

Yellow Dog 6.1 PS3 Cluster guide

Hello everyone, my name is Samir Gartner and this guide is part of my university graduation project. I decided to share it because of the lack of clear documentation about the process of deploying a PS3/YDL cluster. Consider this almost a “PS3 Cluster Deployment for Dummies” guide since I putted some very clear and for some people unnecessary and obvious instructions. I apologize in advance for my grammar but English is not my main language. If you find any technical errors please let me know.

The instructions to install GNU/Linux or other opensource OS in the PS3 are widely available (you can use the first part of the PS3/FEDORA instructions) so refer to those instructions for installing Yellow Dog Linux first. I recommend installing all the development tools as well as KDE and Enlightenment GUI even if you don't like it. In my case I didn't installed enlightenment and experienced some weird behaviors that lead me to a reinstall.

Despite yellow dog has its own boot loader it is very raw and not pretty so I recommend to use OpenSUSE boot loader because it has a beautiful GUI with functional mouse and it allows you to boot into game mode very easily.

I also recommend to use identical users and passwords across every node in the cluster and if you pretend to use different operating systems I also recommend to assign identical permissions users and group IDs in every machine, this will facilitate some processes and services like NFS and file sharing,

Shell environment variables

You must add some paths to the shell environment variables to simplify some commands and access. In order to do that you must modify the `~/.bash_profile` y `~/.bashrc` files in each system user that you are pretending to use. The first file is used to authentication terminals like SSH the other one is used for interactive terminals without authentication (console windows).

Execute this or use your favorite text editor.

Nano `~/.bash_profile`

nano ~/.bashrc

Add the lines below to each file:

```
PATH=/usr/lib/openmpi/1.2.5-  
gcc/bin:/usr/sbin:/usr/bin:/sbin:/bin:/mpiexec:$HOME/bin:$PATH  
LD_LIBRARY_PATH=/usr/lib:/usr/lib/openmpi/1.2.5-  
gcc/lib:$LD_LIBRARY_PATH
```

```
export PATH  
export LD_LIBRARY_PATH  
unset USERNAME
```

It is also possible to add paths in an individual way like this:

```
PATH=$PATH:HOME/bin  
PATH=/usr/lib/openmpi/1.2.5-gcc/bin:$PATH  
PATH=/usr/sbin:$PATH  
PATH=/sbin:$PATH  
PATH=/bin:$PATH  
PATH=/mpiexec:$PATH  
  
LD_LIBRARY_PATH=/usr/lib/openmpi/1.2.5-gcc/lib:$LD_LIBRARY_PATH  
LD_LIBRARY_PATH=/usr/lib:$LD_LIBRARY_PATH  
  
export PATH  
export LD_LIBRARY_PATH  
unset USERNAME
```

Hostname Configuration.

The `/etc/hosts` file contains the IP address/hostname relation for all the nodes or computers in the network:

```
# Do not remove the following line, or various programs  
# that require network functionality will fail.  
127.0.0.1    localhost.localdomain localhost  
::1         localhost6.localdomain6 localhost6  
192.168.2.4 pelusa.perrera.local pelusa
```

192.168.2.6 auyin.perrera.local auyin
192.168.2.5 rufian.perrera.local rufian
192.168.2.2 lamparita.perrera.local lamparita

Since we are going to modify the default hostname in the `/etc/sysconfig/network` file we must also include a line that specifies each nodes IP address/hostname as redundant as it may look, some services like NFS, STATD, CUPS, SENDMAIL y SM-CLIENT need that line and if it is missing the operating system boot can take up to 30 minutes (I'm not sure if that is a good thing to do and if someone knows a better way to fix that please advice me).

Hostname configuration.

The `/etc/sysconfig/network` file specifies each machine or node hostname like this:

```
NETWORKING=yes  
HOSTNAME=pelusa.perrera.local  
DOMAINNAME=perrera.local
```

Network configuration.

Network configuration isn't complex by convenience you should modify the `ifcfg-eth0`, or, `ifcfg-wlan0` files located in `/etc/sysconfig/network-scripts/`. I recommend not using DHCP since it may end up assigning arbitrary IP addresses that will break some cluster configuration parameters.

In this case since I used a wireless connection among all the nodes I disabled the eth0 interface in the `ifcfg-eth0` file like this:

```
DEVICE=eth0  
  
BOOTPROTO=dhcp  
HWADDR=XX:XX:XX:XX:XX:XX  
ONBOOT=no
```

`XX:XX:XX:XX:XX:XX` correspond to the MAC address of each network card. For security reasons I'm not putting my addresses.

The *ifcfg-wlan0* file will contain this:

```
DEVICE=wlan0  
BOOTPROTO=static  
BROADCAST=192.168.2.255  
IPADDR=192.168.2.4  
NETMASK=255.255.255.0  
HWADDR=XX:XX:XX:XX:XX:XX  
ONBOOT=yes
```

Operating System tuning.

Before installing and configuring the cluster programs and services we must prepare the operating system. Yellow Dog 6.1 is far from perfect and it has some annoyances you could use FEDORA OPENSUSE or even UBUNTU if you prefer but this one is yellowdog guide and if you are aiming to develop for the cell this is the officially supported operating system

According to this we must do four things to tune the operating system

- Fix the default WICD Manager so it can handle 802.11g with WPA2/PSK. (In my case I used wifi for the node interconnection)
- Fix the sound controller.
- Enable gpu memory as swap space to speed up the system
- Update the system

WICD wireless manager fix.

First you must download the next files

```
wicd-1.5.9.tar.gz  
wnettools.py.tar.gz for wicd-1.5.9
```

You can download the files from the addresses.

<http://downloads.sourceforge.net/wicd/wicd-1.5.9.tar.gz>
<http://www.yellowdog-board.com/download/file.php?id=62>

After downloading the **wicd-1.5.9.tar.gz** file you should put it in `/usr/local/src`, and then execute:

```
tar -xf wicd-1.5.9.tar.gz  
cd wicd-1.5.9  
python setup.py configure  
python setup.py install
```

In the case of **wicd-1.5.9.tar.gz** you must decompress and copy the **wicd-1.5.9** file in `/usr/lib/python2.4/site-packages/wicd/`,

You can reboot the system or continue with the sound controller fix and reboot later

Sound controller fix

To fix the sound controller you must download the **PS3.conf.alsa** file and rename it to **PS3.conf** and copy the file to `/etc/alsa/cards/` and `/usr/share/alsa/cards/`, if the cards folder doesn't exist you must create it.

Unfortunately the link to the **PS3.conf.alsa** file no longer works so you have to copy the source below into a file and name it **PS3.conf** and proceed as instructed above.

```
# Alsa configuration for PS3  
# 08.04.09  
# Place in your system's alsa cards directory:  
# /etc/alsa/cards/PS3.conf  
# /usr/share/alsa/cards/PS3.conf  
#
```

```
<confdir:pcm/front.conf>
```

```
PS3.pcm.front.0 {  
    @args [ CARD ]  
    @args.CARD {
```

```

        type string
    }
    type softvol
    slave.pcm {
        type hw
        card $CARD
        device 0
    }
    control {
        name "PCM Playback Volume"
        card $CARD
    }
}

```

Default to dmix + softvol

```

PS3.pcm.default {
    @args [ CARD ]
    @args.CARD {
        type string
    }
    type asym
    playback.pcm {
        type plug
        slave.pcm {
            type softvol
            slave.pcm {
                @func concat
                strings [ "dmix:CARD=" $CARD ",FORMAT=S16" ]
            }
            control {
                name "PCM Playback Volume"
                card $CARD
            }
        }
    }
}
}

```

GPU memory as swap space

To enable the GPU memory as swap space to speed up the operating system I recommend you to reboot if you didn't before during the previous steps and configure the Internet connection.

Log into a shell as root

su - (don't forget the -)

Input your password and execute one by one the commands below.

cd /etc/7ell.d

wget http://us.fixstars.com/support/solutions/vdl_6.x/ps3-vram-swap

chmod 755 ps3-vram-swap

chkconfig --add ps3-vram-swap

chkconfig ps3-vram-swap on

service ps3-vram-swap start

To verify the success execute:

swapon -s

You should get something like this:

<i>Filename</i>	<i>Type</i>	<i>Size</i>	<i>Used</i>	<i>Priority</i>
<i>/dev/ps3da3</i>	<i>partition</i>	<i>522104</i>	<i>12196</i>	<i>-1</i>
<i>/dev/mtdblock0</i>	<i>partition</i>	<i>241656</i>	<i>0</i>	<i>1</i>

Cluster tools.

To deploy a very basic cluster without fancy administration tools or schedulers you need openMPI NFS and SSH.

Some Systems already include those tools but if not, you must install them by using the respective OS facilities like "apt-get" for Debian based operating systems or "yum" for yellow dog Linux or download the source and compile by yourself.

In our case we use Yum since we are using Yellow dog Linux.

Execute this commands bellow separately to search for installed tools:

```
yum search nfs
```

```
yum search openmpi
```

```
yum search ssh
```

If for some reason they are not installed you must execute one by one the commands below in order to install the tools (you must have an active Internet connection):

```
yum install nfs-utils
```

```
yum install openmpi
```

```
yum install opensshssh-clients
```

```
yum install openssh-server
```

SSH configuration.

OpenMPI requires transparent SSH access to all the slave nodes in the cluster, in order to do that we must generate a public “passwordless” key in every slave node and then export it to the master node.

In most cases you will only have one display for your cluster unless you have money to throw up to the sky. If you want to avoid the hassle of unplugging and plugging you display across every node you must generate keys to and from each node of your cluster since you will be jumping across them and trust me it will become very annoying to input the password every time.

This can be done directly into each machine or remotely by using SSH itself, weather you decided to do it directly from each node or by login into each node remotely with from the master node you must execute as root and even as the administrator account in every node:

```
ssh-keygen -t rsa
```

This command generates both the public and private key using RSA cryptography. There are other cryptography methods like DSA but they are obsolete and should only be used if you are working with systems that don't support anything different to DSA.

After executing the command the system will ask for a location to store the keys, just press enter to use the default location `~/.ssh`. Then it will ask for a password and confirmation to which you will press enter without putting anything else (remember passwordless)

Generating public/private rsa key pair.

Enter file in which to save the key [/root/.ssh/id_rsa]:

Enter passphrase (empty for no passphrase):

Enter passphrase again:

Your identification has been saved in /root/.ssh/id_rsa.

Your public key has been saved in /root/.ssh/id_rsa.pub.

Once the keys are created you must export them to the master node. The most secure way to do that is by copying the `id_rsa.pub` keys in a USB memory and the using the "cat" command dump its content to the `~/.ssh/authorized_keys` file. Remember to use the "cat" command only because if you use a text editor it will ignore some characters and render you key useless.

The other and easy way to do that is executing the ***ssh-copy-id*** from each node to the rest of the nodes like this:

ssh-copy-id -i ~/.ssh/id_rsa.pub [root@pelusa](#)

this command will copy the keys into the remote machine (master node) automatically, but you cannot depend on this command since not all systems support it.

After this we proceed as if we were using the machine directly. At the end we must reboot the whole system or just the service.

If you did it remotely you must log on using SSH

To reboot:

shutdown -r now

To close the SSH session if you did remotely:

logout

To restart the ssh service:

/etc/10ell.d/ssh restart, or, ***service ssh restart***

Finally to test everything up we must “log into every node from every node” like this:

ssh rufian

If after executing the command we are not prompted for a password or user name and we are logged in the remote machine it means everything went according the plan.

NFS configuration for file sharing.

In order to share executable files across the cluster we are going to use NFS. To do so we must configure the service by adding the parameters below to the ***/etc/exports*** file which specifies the shared resources and other things

/mpiexec rufian(rw,sync,no_root_squash) auyin(rw,sync,no_root_squash)

The first parameter is the folder path we are going to share. “Rufian” and “Auyin” are the slave nodes hostnames. The remaining parameters meaning can be checked in the respective man pages.

As always you can change the hostnames to those who suit you better or with IP addresses for the whole network or for each node, like this:

/mpiexec 192.168.2.0/255.255.255.0(rw,sync,no_root_squash)

Or

```
/mpiexec 192.168.2.0/24(rw,sync,no_root_squash)
```

Or

```
/mpiexec 192.168.2.2/24(rw,sync,no_root_squash)
```

```
/mpiexec 192.168.2.5/24(rw,sync,no_root_squash)
```

Automatic remote folder mounting configuration

If we leave everything like this we will end up manually mounting the `/mpiexec` folder or worst copy the mpi executable node by node. Fortunately we have FSTAB to ease things up

To use fstab you must modify `/etc/fstab` file. You must be very careful when modifying this file and not touching any of the default parameters since you may end up screwing up the operating system boot and fixing it is not easy if you used the OpenSUSE boot manager.

To open the file you must execute one of the following commands depending of your favorite text editor

```
nano /etc/fstab or vi /etc/fstab
```

At the end of the file we will add the following parameters to mount the `/mpiexec` folder which will contain the MPI executable files.

```
Pelusa:/mpiexec /mpiexec nfs rw,hard,intr 0 0
```

“Pelusa” is the main node hostame, which can be replaced for an IP address or the name that suits you better

The `:/mpiexec` parameter specifies the shared folder in the main node and the next specifies the local path to mount the remote folder. Next is the file type that in this case is NFS. The other parameters meaning can be found in the “man pages”.

Services start up order.

If the nfs mounting fails it could be due to the service start up order. In my case since I used a wireless connection and the wicd manager is executed at last the nfs mounting was failing so I had to change the service start up process so NETFS was executed after WICD. Remember to change the timeserver sync too if your cluster needs it. (If you used a wired connection there is no need to change the services start up order)

To do this we must find out the boot process execution level that in yellow dog case is 5.

First we must go to */etc/rc5.d*, where the symbolic links to the services are in a level 5 execution boot. The format is *KXXservicename* or *SXXservicename* where XX indicates the numeric start order.

After the numeric order is verified the system proceeds in alphabetic order in case two services has the same number. In this case the service we are interested is *S25netfs* and *S98wicd*. First we rename *S25netfs* to *S98netfs* by executing:

```
mv S25nfs S98netfs
```

Next we rename *S98wicd* to *S97wicd* by executing:

```
mv S98wicd S97wicd
```

In theory any order is fine as long as WICD is executed before NETFS

OpenMPI configuration.

In the master node you must modify the */usr/lib/openmpi/1.2.5-gcc/etc/openmpi-default-hostfile* file, this file contains a list of available node and the amount of processors or cores like this (remember we are using hostnames instead of IP addresses for practical reasons)

```
auyin slots=2
```

```
rufian slots=2
```

pelusa slots=2

Although the Playstation 3 has 8 processors you must remember that the cell is an asymmetric processor and GNU/Linux as well as OpenMPI can only use the main processor since is basically a regular two-core processor with some changes. (It doesn't have on the fly instruction reordering so it depends on the compiler and the programmer) in order to use the 8 cores (6 actually) there must be some big changes in OpenMPI to support it.

Parallel execution.

The code that we are using is very simple, there are other examples around but this one is very useful since it clearly shows the nodes that executed some load

```
#include <stdio.h>  
#include <mpi.h>  
  
int main(int argc, char *argv[]) {  
int numprocs, rank, 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, &rank);  
MPI_Get_processor_name(processor_name, &namelen);  
  
printf("Process %d on %s out of %d\n", rank, processor_name,  
numprocs);  
  
MPI_Finalize();  
}
```

To execute the program you just need to copy the source above in an empty file (you can use gedit) and name it with a .c as suffix and then compile it by executing this:

(You may need to put the source file and go to the `/usr/lib/openmpi/1.2.5-gcc/bin` folder if you didn't add the path to the environment variables)

mpicc -o hello hello.c

Once this is done the executable file must be moved or copied to */mpiexec*:

```
mv hello /mpiexec
```

Finally when everything is installed and configured you can run your MPI program by executing *mpirun* like this:

```
mpirun -np 8 /mpiexec/hello
```

The *-np* option specifies the number of processes and the next part specifies the mpi executable path which in our case is named "hello"