






Release Notes

SUSE Linux Enterprise High Performance Computing 15 GA

This document provides guidance and an overview to high-level general features and updates for the SUSE Linux Enterprise High Performance Computing 15 GA. It describes the capabilities and limitations of the SUSE Linux Enterprise High Performance Computing 15 GA.

These release notes are updated periodically. The latest version is always available at <https://www.suse.com/releasenotes> . General documentation can be found at: <https://www.suse.com/documentation/sles-15> .

General documentation for SLES and SUSE Linux Enterprise High Performance Computing can be found at: <http://www.suse.com/documentation/> .

Publication Date: 2018-07-13, Version: 15.20180713

Contents

- 1 SUSE Linux Enterprise High Performance Computing 15 GA 3
- 2 Hardware Platform Support 3
- 3 Support and Life Cycle 4
- 4 Documentation and Other Information 4
- 5 How to Obtain Source Code 4
- 6 Support Statement for SUSE Linux Enterprise High Performance Computing 5
- 7 Installation and Upgrade 6
- 8 Functionality 8

9	HPC Libraries	20
10	Legal Notices	28

1 SUSE Linux Enterprise High Performance Computing 15 GA

SUSE Linux Enterprise High Performance Computing is a highly scalable, high performance open-source operating system designed to utilize the power of parallel computing for modeling, simulation and advanced analytics workloads.

SUSE Linux Enterprise High Performance Computing 15 GA provides tools and libraries related to High Performance Computing. Presently, the tools include:

- Workload manager
- Remote and parallel shells
- Performance monitoring and measuring tools
- Serial console monitoring tool
- Cluster power management tool
- A tool for discovering the machine hardware topology
- System monitoring
- A tool for monitoring memory errors
- A tool for determining the CPU model and its capabilities (x86-64 only)
- User-extensible heap manager capable of distinguishing between different kinds of memory (x86-64 only)
- Serial and parallel computational libraries providing the common standards BLAS, LAPACK, ...
- Various MPI implementations
- Serial and parallel libraries for the HDF5 file format

2 Hardware Platform Support

SUSE Linux Enterprise High Performance Computing 15 GA is available for the Intel 64/AMD64 (x86-64) and AArch64 platforms.

3 Support and Life Cycle

SUSE Linux Enterprise High Performance Computing 15 GA is supported throughout the life cycle of SLE 15 GA. ESPOS (Extended Service Overlap Support) as well as Long Term Support Service are also available for this product. Any release package is fully maintained and supported until the availability of the next release.

For more information, see the Support Policy page <https://www.suse.com/support/policy.html>.

4 Documentation and Other Information

Accessing the documentation on the product media:

- Read the READMEs on the media.
- Get the detailed change log information about a particular package from the RPM (where *FILENAME.rpm* is the name of the RPM):

```
rpm --changelog -qp FILENAME.rpm
```

- Check the ChangeLog file in the top level of the media for a chronological log of all changes made to the updated packages.
- These Release Notes are identical across all architectures, and the most recent version is always available online at <https://www.suse.com/releasesnotes/>. Some entries may be listed twice, if they are important and belong to more than one section.

5 How to Obtain Source Code

This SUSE product includes materials licensed to SUSE under the GNU General Public License (GPL). The GPL requires SUSE to provide the source code that corresponds to the GPL-licensed material. The source code is available for download at <https://www.suse.com/download-linux/source-code.html>.

Also, for up to three years after distribution of the SUSE product, upon request, SUSE will mail a copy of the source code. Requests should be sent by e-mail to mailto:sle_source_request@suse.com or as otherwise instructed at <https://www.suse.com/download-linux/source-code.html>. SUSE may charge a reasonable fee to recover distribution costs.

6 Support Statement for SUSE Linux Enterprise High Performance Computing

To receive support, you need an appropriate subscription with SUSE. For more information, see <https://www.suse.com/products/server/services-and-support/> .

The following definitions apply:

L1

Problem determination, which means technical support designed to provide compatibility information, usage support, ongoing maintenance, information gathering and basic troubleshooting using available documentation.

L2

Problem isolation, which means technical support designed to analyze data, reproduce customer problems, isolate problem area and provide a resolution for problems not resolved by Level 1 or alternatively prepare for Level 3.

L3

Problem resolution, which means technical support designed to resolve problems by engaging engineering to resolve product defects which have been identified by Level 2 Support.

For contracted customers and partners, the SUSE Linux Enterprise High Performance Computing 15 GA is delivered with L3 support for all packages, except the following:

- Technology Previews
- sound, graphics, fonts and artwork
- packages that require an additional customer contract
- development packages for libraries which are only delivered with L2 support

SUSE will only support the usage of original (that is, unchanged and un-recompiled) packages.

7 Installation and Upgrade

SUSE Linux Enterprise High Performance Computing comes with a number of preconfigured system roles for HPC. These roles provide a set of preselected packages typical for the specific role, as well as an installation workflow that will configure the system to make the best use of system resource based on a typical role use case.

7.1 System Roles for SUSE Linux Enterprise High Performance Computing 15 GA

With SUSE Linux Enterprise High Performance Computing 15 GA, it is possible to choose specific roles for the system based on modules selected during the installation process. When the HPC Module is enabled, these three roles are available:

HPC Management Server (Head Node)

This role includes the following features:

- Uses XFS as the default root file system
- Includes HPC-enabled libraries
- Disables firewall and Kdump services
- Installs controller for the Slurm Workload Manager
- Mounts a large scratch partition to /var/tmp

HPC Compute Node

This role includes the following features:

- Uses XFS as the default root file system
- Includes HPC-enabled libraries
- Disables firewall and Kdump services
- Based from minimal setup configuration
- Installs client for the Slurm Workload Manager
- Does not create a separate home partition
- Mounts a large scratch partition to /var/tmp

HPC Development Node

This role includes the following features:

- Includes HPC-enabled libraries
- Adds compilers and development toolchain

The scratch partition `/var/tmp/` will only be created if there is sufficient space available on the installation medium (minimum 32 GB).

The Environment Module `Lmod` will be installed for all roles. It is required at build time and run time of the system. For more information, see [Section 8.7, “Lmod # Lua-based Environment Modules”](#).

All libraries specifically build for HPC will be installed under `/usr/lib/hpc`. They are not part of the standard search path, thus the `Lmod` environment module system is required.

`Munge` authentication is installed for all roles. This requires to copy the same generated munge keys to all nodes of a cluster. For more information, see [Section 8.15, “mrsh/mrlogin # Remote Login Using munge Authentication”](#) and [Section 8.14, “munge Authentication”](#).

From the Ganglia monitoring system, the data collector `ganglia-gmod` is installed for every role, while the data aggregator `ganglia-gmetad` needs to be installed manually on the system which is expected to collect the data. For more information, see [Section 8.3, “Ganglia # System Monitoring”](#).

The system roles are only available for new installations of SUSE Linux Enterprise High Performance Computing.

7.2 Installation

This section includes information related to the initial installation of the SUSE Linux Enterprise High Performance Computing 15 GA.

7.3 Upgrade-Related Notes

This section includes upgrade-related information for the SUSE Linux Enterprise High Performance Computing 15 GA.

You can upgrade to SUSE Linux Enterprise High Performance Computing 15 GA from SLES 12 SP3 or SLE HPC 12 SP3. When upgrading from SLES 12 SP3, the upgrade will only be performed if the SLE HPC module has been registered prior to upgrading. Otherwise, the system will instead be upgraded to SLES 15.

To upgrade from SLES 12 to SLES 15, make sure to unregister the SLE HPC module prior to upgrading. To do so, open a root shell and execute:

```
SUSEConnect -d -p sle-module-hpc/12/ARCH
```

Replace ARCH with the architecture used (x86_64, aarch64).

When migrating to SUSE Linux Enterprise High Performance Computing 15 GA, all modules not supported by the migration target need to be deregistered. This can be done by executing:

```
SUSEConnect -d -p sle-module-MODULE_NAME/12/ARCH
```

Replace MODULE_NAME by the name of the module and ARCH with the architecture used (x86_64, aarch64).

8 Functionality

This section comprises information about packages and their functionality, as well as additions, updates, removals and changes to the package layout of software.

8.1 `cpuid` – x86 CPU Identification Tool

`cpuid` executes the x86 CPUID instruction and decodes and prints the results to stdout. Its knowledge of Intel, AMD and Cyrix CPUs is fairly complete. It also supports Intel Knights Mill CPUs (x86-64).

To install `cpuid`, run: `zypper in cpuid`.

For information about runtime options for `cpuid`, see the man page `cpuid(1)`.

Note that this tool is only available for x86-64.

8.2 ConMan – The Console Manager

ConMan is a serial console management program designed to support a large number of console devices and simultaneous users. It supports:

- local serial devices
- remote terminal servers (via the telnet protocol)
- IPMI Serial-Over-LAN (via FreeIPMI)

- Unix domain sockets
- external processes (for example, using 'expect' scripts for telnet, ssh, or ipmi-sol connections)

ConMan can be used for monitoring, logging and optionally timestamping console device output. To install ConMan, run `zypper in conman`.

! Important: conmand Sends Unencrypted Data

The daemon `conmand` sends unencrypted data over the network and its connections are not authenticated. Therefore, it should be used locally only: Listening to the port `local-host`. However, the IPMI console does offer encryption. This makes `conman` a good tool for monitoring a large number of such consoles.

Usage:

- ConMan comes with a number of expect-scripts: check `/usr/lib/conman/exec`.
- Input to `conman` is not echoed in interactive mode. This can be changed by entering the escape sequence `&E`.
- When pressing `Enter` in interactive mode, no line feed is generated. To generate a line feed, press `Ctrl-L`.

For more information about options, see the man page of ConMan.

8.3 Ganglia – System Monitoring

Ganglia is a scalable distributed monitoring system for high-performance computing systems, such as clusters and grids. It is based on a hierarchical design targeted at federations of clusters.

To use Ganglia, make sure to install `ganglia-gmetad` on the management server then start the Ganglia meta-daemon: `rcgmetad start`. To make sure the service is started after a reboot, run: `systemctl enable gmetad`. On each cluster node which you want to monitor, install `ganglia-gmond`, start the service `rcgmond start` and make sure it is enabled to be started automatically after a reboot: `systemctl enable gmond`. To test whether the `gmond` daemon has connected to the meta-daemon, run `gstat -a` and check that each node to be monitored is present in the output.

When using the Btrfs file system, the monitoring data will be lost after a rollback and the service `gmetad`. To be able to start it again, either install the package `ganglia-gmetad-skip-bcheck` or create the file `/etc/ganglia/no_btrfs_check`.

To use the Ganglia Web interface, it is required to add the "Web and Scripting Module" first. This can be done by running `SUSEConnect -p sle-module-web-scripting/15/x86_64`. Install `ganglia-web` on the management server. Depending on which PHP version is used (default is PHP 5), enable it in Apache2: `a2enmod php5` or `a2enmod php7`. Then start Apache2 on this machine: `rcapache2 start` and make sure it is started automatically after a reboot: `systemctl enable apache2`. The Ganglia Web interface should be accessible from `http://MANAGEMENT_SERVER/ganglia`.

8.4 Genders – Static Cluster Configuration Database

Support for Genders has been added to the HPC module.

Genders is a static cluster configuration database used for configuration management. It allows grouping and addressing sets of hosts by attributes and is used by a variety of tools. The Genders database is a text file which is usually replicated on each node in a cluster.

Perl, Python, C, and C++ bindings are supplied with Genders, the respective packages provide man pages or other documentation describing the APIs.

To create the Genders database, follow the instructions and examples in `/etc/genders` and check `/usr/share/doc/packages/genders-base/TUTORIAL`. Testing a configuration can be done with `nodeattr` (for more information, see `man 1 nodeattr`).

List of packages:

- `genders`
- `genders-base`
- `genders-devel`
- `python-genders`
- `genders-perl-compat`
- `libgenders0`
- `libgendersplusplus2`

8.5 GNU Compiler Collection for HPC

`gnu-compilers-hpc` installs the base version of the GNU compiler suite and provides environment files for Lmod to select this compiler suite and provides environment module files for them. This version of the compiler suite is required to enable linking against HPC libraries enabled for environment modules.

This package requires `lua-lmod` to supply environment module support.

To install `gnu-compilers-hpc`, run:

```
zypper in gnu-compilers-hpc
```

To set up the environment appropriately and select the GNU toolchain, run:

```
module load gnu
```

If you have more than one version of this compiler suite installed, add the version number of the compiler suite. For more information, see [Section 8.7, “Lmod # Lua-based Environment Modules”](#).

8.6 hwloc – Portable Abstraction of Hierarchical Architectures for High-Performance Computing

`hwloc` provides command-line tools and a C API to obtain the hierarchical map of key computing elements, such as: NUMA memory nodes, shared caches, processor packages, processor cores, processing units (logical processors or "threads") and even I/O devices. `hwloc` also gathers various attributes such as cache and memory information, and is portable across a variety of different operating systems and platforms. Additionally it may assemble the topologies of multiple machines into a single one so as to let applications consult the topology of an entire fabric or cluster at once.

In graphical mode (X11), `hwloc` can display the topology in a human-readable format. Alternatively, it can export to one of several formats, including plain text, PDF, PNG, and FIG. For more information, see the man pages provided by `hwloc`.

It also features full support for import and export of XML-formatted topology files via the `libxml2` library.

The package `hwloc-devel` offers a library that can be directly included into external programs. This requires that the `libxml2` development library (package `libxml2-devel`) is available when compiling `hwloc`.

8.7 Lmod – Lua-based Environment Modules

Lmod is an advanced environment module system which allows the installation of multiple versions of a program or shared library, and helps configure the system environment for the use of a specific version. It supports hierarchical library dependencies and makes sure that the correct version of dependent libraries are selected. Environment Modules-enabled library packages supplied with the HPC module support parallel installation of different versions and flavors of the same library or binary and are supplied with appropriate `lmod` module files.

Installation and Basic Usage

To install Lmod, run: **`zypper in lua-lmod`**.

Before Lmod can be used, an init file needs to be sourced from the initialization file of your interactive shell. The following init files are available:

```
/usr/share/lmod/<lmod_version>/init/bash
/usr/share/lmod/<lmod_version>/init/ksh
/usr/share/lmod/<lmod_version>/init/tcsh
/usr/share/lmod/<lmod_version>/init/zsh
/usr/share/lmod/<lmod_version>/init/sh
```

Pick the one appropriate for your shell. Then add the following to the init file of your shell:

```
. /usr/share/lmod/<LMOD_VERSION>/init/<INIT-FILE>
```

To obtain `<lmod_version>`, run:

```
rpm -q lua-lmod | sed "s/.*-([^-]\+\)-.*\/\1/"
```

The init script adds the command **`module`**.

Listing Available Modules

To list the available all available modules, run: **`module spider`**. To show all modules which can be loaded with the currently loaded modules, run: **`module avail`**. A module name consists of a name and a version string separated by a `/` character. If more than one version is available for a certain module name, the default version (marked by `*`) or (if this is not set) the one with the highest version number is loaded. To refer to a specific module version, the full string `NAME/VERSION` may be used.

Listing Loaded Modules

`module list` shows all currently loaded modules. Refer to `module help` for a short help on the module command and `module help MODULE-NAME` for a help on the particular module. Note that the `module` command is available only when you log in after installing `lua-lmod`.

Gathering Information About a Module

To get information about a particular module, run: `module whatis MODULE-NAME` To load a module, run: `module load MODULE-NAME`. This will ensure that your environment is modified (that is, the `PATH` and `LD_LIBRARY_PATH` and other environment variables are prepended) such that binaries and libraries provided by the respective modules are found. To run a program compiled against this library, the appropriate `module load` commands must be issued beforehand.


Loading Modules

The `module load MODULE` command needs to be run in the shell from which the module is to be used. Some modules require a compiler toolchain or MPI flavor module to be loaded before they are available for loading.

Environment Variables

If the respective development packages are installed, build time environment variables like `LIBRARY_PATH`, `C_PATH`, `C_INCLUDE_PATH` and `CPLUS_INCLUDE_PATH` will be set up to include the directories containing the appropriate header and library files. However, some compiler and linker commands may not honor these. In this case, use the appropriate options together with the environment variables `-I PACKAGE_NAME_INC` and `-L PACKAGE_NAME_LIB` to add the include and library paths to the command lines of the compiler and linker.

For More Information

For more information on Lmod, see <https://lmod.readthedocs.org> .

8.8 ohpc – OpenHPC Compatibility Macros

`ohpc` contains compatibility macros to build OpenHPC packages on SUSE Linux Enterprise.

To install `ohpc`, run: `zypper in ohpc`.

8.9 `pdsh` – Parallel Remote Shell Program

`pdsh` is a parallel remote shell which can be used with multiple back-ends for remote connections. It can run a command on multiple machines in parallel.

To install `pdsh`, run `zypper in pdsh`.

On SLES 12, the back-ends `ssh`, `mrsh`, and `exec` are supported. The `ssh` back-end is the default. Non-default login methods can be used by either setting the `PDSH_RCMD_TYPE` environment variable or by using the `-R` command argument.

When using the `ssh` back-end, it is important that a non-interactive (that is, passwordless) login method is used.

The `mrsh` back-end requires the `mrshd` to be running on the client. The `mrsh` back-end does not require the use of reserved sockets. Therefore, it does not suffer from port exhaustion when executing commands on many machines in parallel. For information about setting up the system to use this back-end, see [Section 8.15, “mrsh/mrlogin # Remote Login Using munge Authentication”](#).

Remote machines can either be specified on the command line or `pdsh` can use a `machines` file (`/etc/pdsh/machines`), `dsh` (Dancer's shell) style groups or netgroups. Also, it can target nodes based on the currently running Slurm jobs.

The different ways to select target hosts are realized by modules. Some of these modules provide identical options to `pdsh`. The module loaded first will win and consume the option. Therefore, we recommend limiting yourself to a single method and specifying this with the `-M` option.

The `machines` file lists all target hosts one per line. The appropriate netgroup can be selected with the `-g` command line option.

The following host-list plugins for `pdsh` are supported: `machines`, `slurm`, `netgroup` and `dsh-group`. Each host-list plugin is provided in a separate package. This avoids conflicts between command line options for different plugins which happen to be identical and helps to keep installations small and free of unneeded dependencies. Package dependencies have been set to prevent installing plugins with conflicting command options. To install one of the plugins, run:

```
zypper in pdsh-PLUGIN_NAME
```

For more information, see the man page `pdsh`.

8.10 PowerMan – Centralized Power Control for Clusters

PowerMan allows manipulating remote power control devices (RPC) from a central location. It can control:

- local devices connected to a serial port
- RPCs listening on a TCP socket
- RPCs which are accessed through an external program

The communication to RPCs is controlled by “expect”-like scripts. For a list of currently supported devices, see the configuration file /etc/powerman/powerman.conf.

To install PowerMan, run **zypper in powerman**.

To configure it, include the appropriate device file for your RPC (/etc/powerman/*.dev) in /etc/powerman/powerman.conf and add devices and nodes. The device “type” needs to match the “specification” name in one of the included device files, the list of “plugs” used for nodes need to match an entry in the “plug name” list.

After configuring PowerMan, start its service by:

```
systemctl start powerman.service
```

To start PowerMan automatically after every boot, do:

```
systemctl enable powerman.service
```

Optionally, PowerMan can connect to a remote PowerMan instance. To enable this, add the option listen to /etc/powerman/powerman.conf.



Important: Unencrypted Transfer

Data is transferred unencrypted, therefore this is not recommended unless the network is appropriately secured.

8.11 rasdaemon – Utility to Log RAS Error Tracings

rasdaemon is a RAS (Reliability, Availability and Serviceability) logging tool. It records memory errors using the EDAC tracing events. EDAC drivers in the Linux kernel handle detection of ECC errors from memory controllers.

`rasdaemon` can be used on large memory systems to track, record and localize memory errors and how they evolve over time to detect hardware degradation. Furthermore, it can be used to localize a faulty DIMM on the board.

To check whether the EDAC drivers are loaded, execute:

```
ras-mc-ctl --status
```

The command should return `ras-mc-ctl: drivers are loaded`. If it indicates that the drivers are not loaded, EDAC may not be supported on your board.

To start `rasdaemon`, run **`systemctl start rasdaemon.service`**. To start `rasdaemon` automatically at boot time, execute **`systemctl enable rasdaemon.service`**. The daemon will log information to `/var/log/messages` and to an internal database. A summary of the stored errors can be obtained with:

```
ras-mc-ctl --summary
```

The errors stored in the database can be viewed with

```
ras-mc-ctl --errors
```

Optionally, you can load the DIMM labels silk-screened on the system board to more easily identify the faulty DIMM. To do so, before starting `rasdaemon`, run:

```
systemctl start ras-mc-ctl start
```

For this to work, you need to set up a layout description for the board. There are no descriptions supplied by default. To add a layout description, create a file with an arbitrary name in the directory `/etc/ras/dimm_labels.d/`. The format is:

```
Vendor: VENDOR-NAME
Model: MODEL-NAME
LABEL: MC.TOP.MID.LOW
```

8.12 Slurm – Utility for HPC Workload Management

Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for Linux clusters containing up to 65,536 nodes. Components include machine status, partition management, job management, scheduling and accounting modules.

For a minimal setup to run Slurm with *munge* support on one compute node and multiple control nodes, follow these instructions:

1. Install `slurm-munge` on the control and compute nodes: **`zypper in slurm-munge`**
2. Configure, enable and start "munge" on the control and compute nodes as described in [Section 8.15, "mrsh/mrlogin # Remote Login Using munge Authentication"](#).
3. On the compute node, edit `/etc/slurm/slurm.conf`:

- a. Configure the parameter `ControlMachine=CONTROL_MACHINE` with the host name of the control node.

To find out the correct host name, run `hostname -s` on the control node.

- b. Additionally add:

```
NodeName=NODE_LIST Sockets=SOCKETS \  
CoresPerSocket=CORES_PER_SOCKET \  
ThreadsPerCore=THREADS_PER_CORE \  
State=UNKNOWN
```

and

```
PartitionName=normal Nodes=NODE_LIST \  
Default=YES MaxTime=24:00:00 State=UP
```

where `NODE_LIST` is the list of compute nodes (that is, the output of `hostname -s` run on each compute node (either comma-separated or as ranges: `foo[1-100]`). Additionally, `SOCKETS` denotes the number of sockets, `CORES_PER_SOCKET` the number of cores per socket, `THREADS_PER_CORE` the number of threads for CPUs which can execute more than one thread at a time. (Make sure that `SOCKETS * CORES_PER_SOCKET * THREADS_PER_CORE` does not exceed the number of system cores on the compute node).

- c. On the control node, copy `/etc/slurm/slurm.conf` to all compute nodes:

```
scp /etc/slurm/slurm.conf COMPUTE_NODE:/etc/slurm/
```

- d. On the control node, start `slurmctld`:

```
systemctl start slurmctld.service
```

Also enable it so that it starts on every boot:

```
systemctl enable slurmd.service
```

- e. On the compute nodes, start and enable `slurmd`:

```
systemctl start slurmd.service  
systemctl enable slurmd.service
```

The last line causes `slurmd` to be started on every boot automatically.



Note: Epilog Script

The standard epilog script will kill all remaining processes of a user on a node. If this behavior is not wanted, disable the standard epilog script.

For further documentation, see the [Quick Start Administrator Guide \(https://slurm.schedmd.com/quickstart_admin.html\)](https://slurm.schedmd.com/quickstart_admin.html) and [Quick Start User Guide \(https://slurm.schedmd.com/quickstart.html\)](https://slurm.schedmd.com/quickstart.html). There is further in-depth documentation on the [Slurm documentation page \(https://slurm.schedmd.com/documentation.html\)](https://slurm.schedmd.com/documentation.html).

8.13 memkind – Heap Manager for Heterogeneous Memory Platforms and Mixed Memory Policies

The `memkind` library is a user-extensible heap manager built on top of `jemalloc` which enables control of memory characteristics and a partitioning of the heap between kinds of memory. The kinds of memory are defined by operating system memory policies that have been applied to virtual address ranges. Memory characteristics supported by `memkind` without user extension include control of NUMA and page size features.

For more information, see:

- the man pages `memkind` and `hbwallow`
- <https://github.com/memkind/memkind>
- <https://memkind.github.io/memkind/>



This tool is only available for x86-64.

8.14 *munge* Authentication

munge allows users to connect as the same user from a machine to any other machine which shares the same secret key. This can be used to set up a cluster of machines between which the user can connect and execute commands without any additional authentication.

The *munge* authentication is based on a single shared key. This key is located under `/etc/munge/munge.key`. At the installation time of the *munge* package an individual munge key is created from the random source `/dev/urandom`. This key has to be the same on all systems that should allow login to each other: To set up *munge* authentication on these machines copy the *munge* key from one machine (ideally a head node of the cluster) to the other machines within this cluster:

```
scp /etc/munge/munge.key root@NODE_N:/etc/munge/munge.key
```

Then enable and start the service *munge* on each machine:

```
systemctl enable munge.service
systemctl start munge.service
```

If several nodes are installed, one key must be selected and synchronized to all the other nodes in the cluster. This key file should belong to the *munge* user and must have the access rights `0400`.

8.15 *mrsh*/*mrlogin* – Remote Login Using *munge* Authentication

mrsh is a set of remote shell programs using the *munge* authentication instead of reserved ports for security.

It can be used as a drop-in replacement for *rsh* and *rlogin*.

To install *mrsh*, do the following:

- If only the *mrsh* client is required (without allowing remote login to this machine), use: **zypper in mrsh**.
- To allow logging in to a machine, the server needs to be installed: **zypper in mrsh-server**.
- To get a drop-in replacement for *rsh* and *rlogin*, run: **zypper in mrsh-rsh-server-compatible** or **zypper in mrsh-rsh-compatible**.

To set up a cluster of machines allowing remote login from each other, first follow the instructions for setting up and starting *munge* authentication in [Section 8.14, “munge Authentication”](#). After *munge* has been successfully started, enable and start **mrlogin** on each machine on which the user will log in:

```
systemctl enable mrlogind.socket mrshd.socket
systemctl start mrlogind.socket mrshd.socket
```

To start mrsh support at boot, run:

```
systemctl enable munge.service
systemctl enable mrlogin.service
```

We do not recommend using *mrsh* when logged in as the user root. This is disabled by default. To enable it anyway, run:

```
echo "mrsh" >> /etc/securetty
echo "mrlogin" >> /etc/securetty
```

9 HPC Libraries

Library packages which support environment modules follow a distinctive naming scheme: all packages have the compiler suite and, if built with MPI support, the MPI flavor in their name: *-[MPI_FLAVOR]-COMPILER-hpc*. To support a parallel installation of multiple versions of a library package, the package name contains the version number (with dots . replaced by underscores _). To simplify the installation of a library, master -packages are supplied which will ensure that the latest version of a package is installed. When these master packages are updated, the latest version of the respective library packages will be installed while leaving previous versions installed. Library packages are split between runtime and compile time packages. The compile time packages typically supply include files and .so-files for shared libraries. Compile time package names end with -devel. For some libraries static (.a) libraries are supplied as well, package names for these end with -devel-static.

As an example: Package names of the ScaLAPACK library version 2.0.2 built with GCC for Open MPI v1:

- library package: libscalapack2_2_0_2-gnu-openmpi1-hpc
- library master package: libscalapack2-gnu-openmpi1-hpc

- development package: libscalapack2_2_0_2-gnu-openmpi1-hpc-devel
- development master package: libscalapack2-gnu-openmpi1-hpc-devel
- static library package: libscalapack2_2_0_2-gnu-openmpi1-hpc-devel-static

(Note that the digit 2 appended to the library name denotes the .so version of the library).

To install a library packages run **zypper in LIBRARY-MASTER-PACKAGE**. To install a development file, run **zypper in LIBRARY-DEVEL-MASTER-PACKAGE**.

Presently, the GNU compiler collection version 4.8 as provided with SUSE Linux Enterprise 15 and the MPI flavors Open MPI v.2 and MVAPICH2 are supported.

9.1 FFTW HPC Library – Discrete Fourier Transforms

FFTW is a C subroutine library for computing the Discrete Fourier Transform (DFT) in one or more dimensions, of both real and complex data, and of arbitrary input size.

This library is available as both a serial and an MPI-enabled variant. This module requires a compiler toolchain module loaded. To select an MPI variant, the respective MPI module needs to be loaded beforehand. To load this module, run:

```
module load fftw3
```

List of master packages:

- libfftw3-gnu-hpc
- fftw3-gnu-hpc-devel
- libfftw3-gnu-openmpi1-hpc
- fftw3-gnu-openmpi1-hpc-devel
- libfftw3-gnu-mvapich2-hpc
- fftw3-gnu-mvapich2-hpc-devel

For general information about Lmod and modules, see [Section 8.7, “Lmod # Lua-based Environment Modules”](#).

9.2 HDF5 HPC Library – Model, Library, File Format for Storing and Managing Data

HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of data types, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and extensible, allowing applications to evolve in their use of HDF5.

There are serial and MPI variants of this library available. All flavors require loading a compiler toolchain module beforehand. The MPI variants also require loading the correct MPI flavor module.

To load the highest available serial version of this module run:

```
module load hdf5
```

When an MPI flavor is loaded, the MPI version of this module can be loaded by:

```
module load phpdf5
```

List of master packages:

- [hdf5-examples](#)
- [hdf5-gnu-hpc-devel](#)
- [libhdf5-gnu-hpc](#)
- [libhdf5_cpp-gnu-hpc](#)
- [libhdf5_fortran-gnu-hpc](#)
- [libhdf5_hl_cpp-gnu-hpc](#)
- [libhdf5_hl_fortran-gnu-hpc](#)
- [hdf5-gnu-openmpi1-hpc-devel](#)
- [libhdf5-gnu-openmpi1-hpc](#)
- [libhdf5_fortran-gnu-openmpi1-hpc](#)
- [libhdf5_hl_fortran-gnu-openmpi1-hpc](#)
- [hdf5-gnu-mvapich2-hpc-devel](#)
- [libhdf5-gnu-mvapich2-hpc](#)

- [libhdf5_fortran-gnu-mvapich2-hpc](#)
- [libhdf5_hl_fortran-gnu-mvapich2-hpc](#)

For general information about Lmod and modules, see [*Section 8.7, “Lmod # Lua-based Environment Modules”*](#).

9.3 NetCDF HPC Library – Implementation of Self-Describing Data Formats

The NetCDF software libraries for C, C++, FORTRAN, and Perl are a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.

[netcdf Packages](#)

The packages with names starting with [netcdf](#) provide C bindings for the NetCDF API. These are available with and without MPI support.

There are serial and MPI variants of this library available. All flavors require loading a compiler toolchain module beforehand. The MPI variants also require loading the correct MPI flavor module.

The MPI variant becomes available when the MPI module is loaded. Both variants require loading a compiler toolchain module beforehand. To load the highest version of the non-MPI [netcdf](#) module, run:

```
module load netcdf
```

To load the highest available MPI version of this module, run:

```
module load pnetcdf
```

List of master packages:

- [netcdf-gnu-hpc](#)
- [netcdf-gnu-hpc-devel](#)
- [netcdf-gnu-hpc](#)
- [netcdf-gnu-hpc-devel](#)
- [netcdf-gnu-openmpi1-hpc](#)

- [netcdf-gnu-openmpi1-hpc-devel](#)
- [netcdf-gnu-mvapich2-hpc](#)
- [netcdf-gnu-mvapich2-hpc-devel](#)

[netcdf-cxx Packages](#)

[netcdf-cxx4](#) provides a C++ binding for the NetCDF API.

This module requires loading a compiler toolchain module beforehand. To load this module, run:

```
module load netcdf-cxx4
```

List of master packages:

- [libnetcdf-cxx4-gnu-hpc](#)
- [libnetcdf-cxx4-gnu-hpc-devel](#)
- [netcdf-cxx4-gnu-hpc-tools](#)

[netcdf-fortran Packages](#)

The [netcdf-fortran](#) packages provide FORTRAN bindings for the NetCDF API, with and without MPI support.

For More Information

For general information about Lmod and modules, see [*Section 8.7, "Lmod # Lua-based Environment Modules"*](#).

9.4 NumPy Python Library

NumPy is a general-purpose array-processing package designed to efficiently manipulate large multi-dimensional arrays of arbitrary records without sacrificing too much speed for small multi-dimensional arrays.

NumPy is built on the Numeric code base and adds features introduced by numarray as well as an extended C API and the ability to create arrays of arbitrary type which also makes NumPy suitable for interfacing with general-purpose data-base applications.

There are also basic facilities for discrete Fourier transform, basic linear algebra and random number generation.

This package is available both for Python 2 and Python 3. The specific compiler toolchain and MPI library flavor modules must be loaded for this library. The correct library module for the Python version used needs to be specified when loading this module. To load this module, run:

- for Python 2: module load python2-numpy
- for Python 3: module load python3-numpy

List of master packages:

- python2-numpy-gnu-hpc
- python2-numpy-gnu-hpc-devel
- python3-numpy-gnu-hpc
- python3-numpy-gnu-hpc-devel

9.5 OpenBLAS Library – Optimized BLAS Library

OpenBLAS is an optimized BLAS (Basic Linear Algebra Subprograms) library based on Goto-BLAS2 1.3, BSD version. It provides the BLAS API. It is shipped as a package enabled for environment modules and thus requires using Lmod to select a version. There are two variants of this library, an OpenMP-enabled variant and a pthreads variant.

OpenMP-Enabled Variant

The OpenMP variant covers all use cases:

- **Programs using OpenMP.** This requires the OpenMP-enabled library version to function correctly.
- **Programs using pthreads.** This requires an OpenBLAS library without pthread support. This can be achieved with the OpenMP-version. We recommend limiting the number of threads that are used to 1 by setting the environment variable OMP_NUM_THREADS=1.
- **Programs without pthreads and without OpenMP.** Such programs can still take advantage of the OpenMP optimization in the library by linking against the OpenMP variant of the library.

When linking statically, ensure that libgomp.a is included by adding the linker flag -lgomp.

pthread Variant

The pthread variant of the OpenBLAS library can improve the performance of single-threaded programs. The number of threads used can be controlled with the environment variable OPEN-BLAS_NUM_THREADS.

Installation and Usage

This module requires loading a compiler toolchain beforehand. To select the latest version of this module provided, run:

- OpenMP version:

```
module load openblas-pthreads
```

- pthreads version:

```
module load openblas
```

List of master package for:

- libopenblas-gnu-hpc
- libopenblas-gnu-hpc-devel
- libopenblas-pthreads-gnu-hpc
- libopenblas-pthreads-gnu-hpc-devel

For general information about Lmod and modules, see *Section 8.7, "Lmod # Lua-based Environment Modules"*.

9.6 PAPI HPC Library – Consistent Interface for Hardware Performance Counters

PAPI (package papi) provides a tool with a consistent interface and methodology for use of the performance counter hardware found in most major microprocessors.

This package serves all compiler toolchains and does not require a compiler toolchain to be selected. The latest version provided can be selected by running:

```
module load papi
```

List of master packages:

- papi-hpc
- papi-hpc-devel

For general information about Lmod and modules, see [Section 8.7, “Lmod # Lua-based Environment Modules”](#).

9.7 PETSc HPC Library – Solver for Partial Differential Equations

PETSc is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations.

This module requires loading a compiler toolchain as well as an MPI library flavor beforehand. To load this module, run:

```
module load petsc
```

List of master packages:

- libpetsc-gnu-openmpi1-hpc
- petsc-gnu-openmpi1-hpc-devel
- libpetsc-gnu-mvapich2-hpc
- petsc-gnu-mvapich2-hpc-devel

For general information about Lmod and modules, see [Section 8.7, “Lmod # Lua-based Environment Modules”](#).

9.8 ScaLAPACK HPC Library – LAPACK Routines

The library ScaLAPACK (short for "Scalable LAPACK") includes a subset of LAPACK routines designed for distributed memory MIMD-parallel computers.

This library requires loading both a compiler toolchain and an MPI library flavor beforehand. To load this library, run:

```
module load scalapack
```

List of master packages:

- [libblacs2-gnu-openmpi1-hpc](#)
- [libblacs2-gnu-openmpi1-hpc-devel](#)
- [libscalapack2-gnu-openmpi1-hpc](#)
- [libscalapack2-gnu-openmpi1-hpc-devel](#)
- [libblacs2-gnu-mvapich2-hpc](#)
- [libblacs2-gnu-mvapich2-hpc-devel](#)
- [libscalapack2-gnu-mvapich2-hpc](#)
- [libscalapack2-gnu-mvapich2-hpc-devel](#)


For general information about Lmod and modules, see [*Section 8.7, “Lmod # Lua-based Environment Modules”*](#).


10 Legal Notices


SUSE makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, SUSE reserves the right to revise this publication and to make changes to its content, at any time, without the obligation to notify any person or entity of such revisions or changes.

Further, SUSE makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, SUSE reserves the right to make changes to any and all parts of SUSE software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classifications to export, re-export, or import

deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical/biological weaponry end uses. Refer to <https://www.suse.com/company/legal/>  for more information on exporting SUSE software. SUSE assumes no responsibility for your failure to obtain any necessary export approvals.

Copyright © 2017-2018 SUSE LLC. This release notes document is licensed under a Creative Commons Attribution-NoDerivs 3.0 United States License (CC-BY-ND-3.0 US, <https://creativecommons.org/licenses/by-nd/3.0/us/> ).

SUSE has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <https://www.suse.com/company/legal/>  and one or more additional patents or pending patent applications in the U.S. and other countries. For SUSE trademarks, see SUSE Trademark and Service Mark list (<https://www.suse.com/company/legal/> ). All third-party trademarks are the property of their respective owners.