

2024-07-10

OpenHPC: Beyond the Install Guide

OpenHPC: Beyond the Install Guide
for PEARC24

Sharon Colson Jim Moroney Mike Renfro

Tennessee Tech University

2024-07-22

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Introduction

└─ Acknowledgments and shameless plugs

└─ Acknowledgments and shameless plugs

x

Acknowledgments and shameless plugs

[OpenHPC](#) especially Tim Middelkoop (Internet2) and Chris Simmons (Massachusetts Green High Performance Computing Center). They have a BOF at 1:30 Wednesday. You should go to it.

[Jetstream2](#) especially Jeremy Fischer, Mike Lowe, and Julian Pistorius. Jetstream2 has a tutorial at the same time as this one. Please stay here.

[NSF CC*](#) for the equipment that led to some of the lessons we're sharing today (award #2127188).

[ACCESS](#) current maintainers of the project formerly known as the XSEDE Compatible Basic Cluster.

2024-07-10

OpenHPC: Beyond the Install Guide

└ Introduction

└ Where we're starting from

└ Where we're starting from

x

Where we're starting from



Figure 1: Two example HPC networks

31 HPC clusters (2 shown) with:

1. Rocky Linux 9
2. OpenHPC 3
3. Warewulf 3
4. Slurm
5. 2 non-GPU nodes
6. 2 GPU nodes (currently without GPU drivers, so: expensive non-GPU nodes)
7. 1 management node (SMS)
8. 1 unprovisioned login node

OpenHPC: Beyond the Install Guide

└─ Introduction

└─ Where we're starting from

└─ Where we're starting from

x

Where we're starting from

We used the OpenHPC automatic installation script from Appendix A with a few variations:

1. Installed `a-mail` to have a valid `MailProg` for `slurm.conf`.
2. Created `user1` and `user2` accounts with password-less `sudo` privileges.
3. Changed `CHROOT` from `/opt/ohpc/admin/images/rocky9.3` to `/opt/ohpc/admin/images/rocky9.4`.
4. Enabled `slurmd` and `slurmgs` in `CHROOT`.
5. Added `nanos` and `yum` to `CHROOT`.
6. Removed a redundant `ReturnToService` line from `/etc/slurm/slurm.conf`.
7. Stored all nodes' SSH host keys in `/etc/ssh/ssh_known_hosts`.

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Introduction

└─ Where we're going

└─ Where we're going

Where we're going

1. A login node that's practically identical to a compute node (except for where it needs to be different)
2. A slightly more secured SMS
3. GPU drivers on the GPU nodes
4. Using node-local storage for the OS and/or scratch
5. De-coupling the SMS and the compute nodes (e.g., independent kernel versions)
6. Easier management of node differences (GPU or not, diskless/single-disk/multi-disk, Infiniband or not, etc.)
7. Slurm configuration to match some common policy goals (fair share, resource limits, etc.)

x

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Making better nodes

└─ A dedicated login node

└─ Assumptions

Assumptions

1. We have a VM named `login`, with no operating system installed.
2. The `eth0` network interface for `login` is attached to the internal network, and `eth1` is attached to the external network.
3. The `eth0` MAC address for `login` is known—check the **Login server** section of your handout for that. It's of the format `aa:bb:cc:dd:ee:ff`.
4. We're logged into the SMS as `user1` or `user2` that has `nudo` privileges.

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ Creating a new login node

x

Creating a new login node

Working from section 3.9.3 of the install guide:

```
[user@node-0 ~]$ sudo wvash -y node new login --netdev with0 \
--ipaddr=172.16.0.2 --hwaddr=__:__:__:__:__:__
[user@node-0 ~]$ sudo wvash -y provision set login \
--pdrp=rocky9.4 --bootstrap=uname -r \
--files=dynamic_hosts,passwords,group,shadow,munge.key,network
```

Make sure to replace the __ with the characters from your login node's MAC address!

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ What'd we just do?

x

What'd we just do?

Ever since login was powered on, it's been stuck in a loop trying to PXE boot. What's the usual PXE boot process for a client in an OpenHPC environment?

1. The client network card tries to get an IP address from a DHCP server (the SMS) by broadcasting its MAC address.

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ What'd we just do?

x

What'd we just do?

Ever since login was powered on, it's been stuck in a loop trying to PXE boot. What's the usual PXE boot process for a client in an OpenHPC environment?

1. The client network card tries to get an IP address from a DHCP server (the SMS) by broadcasting its MAC address.
2. The SMS responds with the client's IP and network info, a next-server IP (the SMS again), and a filename option (a bootloader from the iPXE project).

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ What'd we just do?

x

What'd we just do?

Ever since login was powered on, it's been stuck in a loop trying to PXE boot. What's the usual PXE boot process for a client in an OpenHPC environment?

1. The client network card tries to get an IP address from a DHCP server (the SMS) by broadcasting its MAC address.
2. The SMS responds with the client's IP and network info, a next-server IP (the SMS again), and a filename option (a bootloader from the IPXE project).
3. The network card gets the bootloader over TFTP and executes it.

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Making better nodes

└─ A dedicated login node

└─ What'd we just do?

x

What'd we just do?

Ever since login was powered on, it's been stuck in a loop trying to PXE boot. What's the usual PXE boot process for a client in an OpenHPC environment?

1. The client network card tries to get an IP address from a DHCP server (the SMS) by broadcasting its MAC address.
2. The SMS responds with the client's IP and network info, a next-server IP (the SMS again), and a filename option (a bootloader from the iPXE project).
3. The network card gets the bootloader over TFTP and executes it.
4. iPXE makes a second DHCP request and this time, it gets a URL (by default, [http://SMS_IP/Win/tpxe/cfg/\\${client_mac}](http://SMS_IP/Win/tpxe/cfg/${client_mac})) for an iPXE config file.

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Making better nodes

└─ A dedicated login node

└─ What'd we just do?

x

What'd we just do?

Ever since login was powered on, it's been stuck in a loop trying to PXE boot. What's the usual PXE boot process for a client in an OpenHPC environment?

1. The client network card tries to get an IP address from a DHCP server (the SMS) by broadcasting its MAC address.
2. The SMS responds with the client's IP and network info, a next-server IP (the SMS again), and a `tftpname` option (a bootloader from the iPXE project).
3. The network card gets the bootloader over TFTP and executes it.
4. iPXE makes a second DHCP request and this time, it gets a URL (by default, `http://SMS_IP/VA/tftp/cfg/${client_mac}`) for an iPXE config file.
5. The config file contains the URL of a Linux kernel and initial ramdisk, plus multiple kernel parameters available after initial bootup for getting the node's full operating system contents.

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ What'd we just do?

What'd we just do?

1. The node name, `--baddr`, and `--ipaddr` parameters go into the SMS DHCP server settings.

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ What'd we just do?

What'd we just do?

1. The node name, `--baddr`, and `--ipaddr` parameters go into the SMS DHCP server settings.
2. The `--bootstrap` parameter defines the kernel and ramdisk for the IPXE configuration.

x

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Making better nodes

└─ A dedicated login node

└─ What'd we just do?

x

What'd we just do?

1. The node name, `--baddr`, and `--ipaddr` parameters go into the SMS DHCP server settings.
2. The `--bootstrap` parameter defines the kernel and ramdisk for the PXE configuration.
3. The node name, `--swdev`, `--ipaddr`, `--baddr` parameters all go into kernel parameters accessible from the provisioning software.

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Making better nodes

└─ A dedicated login node

└─ What'd we just do?

x

What'd we just do?

1. The node name, `--baddr`, and `--ipaddr` parameters go into the SMS DHCP server settings.
2. The `--bootstrap` parameter defines the kernel and ramdisk for the IPXE configuration.
3. The node name, `--swtdev`, `--ipaddr`, `--baddr` parameters all go into kernel parameters accessible from the provisioning software.
4. During the initial bootup, the `--baddr` parameter is passed to a CGI script on the SMS to identify the correct VNFS for the provisioning software to download (set by the `--vzfs` parameter).

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Making better nodes

└─ A dedicated login node

└─ What'd we just do?

x

What'd we just do?

1. The node name, `--baddr`, and `--ipaddr` parameters go into the SMS DHCP server settings.
2. The `--bootstrap` parameter defines the kernel and ramdisk for the IPXE configuration.
3. The node name, `--rootdev`, `--ipaddr`, `--baddr` parameters all go into kernel parameters accessible from the provisioning software.
4. During the initial bootup, the `--baddr` parameter is passed to a CGI script on the SMS to identify the correct VNFS for the provisioning software to download (set by the `--vzfs` parameter).
5. After downloading the VNFS, the provisioning software will also download files from the SMS set by the `--files` parameter.

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ Did it work? So far, so good.

Did it work? So far, so good.

```
[user@node-0 ~]$ sudo ssh login
[root@login ~]# df -h
Filesystem
...
172.16.0.1:/home
172.16.0.1:/opt/ohpc/pub
```

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ Did it work? Not entirely.

x

Did it work? Not entirely.

```
[root@login ~]# sinfo
sinfo: error: resolve_ctls_from_dns_srv: res_nsearch error:
Unknown host
sinfo: error: fetch_config: DNS SRV lookup failed
sinfo: error: _establish_config_source: failed to fetch config
sinfo: fatal: Could not establish a configuration source
```

systemctl status slurm is more helpful, with
fatal: Unable to determine this slurm's NodeName. So how do we fix this one?

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ Option 1: take the error message literally

x

Option 1: take the error message literally

So there's no entry for login in the `SMS azure.conf`. To fix that:

1. Run `slurmstat -C` on the login node to capture its correct CPU specifications. Copy that line to your laptop's clipboard.

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Making better nodes

└─ A dedicated login node

└─ Option 1: take the error message literally

x

Option 1: take the error message literally

So there's no entry for login in the SMS `slurm.conf`. To fix that:

1. Run `slurm -C` on the login node to capture its correct CPU specifications. Copy that line to your laptop's clipboard.
2. On the SMS, run `nano /etc/slurm/slurm/slurm.conf` and make a new line of all the `slurm -C` output from the previous step (pasted from your laptop clipboard).

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Making better nodes

└─ A dedicated login node

└─ Option 1: take the error message literally

x

Option 1: take the error message literally

So there's no entry for login in the SMS `slurm.conf`. To fix that:

1. Run `slurmstat -C` on the login node to capture its correct CPU specifications. Copy that line to your laptop's clipboard.
2. On the SMS, run `nano /etc/slurm/slurm/slurm.conf` and make a new line of all the `slurmstat -C` output from the previous step (pasted from your laptop clipboard).
3. Save and exit nano by pressing `Ctrl-X` and then `Enter`.

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Making better nodes

└─ A dedicated login node

└─ Option 1: take the error message literally

x

Option 1: take the error message literally

So there's no entry for login in the SMS `slurm.conf`. To fix that:

1. Run `slurm -C` on the login node to capture its correct CPU specifications. Copy that line to your laptop's clipboard.
2. On the SMS, run `nano /etc/slurm/slurm/slurm.conf` and make a new line of all the `slurm -C` output from the previous step (pasted from your laptop clipboard).
3. Save and exit nano by pressing `Ctrl-X` and then `Enter`.
4. Reload the new Slurm configuration everywhere (well, everywhere functional) with `nano scontrol reconfigure` on the SMS.

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Making better nodes

└─ A dedicated login node

└─ Option 1: take the error message literally

x

Option 1: take the error message literally

So there's no entry for login in the SMS `slurm.conf`. To fix that:

1. Run `slurm -C` on the login node to capture its correct CPU specifications. Copy that line to your laptop's clipboard.
2. On the SMS, run `nano /etc/slurm/slurm/slurm.conf` and make a new line of all the `slurm -C` output from the previous step (pasted from your laptop clipboard).
3. Save and exit nano by pressing `Ctrl-X` and then `Enter`.
4. Reload the new Slurm configuration everywhere (well, everywhere functional) with `sudo scontrol reconfigure` on the SMS.
5. `ssh` back to the login node and restart `slurmd`, since it wasn't able to respond to the `scontrol reconfigure` from the previous step (`sudo ssh login ssystemctl restart slurmd` on the SMS).

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ Option 1: take the error message literally

Option 1: take the error message literally

Now an `sinfo` should work on the login node:

```
[root@login ~]# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
normal*    up   1-00:00:00        1  idle  cl
```

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ Option 2: why are we running slurmd anyway?

Option 2: why are we running `slurmd` anyway?

The `slurmd` service is really only needed on systems that will be running computational jobs, and the login node is not in that category.

Running `slurmd` like the other nodes means the login node can get all its information from the SMS, but we can do the same thing with a very short customized `slurm.conf` with two lines from the SMS' `slurm.conf`:

```
ClusterName=cluster
SlurmctldHost=zma-0
```

(where `zma-0` should be **your** SMS hostname from your handout) and stopping/disabling the `slurmd` service.

x

OpenHPC: Beyond the Install Guide

- └─ Making better nodes
 - └─ A dedicated login node
 - └─ Interactive testing

x

Interactive testing

1. On the login node as root, temporarily stop the slurm service with `systemctl stop slurm`
2. On the login node as root, edit `/etc/slurm/slurm.conf` with `nano /etc/slurm/slurm.conf`
3. Add the two lines to the right.
4. Save and exit nano by pressing `Ctrl-X` and then Enter.

Verify that `sinfo` still works without `slurm` and with the custom `/etc/slurm/slurm.conf`.

```
[root@login ~]# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
normal*   up 1-00:00:00      1  idle  c1
```

```
/etc/slurm/slurm.conf on login
node
```

```
ClusterName=cluster
SlurmctldHost=ams-0
```

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ Making permanent changes from the SMS

Making permanent changes from the SMS

Let's reproduce the changes we made interactively on the login node in the Warewulf settings on the SMS.

For the customized `slurm.conf` file, we can keep a copy of it on the SMS and add it to the Warewulf file store.

We've done that previously for files like the shared `munge.key` for all cluster nodes (see section 3.8.5 of the OpenHPC install guide).

We also need to make sure that file is part of the login node's provisioning settings.

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ Making permanent changes from the SMS

Making permanent changes from the SMS

On the SMS:

```
[user@smc-0 ~]$ sudo scp login:/etc/slurm/slurm.conf \
/etc/slurm/slurm.conf.login
slurm.conf                                100% 40    87.7KB/s   00:00
[user@smc-0 ~]$ sudo wvsh -y file import \
/etc/slurm/slurm.conf.login --name=slurm.conf.login \
--path=/etc/slurm/slurm.conf
```

Now the file is available, but we need to ensure the login node gets it. That's handled with wvsh provision.

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ A quick look at wvsh provision

A quick look at wvsh provision

What are the provisioning settings for compute node c1?

```
[username@0 ~]$ wvsh provision print c1
#### c1 #####
ci: MASTER           = UNDEF
ci: BOOTSTRAP        = c1.96-1.el9.elrepo.x86_64
ci: VMFS              = rocky9.4
ci: VALIDATE         = FALSE
ci: FILES            = dynamic_hosts.group.munge.key.network,
                        passwd.shadow
...
ci: KARGS             = "net.ifnames=0 biosdevname=0 quiet"
ci: BOOTLOCAL        = FALSE
```

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ A quick look at wvsh provision

A quick look at wvsh provision

What are the provisioning settings for node login?

```
[username=0 -] $ wvsh provision print login
### login #####
login: MASTER      = UNDEF
login: BOOTSTRAP    = 0.1.50-1.el9.elrepo.x86_64
login: VNF2         = rocky9.4
login: VALIDATE     = FALSE
login: FILES        = dynamic_hosts,group,range,key,network,
                    passwd,shadow
...
login: XARGS        = "set.ifnames=0 biosdevname=0 quiet"
login: BOOTLOCAL    = FALSE
```

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ A quick look at `wwsh` provision

[A quick look at `wwsh` provision](#)

The provisioning settings for `c1` and `login` are identical, but there's a lot to read in there to be certain about it.

We could run the two outputs through `diff`, but every line contains the node name, so **no lines are literally identical**.

Let's simplify and filter the `wwsh` provision output to make it easier to compare.

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A dedicated login node

- └─ Filtering `wwsh provision` output

x

Filtering `wwsh provision` output

► I only care about the lines containing `= signs`, so

```
wwsh provision print cl | grep "="
```

is a start.

2024-07-10

OpenHPC: Beyond the Install Guide

└─ Making better nodes

└─ A dedicated login node

└─ Filtering `wwsh provision` output

x

Filtering `wwsh provision` output

► I only care about the lines containing `= signs`, so

```
wwsh provision print c1 | grep "="
```

is a start.

► Now all the lines are prefixed with `c1::`, and I want to keep everything after that, so

```
wwsh provision print c1 | grep "=" | cut -d: -f2-
```

will take care of that.

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ A bit more security for the SMS

- └─ A bit more security for the SMS

A bit more security for the SMS

(going to talk about fail2ban here, maybe also firewall)

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ Semi-stateful node provisioning

- └─ Semi-stateful node provisioning

Semi-stateful node provisioning

(talking about the parted and filesystem-related pieces here.)

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Making better nodes

- └─ Management of GPU drivers

- └─ Management of GPU drivers

Management of GPU drivers

(installing GPU drivers – mostly rsync'ing a least-common-denominator chroot into a GPU-named chroot, copying the NVIDIA installer into the chroot, mounting /proc and /sys, running the installer, unmounting /proc and /sys, and building a second VNFs)

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Managing system complexity)

- └─ Configuration settings for different node types

- └─ Configuration settings for different node types

Configuration settings for different node types

(have been leading into this a bit with the `wwsh` file entries, `systemd` conditions, etc.
But here we can also talk about nodes with two drives instead of one, nodes with and without Infiniband, nodes with different provisioning interfaces, etc.)

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Managing system complexity)
 - └─ Automation for Warewulf3 provisioning
 - └─ Automation for Warewulf3 provisioning

Automation for Warewulf3 provisioning

(here we can show some sample Python scripts where we can store node attributes and logic for managing the different VNFSees)

x

2024-07-10

OpenHPC: Beyond the Install Guide

- └─ Configuring Slurm policies

- └─ Configuring Slurm policies

[Configuring Slurm policies](#)

Can adapt a lot of Mike's CaRCC Emerging Centers talk from a couple years ago for this. Fair share, hard limits on resource consumption, QOSes for limiting number of GPU jobs or similar.

x

OpenHPC: Beyond the Install Guide

└─ Configuring Slurm policies

└─ Sample slide

This is my note.

- It can contain Markdown
- like this list

Left column

This slide has two columns. They don't always have to have columns. It also has a titled block of content in the left column. Make sure you've always got a `::: notes` block after the slide content, even if it has no content.

Use `#` and `##` headers in the Markdown file to make level-1 and level-2 headings. `###` headers to make slide titles, and `####` to make block titles.