Perceus 1.6 User Guide

www.Infiscale.com

June 7, 2010

0.1 Introduction

Perceus is the next generation cluster and enterprise tool kit for the deployment, provisioning, and management of groups of servers. Employing the power of the Perceus OS and framework, the user can quickly purpose a machine out of the box. Perceus truly makes the computer a commodity, allowing an organization to manage large quantities of machines in a scalable fashion.

This document is designed to satisfy the general needs of the Perceus user. The full manual, programming, building, and integration guides are available from Infiscale.com as a licensed product. Licensing Perceus and/or these documents from Infiscale.com supports the development of Infiscale's open source projects.

0.1.1 Infiscale.com

Perceus is developed and provided to the world under the GNU GPL by Infiscale.com. Infiscale.com has a multitude of product and service offerings based around Perceus, and we encourage you to help support continued development and enhancement of Perceus by supporting Infiscale.com!

Contents

Chapter 1

About Perceus

Created by the developers of Warewulf, one of the most widely-utilized Linux cluster toolkits and the de facto standard open source stateless solution for many years now, Perceus redefines the limits of scalability, flexibility, and simplicity. Perceus was architected from its inception to leverage the knowledge and experience gained from the original Warewulf project while at the same time addressing some of its perceived shortcomings. This unique vantage point enables us to deliver what we feel is the definitive solution for all cluster and highly-scalable platforms regardless of their size or specific requirements.

While other cluster toolkit solutions have traded ease of use for scalability, flexibility, and/or simplicity, Perceus leverages a revolutionary new design which makes it extremely well-suited to fit the needs of a wide array of environments, from highly-scalable and performance-centric clustering to new system provisioning. Without making any sacrifices, Perceus can be extremely simple to use and "turnkey," or it can be used as a development-enabling platform designed for very specialized roles.

Additional information, news, resources and downloads can be found at:

`http://www.perceus.org/`

1.1 Perceus Terms and Definitions

Before we begin, lets go over a few terms which will be commonly referred to throughout this document.

Node: Nodes are traditionally the slave systems in a clustered environment. This document will use the nomenclature "node" to describe all systems provisioned by Perceus.

VNFS: The Virtual Node File System (VNFS) is the template image that will be used to provision the nodes.

VNFS capsule: The capsule is the packaged up version of the VNFS which contains all of the necessary information for provisioning to properly take place.

VNFS image: When a node gets provisioned, the VNFS image is the actual filesystem that will be transported to and installed on each node. For example, changes to the VNFS itself would require updating the VNFS image.

Stateless/Stateful: These are the terms pertaining to the volatility of the operating system. If the operating system is never installed to local node media (e.g. a disk or flash card) and must be provisioned on every boot, then the operating system is considered stateless. It does not directly imply or correlate to diskful or diskless systems. As a matter of fact, if a stateless operating system has a local disk configured with an active swap partition, then the operating system may be eventually swapped to disk as needed by application priority to the local and fast memory (the operating system is still considered stateless even if it has been swapped to disk).

Filesystem hybridization: When dealing with stateless systems most people prefer to make the VNFS image as small as possible while still maintaining the functionality of a fully installed system. One of the tricks to do this is to have many of the less frequently-accessed programs, libraries, and data as non-local parts of the stateless image. For example, it is usually safe to assume that `/usr/share` and various parts of `/var` do not need to be locally consuming stateless memory. Thus these parts of the operating system can actually reside elsewhere and then symbolically linked back to the proper location. This model can be taken rather aggressively to conserve on memory space.

Modules: Perceus uses modules to enhance the functionality of node provisioning by directly linking into the node provisioning subsystem itself. Modules can also be used for doing things like deploying hardware-specific microcode updates, or BIOS flashes, during provisioning.

Provisionary state: When a node checks in, it communicates to the master Perceus server what stage in the provisioning process it is in. This is its provisionary state. For instance, when a node first boots, it will check in with the "init" state; when it is running, it may check in with the "ready" state. These states may be futher classified by including a grouping delineation; for example, "init/all" specifies all nodes in the "init" state, and "ready/group/cluster" specifies all nodes in the "cluster" group in the "ready" state.

Cluster: Historically a cluster is a group of systems working together for a common task or infrastructure. In this case, the documentation will use the term "cluster" to relate to the group(s) of nodes that are maintained by Perceus.

1.2 How Perceus Works (High-Level View)

Perceus functions like a two stage rocket. In the first stage the computer is booted into the Perceus operating system. While there are multiple methods to get the first stage to boot the node, the default and most common method utilizes the Preboot Execution Environment (PXE) service. Perceus is built with this model enabled. Another model involves running the Perceus OS as an embedded object on the motherboard of the node itself (e.g. Intel's Rapid Boot solution).

Once the first stage operating system has booted it will launch the Perceus client daemon which will locate and interact with the Perceus service on the master server, communicating to it its provisionary state. The Perceus server will then send the node the commands needed to provision the second stage (the runtime operating system).

From the perspective of the newly booted node, there is no state as to its configuration, operating system, or even designation in the cluster before booting into Perceus. The node is identified to the Perceus server by the network hardware (MAC) address associated with the interface the node used to communicate with the server; thus, the MAC address is the primary key for node identification within Perceus.

Once Perceus has established a relationship with a node, attributes can be set which will enable or disable particular features. These features can be in the form of VNFS specific options, Perceus modules or assorted node scripts.

Here are the logical events as they would occur for a node to be provisioned in a relatively simplistic environment:

1. The node would be configured to automatically boot using PXE over a network interface card that is directly connected to the Perceus master server.
2. The node's PXE implementation will request an IP address via DHCP. The DHCP response will contain the information required for PXE to download (via TFTP) the pxelinux and Perceus boot software.
3. The Perceus operating system (stage one) will boot and request a DHCP address (the PXE-assigned address is no longer applicable, so it must make another DHCP request) in order to connect to the Perceus server and communicate that the node is in the "init" provisionary state. This is the first time that Perceus will actually know about the node. (This means that, even if a node can PXE boot, if it doesn't make this connection to Perceus, that node is not configured).

4. The Perceus server will now add this as a new node if it doesn't already exist and will then assemble the command sequence and send that through the open socket to the node.
5. On a typical stateless VNFS, the command sequence will transfer the VNFS to the node and prepare it for running stateless. Once the filesystem environment has been prepared, Perceus will execute the run time kernel contained in the VNFS.
6. Once the run time kernel has been started, all of the stage one Perceus operating system gets purged from memory. The booting run time operating system has no knowledge of the previous environment and runs as normal.

1.3 Flexibility

By expanding on the proven VNFS technology invented by Warewulf, Perceus can provision hardware from bare metal with any of the premade VNFS capsules or a custom VNFS capsule that you can create for it. Some of its features include:

- Intuitive interface and design
- Extremely modular and customizable architecture
- Most popular freely available VNFS capsules supported and downloadable
- Capable of provisioning and managing systems from bare metal (local disks optional)
- Automatically detect "new" machines (nodes) as they boot
- Deploy a machine with a Linux OS in RAM or via a network filesystem (stateless) or to local disk statefully
- Scalably manage machines/nodes

The open architecture for Perceus allows for users and programmers to endlessly manipulate as much or as little of the provisioning process as they wish. Perceus can emulate other provisioning systems, or one can invent new and unique mechanisms from within Perceus.

Results of this flexibility have allowed Perceus to grow into areas that were not even conceived of during its original design. Some of these roles are:

- Scalable cloud infrastructure provisioning
- High Performance Computing Clusters (HPCC)
- Various other cluster environments: Large scale web services, database, filesystem, virtual machine (VMWare, Xen, KVM), utility computing, security/intrusion detection, etc...
- New system installations (workstations, servers, standard builds, etc..)
- Appliances and various network devices

1.4 System Requirements

1.4.1 Architectures

Supported architectures are x86 (ia32) and x86_64. There have been people that have ported Perceus to other platforms, but do this at your own risk. If you do however successfully do this, please share your notes with us upstream.

We have had requests to support ia64 and PPC64 as well, and we will do so when hardware and development resources become available.

1.4.2 Operating Systems

While there are plans on supporting other Operating Systems, presently Perceus is a Linux solution. Even more specifically, Perceus is developed and primarily tested on the following Operating Systems:

Operating systems including Perceus: While we support other operating systems, the operating system vendors that have included Perceus within their distribution have shown the greatest commitment to compatibility and thus get listed first.

- **GravityOS** is a Debian-based Linux distribution released by Infiscale to meet enterprise cloud and datacenter needs. It provides a tuned, stable, and secure platform for Perceus provisioning with easy acces to +40k mainstream binary packages. A variety of GravityOS VNFS capsules are available from Infiscale.
- **Caos NSA** is a distribution of Linux focused on doing what Linux already does best: high performance compute Nodes, Servers and Appliances. An automated bootstrapping configuration tool is included. Caos NSA is the only freely-available Linux distribution which is certified Intel Cluster Ready!

Supported Operating Systems: We (as users and developers) tend to develop on a specific set of operating systems depending on vendor partnerships, preference, and familiarity. These are the distributions that you will have the best luck of everything just working on (although by no means is Perceus specific to any of them).

- **Red Hat Enterprise Linux** RHEL 4 (update 5 or newer) and RHEL 5 are supported directly; compatibles/rebuilds like Scientific Linux or Centos are not thoroughly tested, but have been widely used in the community. .RPM binary packages are available on the Infiscale mirrors.
- **Debian** Perceus is supported on Debian Squeeze. Compatibility with Ubuntu is not directly supported, but has been sucessfully tested in the community. The Infiscale mirrors host .DEB binary packages.

Development in progress or coming soon: Some operating system developers and/or vendors are also working with us to include Perceus and maintain it for themselves. Among the publicly known vendors are:

- **Gentoo** and Infiscale are working together to include Perceus natively into the portage tree and create VNFS capsules.

Known to work on: Many people have gotten Perceus to work on other non-supported operating systems with little or no extra effort. These are considered "at your own risk", but at least you are not alone... If you find something that doesn't work quite right here, we would appreciate you letting us know so that others might not run into the same issue.

- **SuSE Linux** Perceus will compile and should run as the server perfectly on SuSE, and is known to have been done in the community. VNFS capsules are available from Infiscale.

1.4.3 Node Requirements

There are several layers of hardware support that is needed for Perceus to function correctly. First the system must support PXE (network boot subsystem). Then the Perceus initial ram filesystem (initramfs) itself must support the hardware. Lastly, the provisioned VNFS must also support the hardware.

We (Infiscale.com) are always interested in building relationships with hardware vendors, so please be sure to communicate with your particular hardware provider that you plan on integrating with Perceus and that they should create a relationship with us so that we can test and certify their hardware stack.

Chapter 2

Installation

2.1 Building the Ideal Cluster Architecture

Perceus takes a typical cluster provisioning approach to the network design: a master-server-to-slave-node relationship. This relationship is networked by one or more private networks used for DHCP, provisioning, and administration of the nodes in addition to other, non-Perceus related tasks.

2.1.1 The Simplest Form

The master node is connected to 2 networks, one being publicly accessible, the other being private and shared only among the nodes and devices that are part of the cluster itself (e.g., network-attached storage).

Literally the master node is connected to the public LAN/WAN on one port, and the next port is connected to a standalone switch or set of switches to which all nodes are connected.

2.1.2 Large-Scale Solution

Instead of using the topology above, the Perceus master(s) and all nodes can occupy a single private network with a dedicated firewall/gateway bridging the cluster-internal network to the LAN/WAN. This gateway can also provide either a proxy/login system permitting internal access or a port/packet forward from the external network to a login system on the private network.

You can even have multiple Perceus masters spread across either a single flat network or multiple subnetworks delineated physically or via VLAN's.

2.2 Architecting the Perceus Infrastructure

When designing a new cluster, it is good to be mindful of the nature of the solution being architected. For example, if this is a small cluster a single Perceus server may be all that is needed. If you plan on scaling the cluster, one must consider the limiting factors associated with the weakest points (which may not be where you think!).

Here are some general guidelines to consider when configuring Perceus. (While decisions made in advance may be reversed or reconsidered later, the required downtime is generally simpler to manage in advance.)

2.2.1 Single Perceus Master/Local State

The default out-of-the-box layout for Perceus involves a single Perceus master utilizing a local directory tree for maintaining state. While more complex configurations are possible, this document will assume and be based on the default layout.

The Perceus state directory will reside on a locally-mounted filesystem on the master itself. In this configuration, the Perceus master will need to either export out this directory to the nodes via NFS (default) or utilize Apache to distribute the VNFS content (non-default and less clean).

How many nodes this method supports is very dependent on the hardware and capabilities of the Perceus master. Theoretically a decent hardware configuration can support about 32 simultaneously booting nodes (this is not the total number of supported nodes, rather how many nodes can be provisioned exactly in parallel). Using a lightweight VNFS capsule, this makes provisioning a 512 node cluster possible in about 4 minutes. As long as you stagger the boots, this model will scale into the thousands of nodes.

Obviously this method is subject to variances in addition to hardware configuration as well. NFS implementation, specific NFS and kernel tuning, etc..

2.2.2 Single Perceus Master/Non-Local State

Similar to the above, the primary difference here is that another system is used for NFS. This could be either another Unix/Linux system or something like a NAS device or even a parallel filesystem. This can drastically increase the number of nodes that can be provisioned simultaneously with ranges highly dependent on the NAS storage system used.

2.2.3 Multiple Perceus Masters/Non-Local State

It is possible to run multiple Perceus masters all mounting the non-local state directory thus sharing all of the state and provisioning load.

2.3 Installation and Configuration of the Host Operating System

Assuming a rather default configuration of Perceus, you can use a standard operating system installation from any of the supported operating systems. If you are going to use an operating system that does not already include Perceus and you will need to build Perceus from source, it may be necessary to select the installation option to include development libraries and tools. (This is an operating-system-specific installation option, so it may or may not be needed depending on your choice of OS).

A typical installation would include multiple network interfaces: the first connected to the "outside world" and the second connected to a private network used solely for node communication. Perceus will run DHCP services on the private network, so it is very important to make sure that the networks are separate and that no other DHCP server is running on the private network.

Before installing Perceus, the host system should be updated and in full working order, including all hardware devices being present and configured. Configurations can always be changed later, but since Perceus reads the current hardware state of the host system, doing this before-hand is good practice.

While Perceus has no particular addressing or routing restrictions (and will instead assume the administrator has done things correctly), the recommended configuration consists of 1 or more network cards connected to each of the external network and the private (node) network. The private (or public) network interface can even be a bridged, bonded Ethernet device, a 10Gig, or an InfiniBand card. (Perceus supports provisioning directly over the IB fabric using IPoIB and a customized DHCP server.)

The network device intended for node communication should be on a non-routable IP address (as specified by RFC 1918). This device and any other static IP addresses should be kept apart from the IP address range

assigned by the Perceus DHCP server.

2.4 Perceus Installation

For the purposes of this document, it will be assumed that Perceus is being installed (or upgraded) on one of the earlier specified supported operating systems (Caos NSA, Caos, or RHEL). RPM will be used for building and installing the Perceus packages.

If you wish to do an installation from source, or if your distribution of choice already has the version of Perceus that you wish to use installed, then you can skip to the next section (Perceus Startup).

2.4.1 Installing Dependencies

There are several packages and libraries that must be installed on the server for Perceus to operate properly. Some of these may be included with your operating system of choice, and others might require manual installation.

- **NFS:** Required if you plan on implementing NFS based provisioning (which is default). If you want to do it otherwise, you can omit this requirement, but you are then straying from this document, and you have been warned.
- **NASM:** Required for building pxelinux support
- **Perl:** Much of Perceus is written in Perl.
- **Perl-Unix-Syslog:** Allows Perceus to output to syslog
- **Perl-IO-Interface:** Required for select based socket loop
- **Perl-Net-ARP (1.0 or newer):** Required for Perceus to look-up the node's MAC address. This is a major point of common fault because older versions utilize a different and non-compatible API. Make sure you use a recent version!
- **bash_completion:** Optional package so that Perceus commands can "auto-complete" on TAB press.

On Red Hat the last four dependencies are not included in the operating system itself (which may be the case in other distributions of Linux as well). These are available at

`http://www.perceus.org/downloads/perceus/v1.x/dependencies/`

If you download the above four packages in '*src.rpm' format, the following command sequence will build and install them using RPM on Red Hat compatible systems:

```
# echo "%debug_package %{nil}" >> ~/.rpmmacros
# rpmbuild --rebuild bash-completion*.src.rpm
# rpmbuild --rebuild perl-IO-Interface*.src.rpm
# rpmbuild --rebuild perl-Net-ARP*.src.rpm
# rpmbuild --rebuild perl-Unix-Syslog*.src.rpm
# cd /usr/src/redhat/RPMS/noarch/
# rpm -Uvh bash-completion*.noarch.rpm
# rpm -Uvh perl-IO-Interface*.noarch.rpm
# rpm -Uvh perl-Net-ARP*.noarch.rpm
# rpm -Uvh perl-Unix-Syslog*.noarch.rpm
```

2.4.2 Local vs. Remote State

If Perceus will utilize a remote state directory, this directory should be created and mounted on the master server before installation of Perceus begins. If this is not done, a local state directory will be created, and its data will have to be relocated to the remote directory at a later time.

For example, using a remote NFS server with the IP address 10.0.0.5, an exported Perceus state directory /vol/perceus, and a local mount point of /var/lib/perceus, the following line should be added to /etc/fstab:

```
10.0.0.5:/vol/perceus   /var/lib/perceus        nfs     defaults        1 2
```

The commands below will then mount the filesystem:

```
# mkdir -p /var/lib/perceus
# mount /var/lib/perceus
```

2.4.3 Building and Installing Perceus

Perceus uses GNU autotools for the majority of Makefile management, so building Perceus from source should be very familiar to administrators. It also includes the ability to build RPMS directly from the source tarball itself (which is the method used in this document).

Here is a build snippet for building Perceus on a Caos NSA system:

```
# export TAR_OPTIONS=--wildcards
# rpmbuild -ta perceus-@perceus_version@.tar.gz
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.68349
+ umask 022
+ cd /usr/src/rpm/BUILD
+ LANG=C
+ export LANG
+ unset DISPLAY
+ cd /usr/src/rpm/BUILD
 ...
Wrote: /usr/src/rpm/SRPMS/perceus-@perceus_version@-@perceus_build@.src.rpm
Wrote: /usr/src/rpm/RPMS/x86_64/perceus-@perceus_version@-@perceus_build@.x86_64.rpm
Wrote: /usr/src/rpm/RPMS/x86_64/perceus-provisiond-@perceus_version@-@perceus_build@.x86_64.
# rpm -ivh /usr/src/rpm/RPMS/x86_64/perceus-@perceus_version@-@perceus_build@.x86_64.rpm
```

Note: The version and/or release may be different for your installation. For example, Red Hat-compatible systems use the base path `/usr/src/redhat/` *when building RPMs.*

Chapter 3

Welcome to Perceus

3.1 Getting to Know Perceus

Perceus is much more than a simple command line application. It is a conglomeration of services, executables, and scripts in multiple languages designed to work closely with the operating system (both the host and the provisioned nodes). Understanding the integration of the different components of Perceus is critical for accurate and effective use of the software.

3.1.1 Filesystem Layout

Perceus stores its configuration, local state, libraries, and other miscellaneous files in various points throughout the filesystem. This is to support not only generally-accepted rules of filesystem layout but also to allow for redundancy and distribution of part or all of the Perceus structures.

The actual file locations can all be manipulated at build time by passing arguments to the GNU style "./configure" script. For example, if –prefix is given with no other location syntaxes, then all directories will be children of the defined prefix. By default RPM separates everything out into the core of the operating system (i.e., a prefix of `/usr`).

The primary configuration for Perceus itself is hard coded to `/etc/perceus` (even if you choose a different –prefix and/or –sysconfdir). Inside this directory, Perceus will expect to find several configuration components which are specific to the Perceus "instance" on the current system. (If multiple masters are being used, each one has its own instance.)

The Perceus state tree contains node-specific configuration, provisioning data, etc. The default location of `/var/lib/perceus` may be adjusted using the –localstatedir option to the configure script. (For example, `--localstatedir=/usr/var` would set the Perceus state directory to `/usr/var/lib/perceus`.) This location will also house the primary database directory, aptly called `database`.

Perceus modules, VNFS capsules, and node scripts are also stored in this state directory; however, links to the installed VNFS capsules and the node scripts will be located in /etc/perceus for convenience. Modules that require configuration files should create them in `/etc/perceus/modules`.

3.1.2 Command Line Tools

For the most part (several minor exceptions), your primary interaction with Perceus will be via the command line. The Perceus command line tool is capable of managing and administrating nearly all aspects of the Perceus realm.

Generally speaking the format is:

```
perceus [command] [subcommand] (arguments...)
```

The following is a brief introduction to the primary Perceus command line commands:

- **node:** Set, display and list node attributes and statistics
- **group:** Set, display and list group attributes and statistics
- **vnfs:** Import, delete, modify, sync VNFS capsules
- **module:** Import, activate, display Perceus modules
- **info:** Print various aspects about Perceus
- **configure:** Have Perceus configure required daemons
- **contact:** Perceus is capable of contacting the developers and maintainers directly.

3.1.3 Node and Group Name Range Operators

All Perceus commands which take a node or a group name as a parameter also accepts bracketed ranges and wild-card based "globs" similar to a traditional shell. For example, the following command will list the nodes n0000-n0199, skipping n0098, followed by nodes n1000-n1999.

```
# perceus node list n00[00-97,99-199] n1*
```

note: *Whenever using the above "range" or "glob" syntax, care must be taken to ensure that the shell does not apply special meaning to any metacharacters used. For example, if the above command is run in a directory that has files or subdirectories whose names match the pattern 'n1*' (e.g., "n1foo" or "n10"), the shell will replace the pattern with the matching names before executing perceus, and in turn perceus will see the filenames instead of the intended wildcard expression. To avoid this, either escape all metacharacters using backslashes or surround the entire range/glob expression in single quotes ('n1').*

3.1.4 Services and Daemons

Perceus will run 2 major services that are controlled with the init script `/etc/init.d/perceus`: the Perceus daemon and the Perceus network services manager. Neither service communicates with other directly, but Perceus links them via a shared backend and configuration interface.

The Perceus daemon: Perceus operates on a client/server concept which has the capability of governing not only how Perceus provisions nodes but also maintains the nodes post provisioning (via the node checking in with specific provisionary states).

In general the Perceus daemon is responsible for assembling the specific nodescript needed for each node (based on the node's configuration in the database) for each provisionary state at which it checks in.

The Perceus network manager: Perceus uses DNSMasq to provide a uniform, seamless internal network service interface. This service provides the essential functions for network booting (TFTP and DHCP) as well as dynamic DNS plus proxying.

This service is actually not specific to Perceus and runs independently of the Perceus core. DHCP addresses can be allocated to non-nodes, workstations, fully-installed systems, etc. without having an adverse affect on the Perceus node configurations.

When nodes (and the server even) set the primary nameserver in /etc/resolv.conf to that of the server itself (the master should be set to "`nameserver 127.0.0.1`" and place that before all other nameserver lines), name resolution will automatically be provided for every node that has booted regardless of whether its address was assigned dynamic or statically.

Configuring the DHCP server to assign and/or resolve static addresses is simply a matter of putting the appropriate entries into the master's /etc/hosts file and reloading the service (`/etc/init.d/perceus reload`).

Note: *Depending on your hostname/domain configuration, you may need to put the fully qualified host name into the hosts file and address the host by its full name.*

3.2 Perceus Startup

The first time Perceus is run, it will automatically configure itself, including setting up the required services and configuration files. (**Note:** If using Caos NSA, and the "Clustering" option in Sidekick has been selected, skip this entire section, including subsections. NSA will automatically build a general usable Perceus configuration and include commonly-used NFS exports needed for HPCC, utility computing, and Visualization clusters.)

3.2.1 Configuration

In the `/etc/perceus/` directory, you will find several configuration files of interest, including `perceus.conf` and `defaults.conf`.

- **/etc/perceus/perceus.conf:** This is the main Perceus configuration file. It contains information specific to the Perceus master service and should only be changed on a newly installed (i.e., not yet production) system. Once a system is in production, changes in this file can break things.

  Before Perceus can successfully run, you must define "`master network device`" so that Perceus knows where to run DHCP services and how to communicate with the nodes.

  The `vnfs transfer method` option specifies what method should be used for transferring the VNFS to each node. The default in 1.4 is `xget` which uses a very scalable tree-spawning propagation algorithm. Each client is also a server for other nodes which are still booting. When the node has completed its own download, it will wait to help provision other nodes before continuing the boot process. (This may result in slightly higher boot times for smaller quantities of nodes, however.) Other options are `nfs` and `http`.

- **/etc/perceus/defaults.conf:** The defaults configuration file gets read as new nodes are added to Perceus. This file can be modified at any time to change how new nodes are initialized. The `First Node` and `Total Nodes` parameters specify the node naming/numbering convention. If you want nodes to begin with 1 instead of the default 0, change the `First Node` value.

3.2.2 Initialization

An interactive setup routine is performed when Perceus is run for the first time. This routine goes through a simple registration and full initialization sequence.

The registration basically just tells us (the developers) who is using Perceus. While registration is certainly not required to use Perceus, we strongly ask people to do so anyway. The data collected from registrations help us better understand the environments in which Perceus is being used as well as provide contact information in case problems arise. This information is strictly for the Perceus developer's use only and will never be distributed, sold, or disseminated in any way to any other individual, company, or entity outside our organization, nor will it be used for purposes of unsolicited commercial or bulk e-mail.

After registration Perceus will guide the user through several basic questions about the Perceus environment. This will facilitate setting up the DHCP service and basic run time options. Perceus prompts for which Ethernet device on the master is connected directly and privately to the nodes. Assuming that the master was configured in a sane matter (as previously described), Perceus will offer reasonable defaults for the node booting ranges and a DNS resolution search path.

Automatically occurring tasks are:

1. Create the Perceus database which stores node configuration in a scalable manner
2. Create NFS exports and start NFS services (e.g., portmap, nfsd, etc.) if the configuration is set to use local NFS-based provisioning.
3. Create SSH host keys and make them available for VNFS capsules as they are imported.
4. Create SSH keys for the root user.
5. Start the Perceus services, and configure them to start on boot as well.

Calling Perceus for the first time will start this proceedure:

```
# perceus
```

Once the Perceus initialization has completed and you have been returned to your shell prompt, Perceus is ready to be used!

Chapter 4

Using Perceus

Perceus takes a very complex problem and solves it in a relatively simple manner. However, simplicity does not necessitate a lack of flexibility, and Perceus is extremely flexible and extensible in both design and implementation. The following sections contain detailed descriptions of the most common tasks involved in Perceus administration as well as specific examples to help solidify the concepts discussed.

4.1 Adding Nodes to Perceus

Once installed, configured, and initialized, Perceus is ready to start provisioning nodes. The daemon process will wait and listen for new nodes to boot or for requests for manual node configuration.

The simplest (but least scalable) method for adding nodes to Perceus is to simply boot them and let the Perceus daemon automatically add and configure them upon initial connection. This method works very well and is usually preferred for smaller clusters (under 128 nodes) or if the hardware MAC addresses for each node cannot be easily obtained. (Recall that Perceus keys off the MAC address of each node to distinguish the different nodes internally.)

The most scalable method is to obtain a list of the hardware (MAC) address of of each node. This can sometimes be provided by the hardware vendor (if requested) or can be obtained by engineers making BIOS adjustments if any are necessary. The list can be used to drive a script which imports each MAC address in order, thus preserving the sequence of nodes.

Adding the nodes to Perceus in the same order as they are mounted in the racks will be very beneficial for isolating and identifying nodes in the future. Best practices also dictate labeling them as they are configured. The more nodes, the more critical this is.

As nodes are added into Perceus, they are assigned a default configuration based on the file `/etc/perceus/defaults.conf`.

4.1.1 Using Node Defaults

Unless overridden, each new node provisioned by or configured in Perceus acquires the default configuration defined in `/etc/perceus/defaults.conf`. This file contains the following configuration variables:

- **Node Name:** Defines the node base name (such as ”node” or ”n”) and where the digits will go when numbering each node. The ’#’ character(s) symbolize the node number, one for each digit. The default value of ”n####” has 4 placeholders for digits and will result in 4-digit node numbers (e.g., ”0015”).
- **Group Name:** Nodes are associated into arbitrary Groups to aid in classification. While each node may have only one group at a time, groups may contain any number of nodes. Groups are primarily used for

performing actions in bulk. Group names may be created on the fly; they are automatically created by Perceus whenever a node is assigned to a group which does not yet exist, so there is no need to worry about creating groups.

- **Vnfs Name:** Specifies the VNFS image to be used by nodes as they are provisioned. This may or may not be the desired behavior; do not set this value if you wish to specify the VNFS for each node by hand.
- **Enabled:** This prevents the node from being provisioned; instead, the node will wait for this value to be defined and not zero.
- **First Node:** The number assigned to the first node is a somewhat religious topic. Some people prefer it start at zero (usually C programmers) and some people prefer that it begin with one (usually FORTRAN programmers). For those people whom do not fall into either of the above groups, you may set this number to any non-negative integer.
- **Total Nodes:** This parameter does not limit the number of nodes that your implementation of Perceus supports; rather, it instructs Perceus to only count this high when looking for new node names. If this number ends up having more digits than were allocated in the "Node Name" field (signified by the '#' characters), that space will be padded to make sure that there is always enough room to allocate.

4.1.2 Booting Nodes Directly into Perceus

When the nodes have booted into the Perceus first stage operating system, the hardware stack will be automatically detected, and drivers will be loaded for all supported interfaces. Perceus will then iterate over each network interface with an active link and send out DHCP requests over that interface (even InfiniBand interfaces). If any network interface gets a DHCP response, an attempt will be made to connect to the Perceus master (usually the same host as the DHCP server, or a pre-defined address implemented by Perceus itself).

When the Perceus server gets a connection from a client node, it will check to see if the node exists in its database. If this is a new node, it will automatically add the node to the database using the configuration options defined by `/etc/perceus/defaults.conf`. Nodes will then be provisioned if their respective configurations are set for provisioning (i.e., specify a VNFS and a transfer method).

When adding large numbers of nodes via this method, it is best to actually watch the console or Perceus itself to make sure that the node has been successfully added before continuing to the next node. Otherwise, if one or more nodes fail to boot (due to, for example, hardware failure), the remaining nodes could be incorrectly named. Since names are assigned by node number, and node numbers are assigned sequentially, each node's assigned name and number would be off by one or more places (depending on the number of nodes which failed to boot). This situation, while correctable, is best avoided.

4.1.3 Manual Addition of Nodes

There are multiple ways to add nodes via the command line. The first and most flexible is to use the Perceus command "`perceus node add [MAC ADDRESS]`". This command can be wrapped inside of a shell script to support reading from a file containing a list of MAC addresses:

```
cat /path/to/file.txt | while read MAC; do
   echo Adding $MAC
   perceus node add $MAC
done
```

When you add a node in this manner, all defaults are still taken from the `/etc/perceus/defaults.conf` file.

4.1.4 Adding Nodes from Other Sources

There are several scripts distributed as part of Perceus to facilitate adding nodes from existing clusters. The freely distributable ones can be found in the /usr/share/perceus/ directory. If you don't see anything suitable to your situation, contact us at `info@Infiscale.com`.

- **import-isc-dhcpd.pl:** This script tries to read a dhcpd.conf file containing static node address assignments in the following format:

```
host n0000 {
   hardware ethernet 00:00:00:00:00:00;
   fixed-address 10.0.1.0;
}
```

- **import-ethers.pl:** This will rather ignorantly scan syslog watching for DHCP requests and immediately add any request to be a new node. While this is dangerous to run because it blindly adds nodes, it is significantly faster for booting a lot of new systems quickly.
- **import-warewulf-2.6.pl:** This will import nodes from a Warewulf 2.6 node directory.
- **import-warewulf-2.4.pl:** This will import nodes from a Warewulf 2.4 node configuration file.
- **import-switch.pl:** With this you can query switches directly and add nodes in the order of the switch port assignment.
- **import-rocks.pl:** Convert a ROCKS cluster
- **import-osc.pl:** Convert a Platform OSC cluster
- **import-oscar.pl:** Convert an OSCAR cluster
- **import-xCAT.pl:** Convert an xCAT cluster
- **import-scm.pl:** Convert an IBM SCM cluster

Infiscale.com offers other import and conversion tools as well; contact the above e-mail address for more information.

4.2 Node Management

Managing nodes with Perceus is easy via the `perceus` command line interface. Providing a scalable, scriptable method of cluster administration, the `perceus` command empowers administrators to control single systems or thousands of nodes with equal speed and efficiency.

Some of the subcommands (status, summary, show, list, nodelist, etc.) don't require a node or group argument; when this argument is omitted, "all entries" are assumed (e.g., '*').

Note: *Take care to quote the wildcard operators (like '*') which most shells also try to expand. For example, a command intended to operate on all nodes whose names begin with 'n' might use "n*" as the node specifier. However, if the current directory contains any files or subdirectories whose names also begin with 'n', the shell will replace "n*" in the original command with the names of the matching directory entries. Consult your shell's documentation for more information on how to inhibit normal command line shell globbing.*

4.2.1 Viewing Node Information

node summary: The `node` command has a `summary` subcommand which displays an overall view of the nodes' configurations. Display includes Hostname, Group name, Enabled status, and the configured VNFS.

node status: As the nodes check in with the Perceus daemon, the daemon will update certain fields in the database (last seen and status). The `status` subcommand displays this information in a human-readable format.

Perceus allows nodes to "time out" after having not checked in for some period of time. This value is defined by the "`node timeout`" parameter in the `/etc/perceus/perceus.conf` configuration file. Nodes which have not checked in recently enough will be noted in the `status` command output; thus, to check for these "problem" nodes, use the following command:

```
# perceus node status | grep "timed out"
```

Note: *This is really only useful if the configured VNFS capsule of each node has* `provisiond` *(the Perceus client daemon) enabled and configured to check in at periodic intervals when nodes are in the* `ready` *state.*

node show: The `show` subcommand will dump most of the database values for each node. Because information is displayed in key/value pairs on separate lines, the output is typically filtered through an external command such as `grep`.

4.2.2 Setting Node Attributes

Just as you can view the database configuration, you can also easily modify the values by using the `perceus node set` command. The "set" command accepts the name of the database field to modify and the new value to assign. Legal field names include:

- **debug:** Set the debug flag to activate a higher level of verbosity and debug/trace information during execution of the node scripts. The bigger the number, the more output will be provided. Sufficiently high values will include shell-level command tracing (i.e., "set -x").
- **desc:** A free-form text string for posting notes and comments about the nodes.
- **enabled:** This must be defined and greater then zero to allow provisioning of this node.
- **group:** Perceus groups are literally a secondary way of identifying nodes logically. The group does not need to exist before a node is set to use it, nor must groups be deleted when no longer in use.
- **hostname:** This specifies not only the host's OS-level identity (i.e., the `hostname` value) as provisioned through Perceus' DHCP service but also the identity Perceus will employ (in general) to refer to each node. Because this value is user-definable, however, it is possible to assign the same `hostname` value to multiple nodes. If this occurs, the value no longer refers uniquely to a node. See below for more details on this.
- **vnfs:** Specify the VNFS that this node should use.

The usage of the set command is as follows:

```
# perceus node set desc "hello world" n0000 'n00[02-09]'
# perceus node set enabled 0 n0015
# perceus node set hostname newname n0000
```

4.2.3 Node Specification via MAC Address/NodeID

Caution is necessary when using the `perceus node set` command, as Perceus will not prevent you from performing potentially harmful operations like the following:

```
# perceus node set hostname duplicatename 'n00\*'
```

This will cause all of the nodes found by the pattern "n00*" to assume the same hostname ("duplicatename"). Though there may be cases where this would be desireable, it will also necessitate the use of the hardware address instead of the hostname as the primary node identifier (due to the latter no longer being unique) for Perceus commands. If this is the case, specify the `--nodeid` (or `-i` for short) option to use the MAC address as the command line node identifier instead of the hostname. For example:

```
# perceus --nodeid node show [MAC address]
# perceus -i node set hostname newname [MAC address]
```

4.2.4 Scaling with Node Groups

Separating nodes into Perceus "groups" allows for logical organization of nodes by criteria such as role, hardware configuration, physical location, etc. Once nodes have been grouped, Perceus allows commands to be performed on groups instead of node names, facilitating the treatment of node groups as logical entities. Using the `perceus group` command, entire groups of nodes can be configured en masse with descriptions, VNFS images, and more. Modules can also have group-specific features or be enabled/managed at the group level. (For example, the `passwdfile` module can create different passwd files for different groups.)

Here are some examples using the `group` command:

```
# perceus node set group interactive n000[0-9]
# perceus node set group compute n[0010-1023]
# perceus group set desc "Interactive and login nodes" interactive
# perceus group set vnfs caos-nsa-full-1.0-1.stateless.x86_64 interactive
# perceus group set vnfs caos-nsa-node-1.0-1.stateless.x86_64 compute
```

4.3 VNFS Management

Perceus supports any number of VNFS images and can assign any imported VNFS to any already known node (or set to new nodes via the `defaults.conf` configuration file.

Note: *All of the VNFS documentation appearing in this document assumes a Linux VNFS. For non-Linux VNFS support, you will need to consult Infiscale.com (info@infiscale.com).*

Another Note: *For the purposes of this document, we will assume that the VNFS you are using came from either a Perceus generation script or from Infiscale.com. Other capsules may behave differently, and this document may or may not apply to them. Whenever possible, distributed capsules are encouraged to follow our standards to keep a general level of consistency and sanity.*

4.3.1 VNFS Standards

Perceus does not make any hard-coded assumptions about the operating system contained in the VNFS. This means that Perceus is operating system agnostic for the slave nodes, but the VNFS has to be built to support the operating system to be provisioned directly. This is a feature of the VNFS and not Perceus directly.

Typically every VNFS capsule will contain a VNFS root filesystem in the form of a filesystem image (loopback), archive file (cpio or tar), or just a root filesystem in a directory (or some combination of all these). This

means that the capsule itself will be not only operating system specific but also stateless or stateful, as the node-scripts to support either model are significantly different. Also, the specific implementation of the `mount` and `umount` commands must be specific to the type of VNFS image used by Perceus.

VNFS capsules are distributed via a single file archive format that Perceus can handle and identified by the file suffix '.vnfs'. These files can be imported directly into Perceus and configured to nodes.

Perceus chooses to work with VNFS's in this manner so that standards can be met and results can be reproducible from site to site. More specifically this means supportability and compatibility assessments. VNFS capsules can be tested and certified for various hardware stacks. They can also be tuned for particular tasks including minimal node builds, network services, web services, database (MySQL and certified Oracle capsules), etc.

The naming convention of each VNFS capsule should provide all pertinent information about that capsule. The first field is the name of the operating system contained in the capsule itself. The next field is the operating system version. After these are the capsule revision number, state descriptor, and architecture of the capsule. The last field simply states that this file type is a VNFS. For example:

```
caos-nsa-node-1.0-1.stateless.x86_64.vnfs
caos-nsa-server-1.0-1.stateless.x86_64.vnfs
caos-nsa-full-1.0-1.stateful.x86_64.vnfs
rhel-node-4.5-1.stateless.x86_64.vnfs
rhel-lamp-5.0-1.stateless.x86_64.vnfs
gentoo-2007.0-1.statefull.x86_64.vnfs
```

Perceus has a `vnfs` command for importing, exporting, manipulating, and deleting VNFS images. This command is used in conjunction with VNFS-specific configuration files to manage VNFS images.

4.3.2 Importing a VNFS Capsule

Once a VNFS capsule has been obtained, it must be imported into Perceus before it may be used to provision nodes. Assuming the VNFS capsule was downloaded to the directory /root/, the command to do this is:

```
# perceus vnfs import /root/caos-nsa-node-1.0-1.stateless.x86_64.vnfs
```

This will not only import the Caos NSA Node VNFS capsule, but it will also run through a basic configuration routine probing for default values and prompting the user when necessary. The details of what happens during the configuration are specific for the VNFS and your server's configuration.

4.3.3 VNFS Image Alterations

After a particular capsule has been imported into Perceus, specific changes can be made to its contents, including adding or removing packages, editing files, changing configuration, etc., by mounting the VNFS image and directly modifying the node's root filesystem data.

To mount a VNFS image (e.g., caos-nsa-node-1.0-1.stateless.x86_64) use the following command:

```
# perceus vnfs mount caos-nsa-node-1.0-1.stateless.x86_64
```

The mount support itself is bundled from within the VNFS; as a result, some of the standards may change depending on where you obtained the VNFS capsule, who created it, and what operating system/filesystem type it is. The capsules will automatically make the filesystem accessible from the `/mnt` directory on the server.

- **Using the "chroot":** The VNFS at this point is essentially a directory of a real operating system, which means `chroot` is used to make changes. For example:

```
# chroot /mnt/caos-nsa-node-1.0-1.stateless.x86_64
# rpm -qa
# rpm -e grub
# exit
```

- **Installing packages into the VNFS using YUM and RPM:** Both YUM and RPM support chroot; thus, packages can be easily installed in VNFS images using these common tools and standards. Assuming the above mounted VNFS:

  ```
  # yum --installroot /mnt/caos-nsa-node-1.0-1.stateless.x86_64 install gcc
  # rpm --root /mnt/caos-nsa-node-1.0-1.stateless.x86_64 -e gcc
  ```

- **Installing packages from source into the VNFS:** Sometimes packages must be installed directly into the chroot from a package that isn't available on the master. Assuming a GNU style package called "package" to be installed in /opt:

  ```
  # tar xvzf package-1.0.tar.gz
  # cd package-1.0
  # ./configure --prefix=/opt/
  # make
  # make DESTDIR=/mnt/caos-nsa-node-1.0-1.stateless.x86_64 install
  ```

  Note: *Keep in mind that the* `configure` *script was not run on the node image, and any dynamic libraries or other dependencies are compiled in on the master server. These may not be present in the VNFS, and not using a package manager circumvents checks for things like this.*

After the required changes have been made to a VNFS image, it **must** be unmounted to activate the changes. The unmounting process takes a bit of time depending on the speed of the system and the size of the VNFS image. This is because the image is compressed after it is unmounted. Use the Perceus `vnfs umount` command to unmount a mounted VNFS image:

```
# perceus vnfs umount caos-nsa-node-1.0-1.stateless.x86_64
```

4.3.4 Hybridization

Hybridization is a feature for stateless VNFS capsules so that some of the VNFS content (files and/or directories) is excluded from the root filesystem itself and replaced with a symbolic link pointing to a network-based filesystem (usually at the configured `localstatedir`). As long as this path exists and is mounted, all of the symbolic links will resolve to their appropriate targets, thus providing a transparently hybridized filesystem. When the VNFS is imported into Perceus, Perceus will attempt to configure the proper mount point so that everything just works without intervention.

Files that are not excluded will be directly present and locally accessible in the node's memory. Files that get hit a lot or that are required for booting should always be local to the node (not hybridized).

By default NFS is used for hybridization, but by editing the `/etc/fstab` in the VNFS, any other filesystem (e.g., SMB/CIFS) may be specified as long support is present in the VNFS kernel.

4.3.5 The VNFS Configuration Files

There are some configuration options that are available for VNFS capsules after they have been imported. These will always be available from the `/etc/perceus/vnfs` symbolic link which points to the real VNFS directory in the configured `localstatedir`.

- **config:** The primary VNFS configuration file offers some very basic configuration options. The file is self documenting, and the only option that should ever require adjustment is `KERNEL_ARGS`.

  Any changes to this file do not require any service or command restarts; the new information will take effect the next time a node is provisioned.

- **hybridize:** Specifies which files/directories are to be hybridized. (See prior section for details.) Wildcards (e.g., '*') may be specified to match multiple filesystem entities. Note that hybridizing too aggressively may result in breakage.

  Warning: *You should never (ever!) include the* `localstatedir` *directory on the master node to be hybridized for any VNFS. In the example in this document, you should never hybridize* `/var/lib/perceus` *or any directory containing* `/var/lib/perceus`*. This is very important!*

  Troubleshooting: *If for any reason the localstatedir of the master node is not getting auto mounted, check the /etc/fstab inside the VNFS image.*

- **livesync.skip:** This is the skip list for the `vnfs livesync` command (which will be described shortly). Each line lists a file or directory name (or partial name) which should not be included in live VNFS synchronization.

  Sane defaults are provided, but due to the delicate nature of manipulating the root filesystem of a live server, any errors in the skip list can cause significant failures. For example, sync'ing `/etc/resolv.conf` after the DHCP client on the node has properly configured it could cause name resolution failures. Similarly, if the `passwdfile` module has updated `/etc/passwd`, a livesync could remove important cluster accounts from the system.

  Make sure to test livesync and this configuration file before running it on a production system!

4.3.6 Exporting a VNFS Capsule

After changes have been made to a VNFS image, it may be prudent to re-export it to capsule form (i.e., *.vnfs) for backup purposes or to share with others. Exported capsules can be (re-)imported into any Perceus installation using the `perceus vnfs import` command. To export an installed VNFS, use `vnfs export` as shown:

```
# perceus vnfs export caos-nsa-node-1.0-1.stateless.x86_64 /path/to/backup.vnfs
```

4.3.7 Distributing VNFS Changes to Nodes

There are multiple ways to update a node after a VNFS image change has been made. The first and most reliable method is to reboot the node. Unfortunately, production nodes are not always rebootable, and it may seem silly to reboot a node simply to update a single file or package.

Perceus supports a feature called "livesync" for situations when rebooting is not an option (or not the best option). It performs a live synchronization of the node's root filesystem image with a currently-mounted VNFS image. This functionality is accessible via the `perceus vnfs livesync` command:

```
# perceus vnfs livesync caos-nsa-node-1.0-1.stateless.x86_64
```

SSH and rsync are used to update the live root filesystem on the running node(s). Any files and directories which should *not* be synchronized must be listed in the `livesync.skip` file (see above) at the top level of the VNFS image.

An optional list of nodes may be added to the end of the `livesync` command; if missing, the default is to sync all nodes currently configured to use the specified VNFS.

Warning: *Again, if* `livesync.skip` *is not accurate, you can render your running node unusable. Always check this file before performing a livesync on a production system!*

4.4 Premade VNFS Capsules

Perceus was designed around the ideas not only of simplicity and ease of use but also supportability. Perceus VNFS capsules can be obtained from Infiscale.com or our Vendor partners tuned for particular tasks or certified/guaranteed to work on particular hardware stacks.

Contact Infiscale.com for more information or see the commercially available features and services section.

4.5 Creating VNFS Capsules

Perceus provides many of the tools necessary to create custom VNFS capsules. Because operating systems and distributions vary widely, no single process or tool will work flawlessly across all possible software platforms; moreover, not all operating systems have licenses which permit VNFS creation or distribution. However, tools for supported, freely available platforms come with Perceus, and these tools offer a starting point for the addition of new platforms as well.

4.5.1 Building the Capsule

Building a VNFS capsule from scratch is a two stage process using the tools in `/usr/share/perceus/vnfs-tools`. First, an image of the root filesystem (akin to the traditional "chroot jail" concept) must be created. This image will be used to make the capsule itself.

Select one of the supported distributions by running the `*-genchroot.sh` script bearing its name (and possibly version). When this script completes successfully, a complete root filesystem hierarchy will have been created in `/var/tmp/vnfs/`*VNFS name*`/`. (Note that these scripts do relatively little error checking for package installation, so watch the output to make sure no errors were encountered.)

Once the directory has been created, it must be converted to a stateless VNFS capsule using the `chroot2stateless.sh` script. The script takes as its arguments the path to the chroot directory and the path and filename for the capsule being created. Some final touchups will be performed on the filesystem image, and the capsule file will be created in the specified location.

For example, the following commands may be used to generate a VNFS capsule for Caos NSA Node version 0.9g on the x86_64 platform:

```
# cd /usr/share/perceus/vnfs-tools/
# ./caos-nsa-1.0-genchroot.sh
# ./chroot2stateless.sh /var/tmp/vnfs/caos-nsa-node-0.9g-1.x86_64 \
  /root/caos-nsa-node-0.9g-1.stateless.x86_64.vnfs
```

Note: *Chroot generation scripts are only distributed for freely-available platforms. For commercially available operating systems and capsules, please contact Infiscale.com.*

4.5.2 Enabling provisiond

The client-side Perceus daemon, `provisiond`, allows nodes to periodically check in with the master server after being provisioned. This provides a facility for ongoing configuration changes and deployments, but in order to function properly, the `perceus-provisiond` package must be installed in the VNFS image. To install this package, perform the following operations:

```
# perceus vnfs mount caos-nsa-node-1.0-1.stateless.x86_64
# rpm -ivh --root /mnt/caos-nsa-node-1.0-1.stateless.x86_64 \
  /usr/src/rpm/RPMS/x86_64/perceus-provisiond-@perceus_version@-@perceus_build@.x86_64.rpm
# perceus vnfs umount caos-nsa-node-1.0-1.stateless.x86_64
```

4.6 Perceus Module Management

Modules provide Perceus with a mechanism for extending and enhancing the capabilities of the provisioning process to include tertiary but important concepts like configuration distribution and service allocation. Modules are external collections of scripts and configuration data which interface with the Perceus daemon during node provisioning to allow additional operations to take place outside the normal Perceus flow. Since modules can be activated for any nodes or groups of nodes at any of the various provisioning states, potential uses for modules range from pre-boot needs (microcode updates, BIOS flashes) to runtime (account synchronization) and beyond.

4.6.1 Importing Modules

The Perceus distribution contains several useful modules already built in (though not set active by default), but numerous additional modules are available as well. Modules distributed separately come in the form of *.pmod files and can be imported into Perceus with the `perceus module import` command:

```
# perceus module import /path/to/module.pmod
```

4.6.2 Activation

When a node boots and connects to the Perceus service, it will begin the communication by stating what provisionary state it is in (e.g. init, boot, ready, etc...) and its NodeID. With this information, the Perceus daemon can figure out what node/boot scripts need to be run for this node and relay them to each node for execution.

Modules can be activated for various provisionary states. Only two states are used by default ("init" and "ready"), but other states are easily added. Each provisionary state has four "sub-states" for the different delineations Perceus supports (all, group/*, vnfs/*, node/*).

The following examples illustrate the various ways of activating modules for different states:

```
# perceus module activate hostname init/all
# perceus module activate groupfile init/node/node0000 init/group/anothergroup
# perceus module activate passwdfile ready/group/newgroup
# perceus module activate virtualization
```

If no state is specified with the `module activate` command, the default for that module is used instead.

4.6.3 Included Modules

The following modules are freely distributable and are supplied in the Perceus distribution. (Contact Infiscale.com if additional modules are required.)

- **masterauth**: This module will synchronize the master's `/etc/passwd` and `/etc/group` files to the nodes. The shadow file is **not** transferred, so all account authorization should be configured using secure SSH keys and shared home directories from the master. (SSH key configuration is demonstrated in the **Command Line Examples** section of this document). This module's default state is both init/all and ready/all.
- **passwdfile**: The passwdfile module facilitates finer-grain control over `/etc/passwd` contents on each node based on groups, VNFS images, etc. Its configuration directory (`/etc/perceus/modules/passwdfile/`) contains a hierarchy through which the `passwd` file may be assembled from node, groups, and vnfs components as well as the global "all" file. The "all" file gets included first and is usually the source of entries such as "root;" other files are appended after. Like the masterauth module above, this requires the use of non-password-based ssh authentication.

- **groupfile**: Similar to the passwdfile module, but manages the `/etc/group` file.
- **ipaddr**: While initial provisioning may configure the node with a dynamic network address (DHCP), the default configuration as of Perceus 1.4 is to use static IP assignments instead of DHCP after the nodes have been provisioned. (Of course, within the VNFS capsule, you can configure the network stack however you wish).

  The 'ipaddr' Perceus module should be used to statically configure the IPv4 network stack on the booted node. Once this module has been activated, you can edit it's configuration in `/etc/perceus/modules/ipaddr`. The format is as follows:

  ```
  [nodename/pattern] device1:ipaddr/netmask/gw device2:ipaddr/netmask/gw ...
  ```

  The default configuration is:

  ```
  * eth0:[default]/[default] eth1:[default]/[default]
  ```

  This matches all nodes via the wildcard `*` and configigures two network devices (eth0 and eth1). The value `[default]` specifies that Perceus should retrieve the node's IP address from the master's `/etc/hosts` and the netmask from the master's own interface.

  note: *This Perceus module is most effective in the "init" provisionary state; it will append a network configuration file suitable for Caos, Red Hat and SuSE.*
- **xcpu**: XCPU is the successor to bproc developed at Los Alamos National Labratory. XCPU provides a framework to run binary applications on nearly-empty nodes in a very scalable manner. This means one can spawn binaries from the master onto the nodes without even provisioning a VNFS; all dependencies are deployed automatically with no additional configuration required. Perceus supports running XCPU directly via the Perceus stage 1 bootstrap.
- **syncnodes**: This module can be dangerous if not used carefully. It will cause a node to automatically be re-provisioned if a major configuration change has occured when the node checks in at the `ready` provisionary state. It does some basic tests to make sure that the node is not busy (i.e., no user processes running) before the reboot is performed.
- **hostname**: Set the hostname of a node. This maybe required when statically addressing the node's IP address when using the `ipaddr` Perceus module.
- **hostfile**: This module will add entries in /etc/hosts automatically as nodes boot. It has some destructive capabilities and should be used with caution.
- **modprobe**: For hardware platforms which require early availability of additional kernel modules, this module provides a mechanism similar to traditional initrd/initramfs bootstrapping methods. Perceus will load requested modules prior to booting the VNFS.

Most HPC-specific installations will require both the `ipaddr` and `hostname` Perceus modules to be activated.

Chapter 5

Command Line Examples

5.1 Quick Install

The following examples illustrate step by step the process of getting a Perceus master up and running. With few exceptions, most of the examples can be copied and pasted into a terminal window on the host in question. One important assumption being made is that `eth1` is the cluster (i.e., private) network interface. A Red Hat operating system (or a rebuild) is also assumed.

5.1.1 Stage One: Host OS Configuration

```
# Install the proper packages after installation
yum -y groupinstall "Development Tools"
yum -y install nasm

# Configure the network device 'eth1'
echo "DEVICE=eth1" > /etc/sysconfig/network-scripts/ifcfg-eth1
echo "BOOTPROTO=static" >> /etc/sysconfig/network-scripts/ifcfg-eth1
echo "ONBOOT=yes" >> /etc/sysconfig/network-scripts/ifcfg-eth1
echo "IPADDR=10.0.0.1" >> /etc/sysconfig/network-scripts/ifcfg-eth1
echo "NETMASK=255.255.0.0" >> /etc/sysconfig/network-scripts/ifcfg-eth1

# Bring up the newly configured network device
ifup eth1

# Just in case your running RH's iptables rule, make sure 'eth1' is wide open
# note: If you are running a custom ruleset, make sure you do this it
# yourself
if [ -f "/etc/sysconfig/iptables" ]; then
   iptables -A RH-Firewall-1-INPUT -i eth1 -j ACCEPT
   /etc/init.d/iptables save
fi

vi /etc/exports
  # Add the following entry:
  # /usr/local    10.0.0.0/255.255.0.0(no_root_squash,async,ro)

exportfs -av
chkconfig nfs on
```

```
# disable selinux (this should reboot if this was enabled)
if ! grep -q "^SELINUX=disabled" /etc/sysconfig/selinux; then
   sed -i -e 's/^SELINUX=.*/SELINUX=disabled/' /etc/sysconfig/selinux
   echo "Will reboot in 10 seconds to disable selinux"
   sleep 10
   /sbin/reboot
fi
```

5.1.2 Stage Two: Perceus Dependencies

```
mkdir /tmp/perceus_deps
cd /tmp/perceus_deps
# Define the download location
MIRROR=http://www.perceus.org/downloads/perceus/v1.x/dependencies/
PACKAGES="bash-completion-20060301-1.caos.src.rpm
          perl-IO-Interface-0.98-1.caos.src.rpm
          perl-Net-ARP-1.0-2.caos.src.rpm
          perl-Unix-Syslog-0.99-1.caos.src.rpm"

echo "%debug_package %{nil}" >> ~/.rpmmacros

for i in $PACKAGES; do
   rpmbuild --rebuild $i || exit 1
done

cd /usr/src/*/RPMS/

for i in $PACKAGES; do
   rpm -Uvh */`echo $i | sed -e 's/[0-9].*//'*.rpm` || exit 1
done
rm -rf /tmp/perceus_deps
```

5.1.3 Stage Three: Perceus

```
# Set the Perceus version
PERCEUS_VERSION=@perceus_version@

# Download the Perceus source code
wget http://www.perceus.org/portal/files/perceus-$PERCEUS_VERSION.tar.gz

# Some versions of tar need this option for RPM to build properly from
# tarball
export TAR_OPTIONS=--wildcards

# Build Perceus RPMS
rpmbuild -ta perceus-$PERCEUS_VERSION.tar.gz

# Install the newly built RPMS
rpm -Uvh /usr/src/redhat/RPMS/*/perceus-$PERCEUS_VERSION-*.rpm

# Configure perceus to use eth1 for the master NFS server
sed -i -e 's/^vnfs transfer master.*$/vnfs transfer master = 10.0.0.1/' \
  /etc/perceus/perceus.conf
```

```
# Run Perceus for the first time
perceus
  # The first time Perceus is run, it will do a first time interactive
  # initialization. Perceus should have configured itself to run. Any
  # errors here, should be reported.

# Restart the perceus daemons
/etc/init.d/perceus restart
```

5.2 Node Commands

5.2.1 Set the VNFS for nodes n0000-n0006, n0020-n0040 and all of the ”io” nodes:

```
# perceus node set vnfs [VNFS NAME] n00[00-05,06,20-40] io*
```

5.2.2 Delete nodes n0000, n0001, and n0002:

```
# perceus node delete n000[0-2]
```

5.2.3 Add a new node and then replace an old node with what was just added:

```
# perceus node add 00:11:22:33:44:55
# perceus node set desc "New node is a hot spare" n1024
# perceus node set desc "This node just burnt up its CPU" n0099
# perceus node replace n0099 n1024
```

5.3 VNFS Commands

5.3.1 Importing a new VNFS capsule

```
# perceus vnfs import /path/to/capsule.vnfs
```

5.4 Module Commands

5.4.1 Importing a new Perceus module

```
# perceus module import /path/to/module.pmod
```

5.4.2 Activating a Perceus module

```
# perceus module activate module (optional provisionary state, e.g. init/all)
```

5.5 Contact Support

Using Perceus itself to contact our support team will prompt for some basic information, and then send the request, ticket number, and a brief synopsis of the Perceus configuration to the Perceus/Infiscale support staff.

```
# perceus contact support
```

5.6 Non-Perceus-Specific Commands

5.6.1 Generating Passphraseless SSH Keys

To spawn commands directly on the nodes, users typically need shell access to each node. Historically the `rsh` command has been used for this purpose, but due to scaling limitations, lack of proper authentication, and the absence of appropriate privacy and security, `ssh` has become the new standard. While the Perceus user authentication modules above will manage the local system accounts, each individual user (or a kind administrator) will need to configure his or her own SSH environment to support passwordless access. The following method will create SSH keys with an empty passphrase and allow non-interactive spawning of commands to the nodes in the cluster:

```
$ mkdir ~/.ssh
$ ssh-keygen -f ~/.ssh/id_dsa -t dsa -P ""
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
$ chmod 400 ~/.ssh/authorized_keys
```

Note: This method assumes a shared home directory on all nodes of the cluster.

Chapter 6

Commercial Features and Services

Perceus is licensed free of charge via the GNU GPL. This means that all costs incurred by its development, testing, and distribution have been borne by Infiscale.com. We encourage both corporate partners and licensees to contact us as it helps us to maintain our freely available product.

If you wish to distribute Perceus on a commercial basis (including utilizing it with commercial software or with a hardware purchase) you should pursue a commercial relationship with Infiscale.com.

6.1 Web Interface

Both users and reseller partners can license our web GUI interface for node management. It works very well for small and large installations and can be easily wrapped with a default theme for partners to provide web-based "value add" to their customers.

6.2 VNFS Capsules

Infiscale.com provides ready-made VNFS capsules tuned for specific needs and tasks and offers complete environments utilizing different VNFS capsules to fit the requirements of more complex installations.

High Performance Computing Cluster (HPCC) environment: VNFS capsules are available premade and tested for your required applications and environment. These can serve either as a baseline starting point for a working cluster or as a turn-key solution capable of provisioning your entire HPCC infrastructure (compute nodes, interactive nodes, scheduling nodes, etc...).

LAMP: Linux, Apache, MySQL and Perl/PHP is a standard in the web computing industry. Our LAMP environment includes all components necessary to build a scalable infrastructure.

Clustered virtualization: Using a clustered virtualization platform, VM's are easily managed and scheduled at scale. The VM-capable VNFS capsules allow for even stateless virtualization and support KVM, Xen, and VMWare.

Web and ISP environments: Large-scale dedicated ISP services can be easily deployed using Perceus. Perceus helps facilitate using inexpensive and somewhat disposable platforms for large scale deployment. Hardware failures are not a problem as the same operating system can be re-provisioned within moments of a failure.

6.3 Embedded Perceus

The architecture of Perceus facilitates the stage one operating system being embedded on the motherboard or in a flash space. This has already been implemented and demonstrated as a working payload for Intel's Rapid Boot; other motherboards have been prototyped as well.

6.4 Hardware Certification

A result of Perceus leveraging a multi-stage boot system means that Perceus must be compatible with the hardware in question. We have already seen cases where motherboard vendors diverged from the chip manufacturer's reference implementation, and this deviation didn't behave properly with Perceus.

We offer a certification for various hardware stacks to assure customers of guaranteed compatibility.

6.5 Support

If you need further assistance past this document, contact Infiscale.com to setup a support relationship either directly through us or via a partner.

6.6 Consulting

Customizations, embedding, feature requests, VNFS capsules, and Perceus modules can all be built on an as-needed basis.

6.7 Licenses

If you would like a non-GPL license of Perceus, we have a secondary license available. This license will allow distribution and linking to Perceus directly by non-GPL compatible applications (including wrapping the command line).