twice the price of an Orange Pi PC.)

In the long run, all of these boards will continue to evolve, so the choice of the Orange Pi as an alternative is really more to demonstrate that there are and will be alternatives with very similar (occasionally identical) set up. The installation process for a variant linux operating system is at least very similar to Raspbian OS as we will see with the Debian variant Lubuntu. We think this is important too. After all, the purpose of building a Raspberry Pi cluster is to learn about High Performance Computing without actually doing to much in the way of high performance computing. As a second project, it is a good idea to build a cluster from different hardware, perhaps with greater performance ambitions. Why? We want learners to be able to transfer and scale the concepts they are learning on small clusters to larger ones. The learning goal is obviously not to teach the making of RPi clusters; educators should avoid brand lock at all costs. (Looking at you, Apple and Adobe fanpeople in education.) Rather, the board(s) you choose will be a means to an end in understanding clusters and becoming lithe in working with rapidly changing hardware (and software) over the time span of careers. Certainly, the goal is not to serve students only over the time span of teaching projects. Nor should teachers be part of the marketing of the hardware or software upgrade cycles. The bread and butter of computing behemoths is to profit from “their” consumers, those who have become mired within narrow product ecosystems. We want to teach students technological freedom and personal agency.

**Let’s go!**

First, we will be using 5 Orange Pis. One is an Orange Pi Plus 2 (which has wireless and some other goodies like a SATA connector) and the rest are Orange Pi PCs (the $15 type). Any number is fine of course, and really this tutorial could be done with 5 Orange Pi PCs (all the same) or 5 or any number of most any other board. But we are going to test a few options that the higher performance hardware of the Orange Pi Plus 2 makes available. (Particularly its gigabit ethernet, as it has exactly the same CPU and clock speed as the others. It does also have twice the RAM.)

OrangePi images can be found at <http://www.orangepi.org/downloadresources/>, and are produced for the various models of Orange Pi. There are many image options, including Android and Raspbian! But we are working with the Lubuntu (LXDE/Ubuntu), as it is the more common distribution for the Orange Pi. Meaning, the Orange Pi Foundation seems to publish images for Lubuntu and and Android before any others, although a wide range of options tend to appear eventually including Raspbian, Kali Linux, ARCH Linux, OpenSuse, etc. Lubuntu is also a good choice because it shares Raspbian’s Debian roots in addition to the LXDE desktop, so we should find only slight variations in the Operating System software. Also, Ubuntu is one of the most popular linux distributions and there is a lot of support for it. It may be fair to say that Ubuntu actually has a much larger support community than Raspbian, thus even if it is not so particularly tuned to educators, new users or hobbyists, the answer to most questions are out there in the Ubuntu forums, and similarly searchable with your favorite search engine. You can treat most Lubuntu questions as Ubuntu questions, because the only really difference between the two is the choice of the LXDE desktop on Lubuntu over the Ubuntu Unity Desktop.

Lubuntu itself is indeed mostly the same as Raspbian, as it turns out. There are a few different choices of packages here and there, including Lubuntu’s use of NetworkManager as noted below. (But we can fix that;-) One more potential advantage of Lubuntu is that it can be compiled for or quite easily directly installed from a DVD image onto standard PCs! So if you want to move from processing power that is counted in watts to processing power counted in kilowatts, these Lubuntu based instructions should be exactly the same if you are building a cluster out of old PCs, even if your electric bill will not be! Further, you can easily add a single PC with higher speed storage, perhaps a RAID, to your Orange Pi cluster as a master node, which is fun and can boost the performance of your cluster. We will be trying this at the end of the tutorial.

The best news here is that you install image files for Orange Pi (and others) onto a microSD card using pretty much exactly the same techniques as you would use for Raspbian on the RPi. (There is no NOOBs for OPi, but the image writing foo is the same.) <https://www.raspberrypi.org/documentation/installation/installing-images/>

Here is what we did from an Ubuntu Laptop:  
$ sudo dd bs=4M if=~/Documents/qp/Lubuntu\_1404\_For\_OrangePiPC\_v0\_8\_0\_.img of=/dev/mmcblk0

One thing that Lubuntu does not come with is the easy to use raspi-config program, which is so useful with Raspbian. (Save, of course, the existence of the Orange Pi Raspbian image, one that we don’t necessarily recommend.) To use Lubuntu, we are going to have to do more command line linux fu instead of terminal menu app fu. Here are the elementary steps we need to attend to according to the SDSC RPi instructions:

* Change the default user (orangepi) password (which is “orangepi” by default)
* Change the hostname
* Expand the file system to use your full storage capacity.
* Set up ssh with keys

password:

Note that the default user account is orangepi, and that it is a different account from the root user. You should probably change both as both are set to “orangepi”. However:

$ sudo passwd  
…

THIS DOES NOT WORK with the stock Lubuntu Orange Pi PC image. Even worse, it will accept your new password and claim to successfully change it. But, on reboot the password will change back to the default value. Odd, yes? The trick that works is to actually switch to the root user:

$ su root # and enter the root password which is also “orangepi” by default.

# passwd

… (change roots’s password to something safer than the default)

# passwd orangepi

… (do the same for the orangepi account

# exit

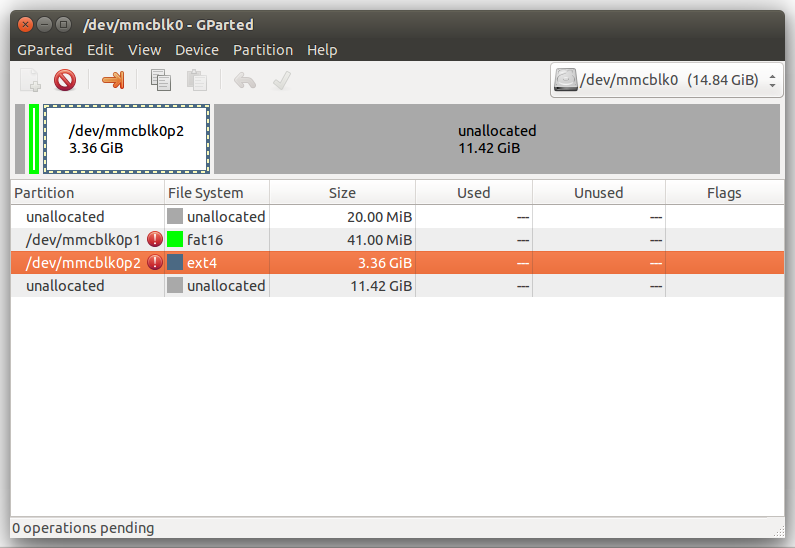
$

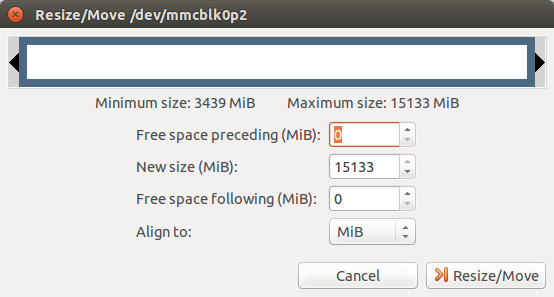
Of course, write these down!

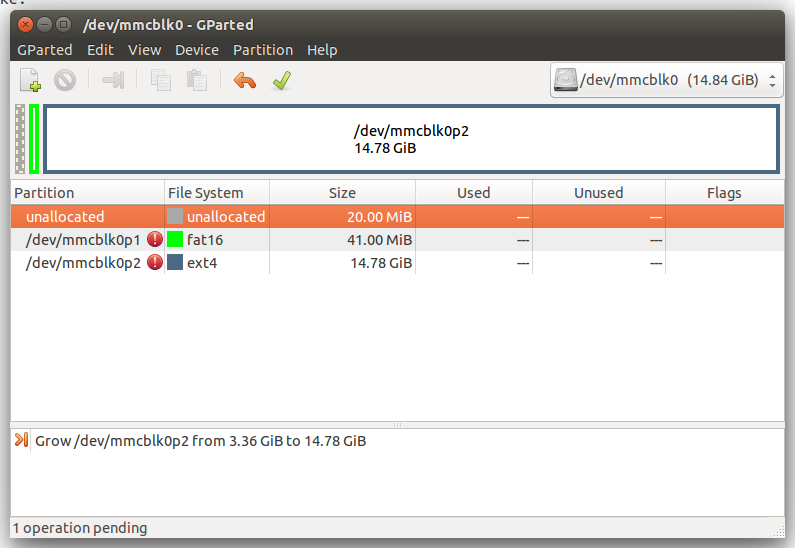
Resize the root partition:

Resizing the system partition is possible with the fs\_resize bash script that you may find in /usr/local/bin. (This is true if you are using the images from orangepi.org.) This script seems to have been borrowed from the Odroid line of single board computers, and works great. If you don’t have this, try doing a search for odroid and fs\_resize. You will probably find what you need.

The easiest approach may apply if you are running another Ubuntu (or some other linux) box with a card reader. In truth, that old laptop with a card reader that you no longer use might make for a very useful computer for doing raspberry pi kinds of things, and the standard Ubuntu distribution is very easy to install, and is efficient enough to give an old notebook new life after Windows. Simply poweroff your OrangePi, remove the SD card and insert it into that other linux box. (Or if you used Ubuntu and dd to write the image to the SD card, just continue from there. Use (you may need to install) the graphical GParted utility (a gui for parted) to resize the partition. This little fall back into gui worlds is useful/easier if you would like to create other partitions. Here is what gparted looks like:



Right click on /dev/mmcblkp2 and select resize. You can simply drag the graphical representation of the available space to the size you wish, or enter the values you want by hand. Then click resize/move:  
  


After this, gparted will show that is has one operation to complete. Click the check symbol and the partition will be resized:  
  


**SSH**

One thing we \*do not\* need to do is install or configure the openssh-server. The raspi-config utility certainly makes this very easy to enable in Raspbian, but it is even easier on Lubuntu! Because it is already installed and enabled as a standard package in Lubuntu:-) (See optional section for some good ideas that should improve login security.)

**Networking**

Another difference is that Lubuntu (as well as desktop Ubuntu) uses NetworkManager out of the box (<https://en.wikipedia.org/wiki/NetworkManager>). NetworkManager is designed to more seamlessly manage network connections for notebooks and desktops, also providing an easy to use GUI. (Even more information on NetworkManager can be found at <https://help.ubuntu.com/community/NetworkManager>.) It works great, and is what you should use if you want to use an SBC with Lubuntu as your daily OS, much as you would with a desktop or notebook. In fact, you could simply use it to change all kinds of things, including the hostname, dns server and hosts as called for in this tutorial. See LXDE Menu -> System Tools -> Network. Similarly, you can look in the system tray area for the Network Connections tool and add a static network connection and ip address, etc, right there. And, you can use the settings below, simply adding them to the GUI! That is easy and may be nice for many. (Not to mention, some of the higher end Orange Pis come with wireless, making NetworkManager even more of important piece of software on your computer.)  
  
NetworkManager obviates many of the normal linux configuration files, as in, takes them out of service. (Boo, nm-tool is a gui utility that causes the interfaces file to be ignored.) If you are working with Orange Pi PCs without wireless for building a cluster, you may want to remove NetworkManager from the system and set up your network using the following files: /etc/hostname, /etc/network/interfaces, and /etc/hosts

1) (Optional) Remove NetworkManager

~~$ sudo apt-get remove network-manager~~

~~…~~

~~And don’t worry, you can always~~

~~$ sudo apt-get install network-manager # if you change your mind~~

~~Note: Of course, you can reinstall network-manager only if you have network access. If the manual network configuration steps, go awry you could be in trouble.~~

1a) (Still Optional) Disable network manager (Note: this is better than removing it, as it is easy to re-enable.)

$ sudo mv /etc/init/network-manager.conf \

/etc/init/network-manager.conf-disabled  
$ sudo mv /etc/xdg/autostart/nm-applet.desktop \

/etc/xdg/autostart/nm-applet.desktop.disabled

By changing the extensions of the /etc/init configuration files for networkmanager and the nm-applet (the gui for network manager) from .conf to .conf-disabled, Ubuntu will not start these processes on boot. If you do this but don’t reboot immediately, you will need to issue the following to stop the services already started at the last reboot.

$ sudo stop network-manager  
$ sudo stop nm-applet

2) Give your computer a name better than “orangepi”. Note that this does not refer to the account name (user “orangepi”) but the name of your machine as it presents itself to whatever network it is connected to.

$ cat hostname # just displays your computer’s name

(probably “orangepi”, it would be “raspberrypi” given the normal Raspbian image…)  
$ cat /etc/hostname  
(This should be the same as /etc/hostname, because /etc/hostname is the file read on boot that sets the hostname. In other words, /etc/hostname is persistent; your computer will get that hostname on reboot.)

$ sudo hostname *somethingelse* # temporary change

$ hostname # should now show your hostname is *somethingelse*

$ sudo (your favorite editor) /etc/hostname # and change from orangepi to opi0 (or whatever you are naming it…)

$ sudo reboot # need to reboot

…

3) Set up your hostname in your hosts table:

$ sudo (your favorite editor) /ect/hosts # and change the localhost entry from orangepi to opi0 (or whatever you are naming it…)

Note that each of your nodes - each board - should have a different hostname. We have reserved the following as our hostnames: opi0, opi1, opi2, opi3, opi4... Note that, after changing the hostnames in /etc/hostname, a restart (reboot) is needed.

4) Here is an example of something like what you want, with our additions in green and one line in red that you should comment out by placing a # in front of it. Depending on your distribution, there may be other things in there too. (See also <http://manpages.ubuntu.com/manpages/utopic/man5/interfaces.5.html>)

First, you might want to make a backup copy of the original interfaces file:

$ sudo cp /etc/network/interfaces /etc/network/interfaces.backup

Then proceed with your favorite editor to edit /etc/network/interfaces

# interfaces(5) file used by ifup(8) and ifdown(8)

# Include files from /etc/network/interfaces.d:

# source-directory /etc/network/interfaces.d

# Other stuff may be in here. For example, wlan0 or other wireless

# lan designation is probably best left alone!

# You can comment out any references to just eth0, as loopback will

# often be configured as we see below. Basically you want something

# like what follows as the setup eth0, the set up for a static ip.

**auto lo**

**iface lo inet loopback**

**auto ethO**

**iface eth0 inet static**

**address 192.168.0.100**

**netmask 255.255.255.0**

**network 192.168.0.1**

**broadcast 192.168.0.255**

**gateway 192.168.0.1**

**dns-nameservers 8.8.8.8 8.8.4.4**

Notes: you might use other dns servers, see the configuration of your router or other institutional info in order to utilize the dns server provided by your isp or other provider. 8.8.8.8 and 8.8.8.4 are Google’s open dns servers, and they work fine for free, thanks google.

Explained, the first block after the line you should comment away if it is there configures your loopback address. The second block achieves our end goal, which is to give each node a static ip address on on your local area network (a private network). In this case our local area network is provided behind a NAT (network address translation) firewall provided by a router (next section) whose LAN is configured as the 192.168.0.1 network, and the above configuration is boiler plate. The only value we change is to give OPi1 an ip address of 192.168.0.100 such that it matches what if found in /etc/hostname. (see the address line.)

Note here that the setup is a little different from the “Raspberry Pi” instructions we offer, but it is important to point out that the differences have almost nothing to do with “Raspberry Pi vs Orange Pi”, or better said, “Raspbian vs Lubuntu.” For the ip addresses we are using different reserved ip address space for private networks, because in this example we are running the cluster on something more similar to the private network address you are more likely to find behind the kind of router that you would normally find connected to a home network. (As opposed to a simple switch sans routing in the Raspberry Pi example.) In this example, we are using an old Netgear FVS318 router, and it’s default ip address for the LAN is 192.168.0.1. So, that is the appropriate value for both the gateway and network setting in the /etc/network/interfaces file. As a big bonus, the router itself provides us with Network Address Translation (it is built in and managed through the router’s web interface!), which saves us from needing to configure NAT or some other firewall. A major limitation exists only if you plan to connect some hundreds of nodes, but the space of 253 addresss between 192.168.0.1 and 192.168.0.255 (reserved for broadcast) is more than enough for our five planned nodes.

In summary the above interfaces file example hardcodes this OPi to use the ip address 192.168.0.100, and on such a LAN the netmask of 255.255.255.0 and broadcast of 192.168.0.255 are also common values. Most of the time, your values will be similar given home gear. For example, if your router’s LAN address is 192.168.11.11 (default values are set by the manufacturer and are often printed on the outside of your router) then you would use that value for network and gateway, and the same broadcast and netmasks as above, and your choice of ip address on the LAN. Each of your Orange Pi nodes will need a different value for only ipaddress: after 192.168.0.100, we use 192.168.0.101, 192.168.0.102, 192.168.0.103, 192.168.0.104 for the Orange Pi PCs, to correspond to our chosen hostnames for each of the devices opi0, opi1, opi2, opi3 and opi4. The numbering and naming correspondence is for no reason other than mental convenience.

4) If you want your nodes to be able to find each other, you can also add something like the following to /etc/hosts (again, using your favorite editor!)

127.0.0.1 localhost OPi0 # local and hostname of this OPi

198.162.0.100 OPi0

198.162.0.101 OPi1

198.162.0.102 OPi2

198.162.0.103 OPi3

198.162.0.104 OPi4

**Setting up your router**

Just because your OPis have a hostname and IP address assigned to them does not mean that your network (LAN) recognizes them. Since in this example, the LAN is managed by a dedicated hardware device, the device also has to know what addresses you want reserved for which devices, and the devices should agree with the router for everyone to be happy. Most every router has these features built in. The old FSV318 uses a still typical web interface, and you will have to refer to your own router what it’s default ip address is and how to fully configure it. If it is a wireless LAN you installed yourself, you probably remember how to do this. Most routers reserve some IP addresses for DHCP. Which assigns IP addresses to devices configured for DHCP from a pool of addresses. So you don’t want to choose IP address from that pool. On our router the range reserved for DHCP was preset ft 192.168.0.2 to 192.168.0.100. Because I wanted to use 192.168.0.100, I had to adjust the router settings for the DHCP range, which was possible on the FVS318. (Not shown here, devices may vary.) The router also has a way to reserve the IP addresses you want for your various hosts, using a simple interface pictured below. You will note that there is an additional bit of data you need to reserve this. Your router knows the ethernet devices attached to it through each device’s MAC address, a unique id burned into all Ethernet hardware such as on the Orange Pis. To find that unique address for each of your OPis, you will need to execute the following command:

$ ifconfig

eth0 Link encap:Ethernet HWaddr 21:1b:07:61:19:d5

inet addr:192.168.0.100 Bcast:192.168.0.255 Mask:255.255.255.0

inet6 addr: fe80::221a:6ff:fe60:18d4/64 Scope:Link

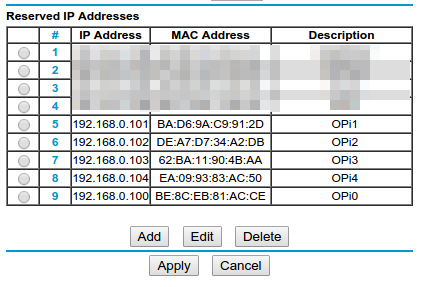
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:274034 errors:0 dropped:0 overruns:0 frame:0

TX packets:223684 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:217297880 (217.2 MB) TX bytes:48166619 (48.1 MB)

You should notice a few now familiar things, such as the OPis ipaddress on the LAN, Bcast or broadcast address, and Mask or netmask address, and some stats and other stuff. (The above is from my notebook computer. The important part for us is listed in red above, the HWaddr or hardware address. That is the value that you will copy into the router’s interface for reserving addresses, as shown below. (I pixelated unrelated machines on my LAN just so it is not confusing.)  
  


Review the values you have entered just to make sure they agree with your nodes. When you are done with all of this, and all of the devices have been restarted, each of your OPis should be visible to each other on the network, even by their host name:  
  
(OPi1)$ sudo ping OPi4

… no lost packets

**Setting up ssh**

You don’t have to, as client and server are installed and working by default in the Orange Pi Lubuntu image.

But we do need to set up a private/public key pair that your controller node (OPi0 in our case) can use to log into the remaining nodes to send them parallel instructions and data, and it is the same as the Raspberry Pi instructions:

$ ssh-keygen -t rsa # on your master node

… do not supply a passphrase here, just leave it blank

$ ssh-copy-id orangepi@OPi1 # to your slave nodes, OPi1, OPi2 (192.168.0.101, 192.168.0.102…)

**Setting up the Network File System (NFS)**  
(todo)

**Some optimizations:**

Security

There are also some configurations you might want to make to help secure your cluster. This is especially true of course if your network is setup, as is common, to allow ssh connections to your OPi from various WANs (wide area network) or the outside internet, the biggest WAN of all.

It can obviously be unsafe to run any device with the default user and password, which is why we changed passwords above. But it can also give an intruder a head start if they know the name of a user on a device that they can try to hack or guess the password for. So you may want to add a new user (such as “picluster”, but choose whatever name suits you) and add them to the sudoers list so picluster can sudo:

$ sudo adduser --system --gid 1001 --home /home/picluster --shell /bin/bash picluster

$ sudo passwd picluster #give the new user a password

$ sudo visudo

… add the line “picluster ALL=(ALL) ALL” under “orangepi ALL=(ALL) ALL” in the “User Privilege Specification” section

$

At this point, you might use deluser to delete the orangepi user altogether (and remove the related line using visudo too), making it much harder to guess passwords because the user name is not generally known, as orangepi is.

**Create a swap partition**

This really has nothing to do with anything. Swap is virtual memory for Linux, meaning that if the computer runs out of RAM, it will start using the swap area of the disk (or a special file in the case of Raspberry Pi) like it was memory. On SBCs that boot from a SD card, this is not actually a useful idea because flash memory is very slow. Once upon a time, it would be valid to complain that flash memory also wears out (has a limited number of read/write cycles), but as flash memory improves this is less and less of an issue. Nevertheless, it is a good system admin exercice. Swap space is great to have on spinning disk media or SSD (the latter also being flash memory devices, but which uses flash memory and caching in very smart ways to make SSDs very fast in comparison to the underlying flash memory, which is all and SD card is. In other words, SD cards are slow and SSD drives are a whole other, very fast, creature.) So here is how to do this if you want to:  
  
First, you need to create an actual swap partition. When expanding the OPi file system to use my whole 16GB SD card(s), I left space for and formatted a 1025GB swap partition at the end, which turns out to be device /dev/mmcblk0p3. This is not possible with the fs\_resize script in /usr/local/bin, as it always extends the /root partition to the end of the device, leaving no swap partition even if you made one. I used a separate linux notebook that is my day to day computer, which happens to have an SD card reader (using an adaptor for the microSD card.). Using that and the good graphical tools “disk” and “gparted”, I expanded the /root partition to 1026MB short of the end of the unallocated space, then created a new partition in that remaining space and formatted it as linux-swap. And for now, we will leave the reader to look for tutorials on how to use these powerful and dangerous tools. Then back into the Orange Pi my (micro)SD card went. This just prepares the space for the swap partition, and the simple procedure below completes the deal.

You will need to find out the UUID of your swap partition, like this:

$ sudo blkid /dev/mmcblk0p3

/dev/mmcblk0p3: UUID="82923b36-6a37-4df5-82b8-19e0ba88fbf3" TYPE="swap"

With this info, add the following line to /etc/fstab:  
UUID=82923b36-6a37-4df5-82b8-19e0ba88fbf3 none swap sw 0 0

On reboot, run the top command and you should see slightly more swap than memory. For most normal purposes, you want close to equal swap space to RAM. Note that the Orange Pi Plus2 has 2GB of RAM so you may want to use 2GB (plus a little) swap. Ram on this generation of single board computer varies widely.

**What kind of parallelism? Starting with Message Passing Interface**

Parallel programming is of course opposed to serial programming. Consider a bowl of breakfast cereal (say Cheerios), and say your problem is to eat the bowl of cereal. If we were approaching this cereal problem as a serial programming problem (get it?) we would simply put one Cheerio at a time into our spoon and follow the rest of the breakfast algorithm. A more parallel approach would look be to put more than one Cheerio in our spoon. A massively parallel approach would be to divide the entire box up into different bowls feeding different mouths - tp make the Cheerios go away very quickly. Essentially the study of parallelism looks at the different ways we can divide a computational task up, send it to different processing units, and get some kind of result from that, in the metaphor at hand, a healthy group of humans. For computers, the result may be as simple as a sum, or as complex as super resolution animations of the behavior of models (think, weather and climate simulations), or the things that can be discovered from data streaming into sensor networks (such as a number of radio telescopes) or sales data.

There are many kinds of parallelism. The type we experience the most comes to us through our multi-core desktop, notebook and mobile computers (some of which are still quaintly called “telephones”.) Most CPUs today process 64 bits of data at a time (64 Cheerios!), but very recently the more common number was 32 bits. Expanding the number of bits processed at any one time is a kind of parallelism. Pipelining is an optimization feature of CPUs allowing their execution units to perform the same kind of operation on multiple streams of data in the CPU’s cycle, sort of like adding the next tree branch to a wood chipper before the last tree branch has gone through. This makes CPU’s capable of performing some multiples of the work it would do if operating one instruction stream at a time. With CPU design now hitting many of the physical limits of what can be done with silicon, the approach has become to include multiple cores, as in, more than one execution unit on the same computer chip. These cores share the same memory (some combination of high speed memory caches and RAM) and require special executive coordination to prevent them from using each other's workspace (memory). In a sense, these CPUs share the same workshop space, and have to be considerate of each other so as to not interfere with the work of others. This is called shared memory parallelism, but in most ways it presents itself as a single computer like a desktop or a notebook.

The kind of parallelism we are interested in is really the present and future of computing.

**Installing MPI on Lubuntu**

To actually write parallel programs

$ sudo apt-get install libcr-dev mpich2 mpich2-doc

…

$ sudo pip install mpi4py

This installs MPI and a number of related libraries, and the pip adds mpi4py to your python interpreter.

**Some teaching notes:**  
The importance of the visual components (having monitors) for learning

Good sorting card trick (52 pickup?) Well with four people, but what about other ways to divide the task? What about with 52 people? (Bubble sort or quick sort yes, but it would not be faster!)

C and Python are Rick’s preferred languages, because C forces you to deal with the machine at a lower level, and Python makes higher level bells and whistles like OOP and garbage collection…

We could both use matrix clips! (I need the rabbit, Baudrillard scene, Rick the Cypher/Neo presentation layer scene.

SDSC: modeling and simulation vs big data problems.

Note on the audience:

So far I am getting the feeling that a basic Linux/Raspian orientation class is necessary. Many today are linux users with CS backgrounds. But some of the members of today’s class lack the necessary scaffolding. One did not know there were operating systems beyond OSX and Windows! For others, using a cli, or editing a text file in a cli was foreign (to a few of our guests), and the idea that a .py file is a text file with python stuff in it was unparsable. (So need to explain cat, sudo, ls, arp, etc is leaving a few of the students feeling lost.) Networking aspects are also mysterious to many in the group.

Workshop might be better behind hardware routers and their networks than with a bare switch?