Intel HPC Platform Specification

April 2021

Version 2.0

Chapter 0. Introduction

0.1. Overview

The Intel HPC Platform Specification defines both software and hardware requirements that form foundations for high performance computing solutions. This document takes a modular approach to defining these requirements and allows application developers to work on a known base solution without knowing the specifics of a system. This approach promotes application interoperability and execution on any solution that complies with a given set of requirements. This specification targets the community working with scalable systems, including designers and builders of clusters, programmers, system administrators, and users.

Each section in this document defines a set of requirements and has a unique and versioned section identifier. A section can also be referred to as a Specification Layer.
A reference architecture is a defined set of sections that, when combined, create the basis for a solution. While a given reference architecture requires a specific set of sections, additional sections may be added to further customize the solution. Table 1 below shows the sections that define reference architectures and sections that are optional for a given reference architecture.
A solution that is based on this document can comply with more than one reference architecture.

Any application that is compatible with a given reference architecture shall execute on a solution that is compliant with that reference architecture, without the user or the system administrator having to worry about details of the systems.

Table 1 Mapping of the section identifiers to a reference architecture.
Sections marked "✓" are required, sections marked "✚" are conditionally required, and sections marked "○" are optional. Please refer to each section for details.

Reference Architecture		Section Number	Section Identifier and Specification Layer
HPC Cluster	HPC AI Cluster		
✓	✓	1.1	`core-2.0`
✓	✓	1.2	`core-intel-runtime-2.0`
✓	✓	2.1.1	`hpc-cluster-2.0`
✓	○	2.1.2	`compat-hpc-cluster-2.0`
○	✓	2.2	`compat-hpcai-2.0`
✚	✓	3.1	`high-performance-fabric-2.0`
○	○	3.2.1	`vis-core-2.0`
○	○	3.2.2	`vis-single-node-2.0`
○	○	3.2.3	`vis-cluster-2.0`
○	○	3.3	`system-management-2.0`

The platform specification documents industry best practices for Intel-based solutions across a wide domain of application spaces. Currently, the sections in this document define the following requirements:

- Sections in Chapter 1 define the core requirements common to all reference architectures.
- Section 2.1 defines the requirements for a classic HPC Cluster reference architecture.
- Section 2.2 defines the requirements for an HPC AI Cluster reference architecture.
- Sections in Chapters 3 define several capabilities that can enhance a reference architecture.

The reference architecture for a classic HPC Cluster offers application compatibility for a wide range of applications used in HPC. The HPC AI Cluster reference architecture offers dedicated application compatibility for HPC AI Clusters.

An implementation may choose the sections to which it is compliant, subject to dependencies between sections. However, an implementation claiming compliance to a section must satisfy all the requirements of the section. In cases where requirements overlap between sections, the implementation must be compliant to the most restrictive requirement. Each section has a corresponding identifier that indicates compliance with the section requirements (see section 1.1.1 for details). These identifiers allow applications, administrators, and users to discover the capabilities of a given solution implementation.

Future versions of this document will also include reference architectures that span small systems through supercomputers and define requirements for additional dedicated workloads.

0.1.1. Conventions

A. Wording

The key words/phrases "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119[1].

Text with a gray background is advisory.

This is advisory text.

B. Units

This document has standardized on using binary units for values representing computational values in bytes and bits. Binary units are defined by IEC 80000-13:2008 and IEEE 1541-2002. These definitions include the Ki (kibi- 2^{10}), Mi (mebi- 2^{20}), and Gi (gibi- 2^{30}) prefixes.

The computing industry uses decimal units for both decimal and binary values, leading to confusion of actual values. At small quantities, the difference between binary and decimal units is minimal, i.e. 1.024 KB = 1 KiB. However, larger values introduce significant discrepancy, i.e. 1 TiB is approximately 10% greater than 1 TB.

As a technical specification, all defined values must be precise; therefore, this document uses binary units when the underlying architecture uses binary format. This includes, but is not limited to, memory, persistent storage, and data transfer. An exception will be made when referencing other specifications if values in those specifications accurately use different units.

When this document uses SI decimal units (K, M, G, etc.), values will always be powers of 10.

C. Power States

This document shall refer to system power states using ACPI (Advanced Configuration and Power Interface) terminology.
Power states are defined under the ACPI Specification 6.2. errata A[2]

The following power states may be referenced in the Intel HPC Platform Specification requirements.

- S0: Working. System is powered on. Also referenced as G0.
- S1-S4: System sleeping states. The system is in a lower-power state, but retains context allowing it to resume. In particular, the S1 state retains full context for low-latency resumption of operations.
- S5: System is powered-off, with no saved context, and a full boot sequence is required to restore operation. Power to the system is maintained, allowing for out-of-band operations.
- G3: System is unpowered, except for RTC battery.

[1] RFC 2119: https://www.ietf.org/rfc/rfc2119.txt

[2] ACPI Specification 6.2. errata A: http://www.uefi.org/sites/default/files/resources/ACPI%206_2_A_Sept29.pdf

0.1.2. Terminology

COTS

Commercial off-the-shelf (COTS) components are standard manufactured products, not custom products.

Compute node

A compute node is a node reserved for running user workloads and is the primary computational resource of a system.

Distribution or Linux distribution

A system software distribution comprises an operating system kernel, user-space tools, and libraries, as well as the documentation. The components of the distribution may originate from different sources, but they are packaged together and released by the distribution supplier; the distribution supplier may be a commercial entity or community based.

External node

An external node provides resources to the system but is not managed as part of the system.

Fully-qualified path name

A fully-qualified path name is the full path of a directory or file, including the root directory in the virtual filesystem hierarchy.

Head node

Some systems combine the logical functions of login, management, and/or service nodes into a single physical system known as the head node.

Interconnected nodes

If one node is capable of communicating to another node (directly or indirectly with switch hops) via a network fabric, then they are interconnected.

Job

A job is a user workload running on one or more allocated compute nodes. Depending on the system configuration, a job may be launched as a script submitted to the resource manager / scheduler or may be launched directly from the login node.

Login node

A login node is typically the primary point for user interaction with the system. Users initiate jobs from this node, either by submitting jobs to the resource manager or by directly starting job execution. The login node may be shared with multiple users and serves as an interactive resource to stage input and output files, monitor workloads, and build software.

Managed sensor

A hardware device that provides environmental or event data to Management Control Services.

Managed system

A physical device, such as a switch, server, or appliance that is the target of Management Control Services. Depending on functionality provided, Management Control Services may read information, set configuration, or remotely control a Managed System.

Management channel

A logical communication link from Management Control Services to a Managed System. The physical interface may be dedicated to the management channel or may be shared with other communication protocols. Multiple management channels may exist on the same physical interface.

Management control services

Software or firmware responsible for issuing management commands and for receiving alerts. These may be multiple services or a single service providing all required functions. Nodes that run these services are often referred to as management nodes. In a cluster, management nodes are a type of service node typically used to perform low-level system management and monitoring.

Node

A node is a single system, comprised of processor(s), memory, and network interface(s) all under the view and control of a single Operating System image.

Package manager

A package manager deals with packages, distributions of software and data in archive files. Packages contain metadata, such as the software's name, description of its purpose, version number, vendor, checksum (preferably a cryptographic hash function), and a list of dependencies necessary for the software to run properly. Upon installation, metadata is stored in a local package database. Package managers typically maintain a database of software dependencies and version information to prevent software mismatches and missing prerequisites. An example on a package manager would be the rpm tool.

Package-management system

A package-management system is a collection of software tools, most importantly a package manager, that automates the process of installing, upgrading, configuring, and removing computer programs in a consistent manner.

When using a package-management system for installing software packages, any dependency will be resolved, and a depended software package will be installed as well.

Many recent Linux distributions package a given software into many packages in order to ease overall maintenance. This approach, implicitly, assumes that the packages will be handled by a package-management system.
Using a package-management system for managing software on a node is the only way to ensure a fully functional system.

Examples of package management systems are yum, dnf, or zypper.

Primary channel or Primary management channel

The default management channel providing full administrative privileges and management capabilities.

Provisioning system

The main task of a cluster provisioning system is the controlled deployment of the operating system to the nodes in the cluster. The methods employed by the provisioning systems to deploy, and maintain, the operating system on the cluster nodes vary.

Service node

A service node provides non-computational resources to the system in support of a job. Users do not typically access a service node directly, but instead through the service that the node supplies.

Runtime Library "SONAME"

The "SONAME" of a runtime library is defined in https://tldp.org/HOWTO/Program-Library-HOWTO/shared-libraries.html as:
"The soname has the prefix "lib", the name of the library, the phrase ".so", followed by a period and a version number that is incremented whenever the interface changes."

User-accessible node

Any node where a user can directly start processes. User access may be temporarily granted in cases where a resource manager is used to schedule user workloads.

User environment management

User environment management provides a way to manage the execution environment of a user.

For example, user environment management may adapt the PATH and LD_LIBRARY_PATH variables to meet the needs of a given application or sets of applications.

0.1.3. Implications on System Security

Unless otherwise stated, the requirements for Linux kernel versions, runtime libraries, applications, or packages put forth in this specification are referring to Linux kernel versions, runtime libraries, applications, or packages that are provided by enterprise class Linux distributions and their open source equivalents[3]. This ensures strong protection against security vulnerabilities[4].

Enterprise class Linux distributions providers will generally ensure that known vulnerabilities are fixed in packages provided by their respective Linux distributions, even when the upstream maintainer of a given runtime library or application has not yet released an updated version.

A reference implementation should use the latest publicly available version of a runtime library, application, or package, when this runtime library, application, or package is not distributed by an enterprise class Linux distribution.

[3] Examples for enterprise class Linux distributions are: RedHat Enterprise Linux (RHEL) and SUSE Enterprise Linux (SLE). Their open source equivalents are, for example, CentOS, Rocky Linux or Alma Linux and openSUSE.

[4] Known vulnerabilities are listed in the Common Vulnerabilities and Exposures (CVE®) List (https://cve.mitre.org)

Chapter 1. Core

1.1. Core

1.1.1. Configuration Information and Compliance

A. Every node shall provide a text file with the pathname "/etc/intel-hpc-platform-release" containing the `INTEL_HPC_PLATFORM_VERSION` variable using POSIX* shell variable assignment syntax. The value of the variable shall be a colon separated list of section identifiers indicating the sections to which the implementation is claiming to be compliant to.

Advisory

The value of the `INTEL_HPC_PLATFORM_VERSION` variable contained in the /etc/intel-hpc-platform-release file provides the mechanism for a system to list compliance to specific versions of each section. `INTEL_HPC_PLATFORM_VERSION` should be updated accordingly to reflect changes in compliance as a system is modified over time.

B. The `INTEL_HPC_PLATFORM_VERSION` identifier for this section is `core-2.0`.

1.1.2. Hardware

A. Each compute node shall have at least one Intel® 64 architecture processor.

Advisory

This section contains the minimum hardware requirements. Additional hardware may be necessary for a fully functional node.

1.1.3. Operating System

A. The operating system on all nodes in the cluster shall be installed using a package manager.
   a. A node shall be able to boot into the operating system and provide the capability for a non-privileged user to log in.
   b. When a provisioning system is used to deploy cluster nodes, the provision system shall use a package manager or a package-management system for creating the operating system that is being deployed.

1.1.4. Kernel

A. The compute node kernel shall be based on Linux* kernel version 4.12.14 or later.

Advisory

The compute node kernel version should be based on functionality in Linux* kernel version 4.18 or later.

It is strongly recommended to use either the latest kernel provided by a major Linux distribution, the latest Long Term Support (LTS) kernel provided on https://www.kernel.org , or the latest kernel provided on https://www.kernel.org

If a kernel is being used that is not based on a kernel provided by a major Linux distribution it is up to the implementer to ensure that the kernel provides the necessary support for hardware features and fulfills the security requirements that have to be met.

Major commercial and popular Linux* distributions backport kernel features and functionality while maintaining version strings corresponding to an earlier kernel base. The intent of this requirement is to ensure a common base that matches these popular distributions without requiring an exception list for distributions that maintain older version strings. It is up to the distribution to provide a kernel built with appropriate flags for security, hardware functionality, or system features. It is up to the distribution to provide a kernel built with appropriate flags for security, hardware functionality, or system features. It is up to the system provider to select a distribution that includes the desired kernel features.

B. Login and service node kernels shall be based on Linux* kernel version 4.12.14 or later.

Advisory

The login and service nodes kernel version should be based on functionality in Linux* kernel version 4.18 or later.

The Linux* kernel requirements for login and service nodes is separated from compute nodes to permit solutions that use a mix of OS environments. For example, a solution may use commercial distributions for login and service nodes with a comparable open source distribution for compute nodes. In general, the same advisories for compute nodes apply to the kernel requirements for login and service nodes.

1.1.5. Programming Interfaces

A. All required APIs shall be defined using the LP64[5] programming model on all nodes.

Advisory

Where explicitly required for compatibility with existing practice or current program binaries, other programming model APIs may be provided in addition to LP64.

B. The ABI behavior shall conform to the Intel® 64 architecture on all nodes.

Advisory

The Intel® 64 architecture ABI is also known as 'x86_64'.

Where explicitly required for compatibility with existing practice or current program binaries, other ABIs may be provided in addition to Intel® 64 architecture.

1.1.6. Runtime Environment

A. At login, the environment variable `$HOME` shall be set on all nodes to the fully qualified path name of the user's home directory.

B. The environment variable `$TMPDIR` shall be set on all nodes to the fully qualified path name of a node-private temporary directory. If the temporary directory is contained within a shared filesystem, then the value of `$TMPDIR` shall be unique to each node.

Advisory

The value of `$TMPDIR` may be set dynamically as long as the value is set and meets the requirement for the duration that a non-privileged user or job has access to the node.

C. The user environment shall allow multiple runtime environments to coexist on user accessible nodes. A user shall be able to select a self-consistent environment.

Advisory

This includes multiple versions of the same component.

It is left to the implementer to choose which runtime environments are default, if any.

[5] LP64 Programming Model: http://archive.opengroup.org/public/tech/aspen/lp64_wp.htm

The naming convention should ensure consistency across all versions of a given component. Where a version is not applicable, the version may be omitted. A hierarchical organizational scheme should be used to handle dependencies, e.g., on a compiler family.

1.1.7. Application software deployment

Advisory

Application software should be deployed by using the installation procedure of the application software. If that is not possible, a package manager or a package-management system should be used for deploying application software in order to ensure that all dependencies are being resolved that affect the functioning and desired functionality of the application software on a given cluster.

If the installation of an application causes conflicts with the user environment management, such conflicts shall be resolved.

1.2. Core Intel Runtime Environment

The Core Intel Runtime Environment section contains many requirements to ensure that functionality and performance opportunities are present. This section is separated from the core layer as the runtime environment may update at a different frequency.

This section enumerates the requirements for baseline interfaces readily available in common Linux distributions and key Intel software runtime components available free of charge. These requirements are the minimum set of elements that must be present to be compliant.

1.2.1. Configuration Information and Compliance

A. The INTEL_HPC_PLATFORM_VERSION identifier for this section is core-intel-runtime-2.0.

B. If an implementation claims compliance to this section, then INTEL_HPC_PLATFORM_VERSION must also contain the core-2.0 section identifier and meet all corresponding requirements.

1.2.2. Operating System and Kernel

Advisory

The operating system on user accessible nodes should materially conform to the Linux* Standard Base 5.0 core specification[6].

The Linux Standard Base 5.0 was chosen as reference, because it is the only widely accepted standard covering Linux distributions and the APIs and ABIs they shall or should expose.

Because the Linux Standard Base has not evolved over the past years, this document can only refer to selected portions of the Linux Standard Base 5.0 specification.

1.2.3. Programming Interfaces

Advisory

Some applications may require additional or newer versions of these runtimes for compatibility.

A. A materially conformant POSIX.1-2008 API[7] shall be provided on user accessible nodes.

[6] Linux Standard Base 5.0: http://refspecs.linuxfoundation.org/LSB_5.0.0/allspecs.shtml

[7] POSIX.1-2008: http://pubs.opengroup.org/onlinepubs/9699919799/

The following runtime libraries, some of which are defined in the Linux Standard Base* (LSB) 5.0[8], and all runtime libraries the required libraries depend on, shall be provided on user accessible nodes and recognized by the dynamic loader for the Intel® 64 architecture.

Table 2 Name of the required runtime library and the corresponding SONAME of the library.

Library	SONAME of the runtime library
libbz2	libbz2.so.1
libc	libc.so.6
libcrypt	libcrypt.so.1
libdl	libdl.so.2
libexpat	libexpat.so.1
libm	libm.so.6
libncursesw	libncursesw.so.5
libnsl	libnsl.so.1
libnuma	libnuma.so.1
libpanelw	libpanelw.so.5
libpthread	libpthread.so.0
libresolv	libresolv.so.2
librt	librt.so.1
libstdcxx	libstdc++.so.6
libthread_db	libthread_db.so.1
libudev	libudev.so.1
libutil	libutil.so.1
libuuid	libuuid.so.1
libz	libz.so.1
proginterp	/lib64/ld-linux-x86-64.so.2

B. The LP64 version of the following runtime libraries shall be provided on user accessible nodes, with runtime environments configurable using user environment management:
   a. ANSI* standard C/C++ language runtime of the GNU* C Compiler version 8.3 or later
   b. ANSI* standard C/C++ language runtime of Intel® C++ Compiler Classic version 2021.1 or later[9]
   c. Standard Fortran language runtime of the Intel® Fortran Compiler Classic version 2021.1 or later
   d. Intel® oneAPI Math Kernel Library version 2021.1 or later
   e. Intel® oneAPI Threading Building Blocks version 2021.1 or later
   f. Intel® MPI Library Runtime Environment version 2021.1 or later
   g. The Intel® Distribution for Python* scripting language version 2021.1 or later, which is based in Python v3.

Advisory

User environment management should load these runtime libraries by default.

[8] Linux Standard Base Core 5.0 Core Specification: http://refspecs.linuxfoundation.org/LSB_5.0.0/LSB-Core-generic/LSB-Core-generic/requirements.html http://refspecs.linuxfoundation.org/LSB_5.0.0/LSB-Core-AMD64/LSB-Core-AMD64/requirements.html
Linux Standard Base 5.0 Desktop Specification: http://refspecs.linuxfoundation.org/LSB_5.0.0/LSB-Desktop-generic/LSB-Desktop-generic/requirements.html

[9] Information about Intel® oneAPI can be found here: https://software.intel.com/content/www/us/en/develop/tools/oneapi.html

For each component, the runtimes are defined to include all of the runtime libraries distributed with the component. For example, the OpenMP* runtime library is included as part of the Intel® C++ Compiler runtime.

Advisory

As of January 1, 2020, Python v2 was retired by the Python Software Foundation. The Intel® Distribution for Python, which is based on Python v2 should, therefore, not be installed.

C. The ILP64 version of the following runtime libraries shall be provided on user accessible nodes, with runtime environments configurable using user environment management:
   a. Intel® oneAPI DPC++/C++ Compiler version 2021.1 or later
   b. Intel® oneAPI Math Kernel Library version 2021.1 or later
   c. Intel® MPI Library Runtime Environment version 2021.1 or later

1.2.4. Command System and Tools

A. The following subset of the Linux Standard Base* (LSB) 5.0 command system[10] shall be provided on all user accessible nodes:

Table 3 The required subset of Linux Standard Base* (LSB) 5.0 command system

`[`	`date`	`file`	`ln`	`patch`	`tail`
`ar`	`dd`	`find`	`locale`	`pathchk`	`tar`
`awk`	`df`	`fold`	`logger`	`pidof`	`tee`
`basename`	`diff`	`fuser`	`logname`	`printf`	`test`
`bc`	`dirname`	`getconf`	`ls`	`ps`	`time`
`cat`	`dmesg`	`grep`	`mkdir`	`pwd`	`touch`
`chmod`	`du`	`gunzip`	`mkfifo`	`rm`	`tr`
`chown`	`echo`	`gzip`	`mktemp`	`rmdir`	`true`
`cksum`	`ed`	`head`	`more`	`sed`	`uname`
`cmp`	`egrep`	`hostname`	`mv`	`seq`	`uniq`
`comm`	`env`	`iconv`	`nice`	`sh`	`wc`
`cp`	`ex`	`id`	`nl`	`sleep`	`xargs`
`cpio`	`expr`	`join`	`nohup`	`sort`	
`csplit`	`false`	`kill`	`od`	`split`	
`cut`	`fgrep`	`killall`	`paste`	`strings`	

Advisory

[10] Linux Standard Base 5.0 Core Specification: http://refspecs.linuxfoundation.org/LSB_5.0.0/LSB-Core-generic/LSB-Core-generic/command.html

A complete and materially conformant POSIX.1-2008* command system[11] should be provided on all user accessible nodes.

A complete and materially conformant Linux Standard Base* (LSB) 5.0 command system should be provided on all user accessible nodes.

B. The Perl* scripting language version 5.26 or later shall be provided on user accessible nodes.

C. The Tcl scripting language version 8.6 or later shall be provided on user accessible nodes.

[11] POSIX.1-2008 Utilities: https://pubs.opengroup.org/onlinepubs/9699919799.2008edition/idx/utilities.html

Chapter 2.
Base Reference Architectures

2.1. Classic High-Performance Compute Cluster

The “classic” High Performance Compute cluster typically uses commercial off-the-shelf components to form a parallel computing platform. Applications that target this type of system typically use the Message Passing Interface (MPI) over a high-performance message fabric for parallel execution.

Typical clusters often employ a head node that serves to manage the system, is the primary login/interface for users, and provides numerous system-wide functions and utilities. Compute nodes based on Intel® 64 architecture processors are the primary computational resource. Additional nodes may provide specific, non-compute cluster services, such as login nodes, storage nodes, etc. Typically only the head node(s) or the login node(s) are directly accessible by users.

High Performance Compute clusters have at least three distinct communications needs: application messaging, system management, and cluster-wide storage. From an architectural standpoint, each of these communication needs, or networks, has distinct logical requirements. However, implementations may choose to combine one or more of these logical networks in a single physical fabric.

2.1.1. Classic High-Performance Compute Cluster Base

A. Configuration Information and Compliance
   a. The `INTEL_HPC_PLATFORM_VERSION` identifier for this section is `hpc-cluster-2.0`.
   b. If an implementation claims compliance to this section, then `INTEL_HPC_PLATFORM_VERSION` must also contain the `core-2.0` and `core-intel-runtime-2.0` section identifiers and meet all corresponding requirements.

B. Operating System and Kernel

Advisory

In most configurations, the hardware components and software environment of compute nodes should be uniform. Files that specify unique identification or configuration of the compute nodes may be different as needed.

In cases where specialized nodes are desired, a resource manager should be provided and should comprehend any differences.

C. Network Fabrics
   a. On all nodes in the cluster, the network host name of each node shall be consistently resolved to the respective network address.
   b. The management fabrics shall be accessible using the standard IP network stack.
   c. All login nodes shall be externally accessible via SSH.

Advisory

Login nodes are generally connected to an externally accessible network. Stand-alone systems that are not connected to an externally accessible network, that is, systems that only allow console access to the login node, are considered to meet this requirement.

D. Authentication and Access Control
   a. For non-privileged users, all nodes shall operate under a single authentication domain, i.e., once authenticated, one set of credentials shall permit access to the user accessible cluster nodes.

Advisory

This requirement is applicable to the resources allocated to an individual user or user's jobs. Local policies control user access to the system, and this requirement is not meant to dictate those polices. This requirement does not apply to privileged users.

b. Privileged users shall be able to execute commands on all nodes.

c. Non-privileged users, or their jobs, shall be able to execute commands on all compute nodes that the user has access to, or have been allocated to a given user or job by a resource management system.

Advisory

Non-privileged users are not required to be able to access a compute node unless they have been allocated resources on that node.

Non-privileged users are not required to have interactive access to a compute node even if they have been allocated resources on that node.

d. Non-privileged users shall be able to access their data stored locally on currently allocated compute nodes.

Advisory

Non-privileged users are not required to be able to access data stored locally on a compute node unless they have been allocated resources on that node.

2.1.2. High Performance Compute Cluster – Application Compatibility

An ecosystem of highly compatible "classic" HPC clusters provides a consistent application target for application developers. Solutions that comply with this section present a known interface to the application layer. In turn, application developers and vendors can compile and distribute binaries for this target platform, enabling application binary mobility.

While this section is intended primarily for MPI applications distributed as binaries, it is also applicable for MPI applications built from source on the intended system as well as non-MPI workloads.

A. Configuration Information and Compliance

a. The `INTEL_HPC_PLATFORM_VERSION` identifier for this section is `compat-hpc-cluster-2.0`.

b. If an implementation claims compliance to this section, then `INTEL_HPC_PLATFORM_VERSION` must also contain the `hpc-cluster-2.0` section identifier and meet all corresponding requirements.

B. Hardware

Advisory

Minimal hardware requirements are described to ensure functional systems are built from this platform definition. This specification does not guarantee that specific implementations built only to these minimal requirements will provide optimal application performance. Implementers must determine when additional hardware resources beyond this set of minimums may be required to provide optimal application performance.

a. Each compute node shall have a minimum of 64 gibibytes of random-access memory.

Advisory

A total of 128 gibibytes of random-access memory or more per node is recommended.

All memory channels of a server should be populated using identical DIMMS to provide optimal performance. This may correspond to an amount of memory that is higher than the requirement.

It is strongly recommended to follow memory population guides for a given platform.

b. Each compute node shall have access to at least 80 gibibytes of persistent storage.

Advisory

The storage may be implemented as direct access local storage or available over a network.

The storage may be globally visible or node private.

c. Login nodes shall have access to at least 200 gibibytes of persistent storage.

Advisory

The storage may be implemented as direct access local storage or available over a network.

C. Application Environment

a. In order to provide the basic environment for graphical user interfaces, which are commonly used by application software, the following runtime libraries shall be provided on user accessible nodes and recognized by the dynamic loader for the Intel® 64 architecture.

Table 4 Name of the required runtime library and the corresponding SONAME of the library.

Library	SONAME of the runtime library
libGL	libGL.so.1
libGLU	libGLU.so.1
libICE	libICE.so.6
libSM	libSM.so.6
libX11	libX11.so.6
libXau	libXau.so.6
libXcomposite	libXcomposite.so.1
libXcursor	libXcursor.so.1
libXdamage	libXdamage.so.1
libXext	libXext.so.6
libXfixes	libXfixes.so.3
libXft	libXft.so.2
libXi	libXi.so.6
libXmu	libXmu.so.6
libXrandr	libXrandr.so.2
libXrender	libXrender.so.1
libXt	libXt.so.6
libXtst	libXtst.so.6
libXxf86vm	libXxf86vm.so.1
libfontconfig	libfontconfig.so.1
libfreetype	libfreetype.so.6
libjpeg	libjpeg.so.62
libpng	libpng16.so.16
libtiff	libtiff.so.5
libxcb	libxcb.so.1
libX11-xcb	libX11-xcb.so.1
libxcb-glx	libxcb-glx.so.0
libxcb-shape	libxcb-shape.so.0
libxcb-shm	libxcb-shm.so.0
libxcb-xfixes	libxcb-xfixes.so.0

Advisory

If backwards compatibility with applications that require obsolete ABI version of graphics libraries is desired, the following runtime library version should be installed: libpng12.so.0, libpng15.so.15 and libtiff.so.3

Since those libraries are no longer maintained by the upstream project, it is strongly advised to only install the libraries that are provided by enterprise class Linux distributions in order to ensure that potential security risks are mitigated.

D. Storage and File System

   a. Cluster file systems shall provide at least the consistency and access guarantees provided by NFS version 4.0[12].

Advisory

Due to a limitation to the AUTH_SYS mechanism, which is defined in RFC 1831 (https://www.ietf.org/rfc/rfc1831.txt), a user can only belong to a maximum of 16 groups. This limitation can be circumvented by enabling the "--manage-gid" option for the rpc.mountd service of NFS. Please refer to the documentation of nfsd for details on how to do this.

Cluster file systems should support POSIX.1-2008 file semantics[13].

[12] Network File System (NFS) version 4 Protocol (RFC 3530): https://www.ietf.org/rfc/rfc3530.txt

[13] POSIX.1-2008 System Interfaces: http://pubs.opengroup.org/onlinepubs/9699919799/functions/V2_chap02.html

2.2. HPC AI Cluster

A node that provides the capabilities for an HPC AI Cluster typically contains off-the-shelf hardware components. An HPC AI Cluster is building upon the a "Classic" High-Performance Compute Cluster capabilities as described in Section 2.1, adding capability to perform machine learning, deep-learning training models and data analytics. It is possible the HPC AI segment of a cluster to be made of different hardware to the Classic High-Performance Compute Cluster but not required.

2.2.1. HPC AI Cluster Capabilities

A. Configuration and Compliance Information
   a. The `INTEL_HPC_PLATFORM_VERSION` identifier for this section is `compat-hpcai-2.0`.
   b. If an implementation claims compliance to this section, then `INTEL_HPC_PLATFORM_VERSION` must also contain the `hpc-cluster-2.0` and `high-performance-fabric-2.0` section identifiers and meet all corresponding requirements.

B. Tools, Toolkits and Packages
   The following tools shall be provided on all user accessible nodes:
   a. Java version 1.8.0

C. Runtime Environment
   The following runtime libraries shall be provided on user accessible nodes and recognized by the dynamic loader for the Intel® 64 architecture:
   a. oneAPI Deep Neural Network Library (oneDNN) version 1.7 or later
   b. Intel® Distribution of OpenVINO™ toolkit version 2021.1 or later
   c. Intel® Distribution of TensorFlow version 2.3.0 or later, with Horovod
   d. PyTorch for oneDNN version 1.7 or later, with Horovod
   e. Apache Spark 2.4.6 or later

Advisory

It is advised to use the latest available versions of the required component that ensures interoperability of all components.

D. User Environment
   The following environment variables shall be set
   a. PYTORCH_ENVIRONMENT_PATH set to the virtual environment for PyTorch
   b. TENSORFLOW_ENVIRONMENT_PATH set to the virtual environment for Intel® Distribution of TensorFlow
   c. SPARK_ENVIRONMENT_PATH set to the virtual environment for Apache Spark

E. Hardware requirements
   a. Each compute node shall use processors that support Intel® Hyper-Threading Technology and have that capability enabled.
   b. Each compute node shall use processors that provide at least two 512-bit Fused-Multiply-Add (FMA) execution units per core.
   c. Each compute node shall use processors that provide Vector Neural Network Instructions (VNNI).
   d. Each compute node shall have a minimum of one memory module per processor memory channel installed.
   e. Each compute node shall have a minimum of 256 gibibytes of total random-access memory.

Chapter 3. Capabilities

3.1. High Performance Fabric

A high-performance network fabric is the interconnect technology that connects systems through one or more network switches. While the standard TCP/IP or UDP/IP based Ethernet network fabric continues to improve, a High Performance Fabric is defined in this context as technology providing capabilities beyond the standard TCP/IP or UDP/IP protocol to deliver higher bandwidth, lower latency, or higher message rate. For an Ethernet based fabric such technologies could be “RDMA over Converged Ethernet version 2”[14] (RoCEv2) or iWARP[15]. High performance fabric is a key technology used in high performance computing but can be desirable in many other types of solutions. This section defines minimum requirements for solutions that contain high performance fabric technology.

Even though an HPC cluster reference architecture is not required to claim compliance with this section, it is strongly recommended.

3.1.1. Configuration and Compliance Information

A. The `INTEL_HPC_PLATFORM_VERSION` identifier for this section is `high-performance-fabric-2.0`.

B. If an implementation claims compliance to this section, then `INTEL_HPC_PLATFORM_VERSION` must also contain the `core-2.0` section identifier and meet all corresponding requirements.

3.1.2. Open Fabric Interface

A. Each node shall include the OpenFabrics Interfaces* (OFI) `libfabric` library version 1.8.1 or later.

B. The version of `libfabric` used by interconnected nodes shall be consistent.

Advisory

Each node should include `libfabric` version 1.10.0 or later.

3.1.3. Remote Direct Memory Access (RDMA)

Every node shall support remote direct memory access (RDMA) to interconnected nodes.
The following packages shall be installed on every node to support RDMA:

A. `rdma-core` version 22.5 or later

Advisory

`rdma-core` is a recent package that is now the upstream location for "user space components for Linux Kernel's drivers/infiniband subsystem."[16]

B. `infiniband-diags` version 2.1.0 or later

C. All nodes with interconnected fabric shall provide the same version of the packages listed above.

3.1.4. Firmware

The firmware shall be consistent across interconnected nodes in the system for the following hardware:

A. Host Fabric Network Device

B. Fabric switches

3.1.5. TCP/IP and UDP/IP Capabilities

The fabric shall support standard TCP/IP and UDP/IP capabilities, providing a functional protocol adhering to the following:

[14] https://www.roceinitiative.org

[15] https://tools.ietf.org/html/rfc7306

[16] https://github.com/linux-rdma/rdma-core

A. The standard IP network stack functionality as described in RFC1180[17]

B. Berkeley Sockets API (BSD Sockets)

Advisory

The fabric should have standard TCP/IP and UDP/IP capabilities configured and enabled.

3.1.6. Subnet Management

For fabrics that require subnet management, there shall be active subnet management visible to all host fabric network devices.

Advisory

Fabrics that require subnet management should have access to a backup subnet manager.

[17] RFC1180: https://tools.ietf.org/html/rfc1180

3.2. High-Fidelity Visualization

A node that provides capability for High-Fidelity Visualization typically contains off-the-shelf hardware components. Applications for High-Fidelity Visualization use any or all of the Intel® oneAPI Rendering Toolkit open source libraries as a basis: the Intel Embree Ray Tracing Kernel Library, the Intel OSPRay Distributed Ray Tracing Engine library, the Intel Open Image Denoise high-quality denoising filters for images rendered with ray tracing and the Intel Open Volume Kernel Library high-performance volume computation kernels. Details can be found here: https://software.intel.com/en-us/oneapi/render-kit

Advisory

A service or login node shall optionally provide the capabilities to drive at least 6 displays with a resolution of 4096x2160 pixels per display.

3.2.1. High-Fidelity Visualization Base Capabilities

A. Configuration and Compliance Information

a. The `INTEL_HPC_PLATFORM_VERSION` identifier for this section is `vis-core-2.0`.

b. If an implementation claims compliance to this section, then `INTEL_HPC_PLATFORM_VERSION` must also contain the `core-2.0` and `core-intel-runtimes-2.0` section identifiers and meet all corresponding requirements.

B. Runtime Environment

a. High-Fidelity Visualization capabilities require that the following libraries be loaded by default in the environment of a user:

i `rkcommon` version 1.6.0 or later
ii `embree` version 3.12.2 or later
iii `Open VKL (openvkl)` version 0.12.0 or later
iv `Open Image Denoise (oidn)` version 1.3.0 or later

Advisory

It is advised to use the latest available versions of the required packages.

3.2.2. High-Fidelity Visualization Single Node Capabilities

This section defines requirements for a single node that serves as the primary interface for the users, providing all required system-wide functions, utilities, and software tools.

A. Configuration and Compliance Information

a. The `INTEL_HPC_PLATFORM_VERSION` identifier for this section is `vis-single-node-2.0`.

b. If an implementation claims compliance to this section, then `INTEL_HPC_PLATFORM_VERSION` must also contain the `vis-core-2.0` section identifier and meet all corresponding requirements.

B. Hardware requirements

a. The single node shall use processors that support Intel® Hyper-Threading Technology and have that capability enabled.

b. The node shall use processors that provide at least two 512-bit Fused-Multiply-Add (FMA) execution units per core.

c. There shall be a minimum of one memory module per processor memory channel installed.

d. A minimum of 3.5 gibibytes of random-access memory per processor core and a minimum of 64 gibibytes of total random-access memory shall be installed.

e. A minimum of 4 tebibytes of persistent storage shall be available to the node.

3.2.3. High-Fidelity Visualization Multi-Node Capabilities

This section defines requirements for High-Fidelity Visualization multi-node capabilities.

The multi-node capabilities provided by the Intel® oneAPI Rendering Toolkit libraries are the basis for the advanced visualization capabilities provided by tools like VTK, ParaView, VMD, or VisIt.

A. Configuration and Compliance Information
   a. The `INTEL_HPC_PLATFORM_VERSION` identifier for this section is `vis-cluster-2.0`.
   b. If an implementation claims compliance to this section, then `INTEL_HPC_PLATFORM_VERSION` must also contain the `classic-hpc-2.0`, `high-performance-fabric-2.0` and `vis-core-2.0` section identifiers and meet all corresponding requirements.

B. Hardware requirements
   a. All nodes that support Visualization workloads shall use processors that support Intel® Hyper-Threading Technology and have that capability enabled.
   b. The login node shall have a minimum of 6 gibibytes of random-access memory per processor core, with a minimum of 192 gibibytes of total random-access memory.
   c. Each compute node shall have a minimum of 2.5 gibibytes of random-access memory per processor core, with a minimum of 96 gibibytes of total random-access memory.
   d. A minimum of 10 tebibytes of shared fault-tolerant storage shall be accessible to all nodes allocated to a job.

C. Visualization Libraries
   High-Fidelity visualization capabilities require that the following libraries be loaded by default in the environment of a user:
   a. `ospray` version 2.5.0 or later

3.3. System Management

The system management interface provides autonomous monitoring and control. System hardware is accessed independently of the operating system, processor(s), or memory on the system; this is referred to as out-of-band (OOB) management. This provides management and monitoring of the target system without impact to computational performance, as well as system control prior to boot and when the OS is non-functional.
Normally, management capability is provided by a dedicated management processor on the Managed System's mainboard. OOB system control must remain operational even when the main CPU is non-responsive, so that the system can be remotely reset.

While there are many server management implementations, the IPMI (Intelligent Platform Management Interface) is widely implemented by server hardware vendors. All requirements in this section are either mandatory or optional features of IPMI but use of a management solution with similar capabilities is not precluded.

It is expected that future implementations of this section will specifically require conformance to Redfish*, a successor to IPMI, as defined by the Desktop Management Taskforce* (DMTF).

3.3.1. Configuration and Compliance Information

A. The `INTEL_HPC_PLATFORM_VERSION` identifier for this section is `system-management-2.0`.

B. If an implementation claims compliance to this section, then `INTEL_HPC_PLATFORM_VERSION` must also contain the `core-2.0` section identifier and meet all corresponding requirements.

3.3.2. Management Control Services Requirements

A. There shall be a minimum of 1 node hosting management control services.

   Advisory

   Management control services should not be hosted on a compute or enhanced compute node.

   Additional management control services may operate outside of the cluster. External management control services may duplicate or enhance, but not replace, required functionality.

   Implementations on Managed Systems should conform to IPMI v2.0 revision 1.1, including Errata 7, published on April 21, 2015.

   Managed Systems should conform to Redfish* 2017.2 (Redfish Specification 1.3.0). This will include the following:

   a. Conformance to Host Interface Specification 1.0

   b. A minimum of one TCP/IP-enabled interface supporting HTTP Redfish.

   c. A recommendation that the Managed System support HTTP and SMBIOS Type 42.

   d. Storage device inventory and management (Swordfish*)

   e. Read system utilization

   f. Support for translation of IPMI to Redfish (PSME)

B. Each node running Management Control Services shall be able to open a session and issue commands to any Managed System's primary channel.

C. Session control and commands shall be scriptable using UNIX V7 Bourne Shell syntax.

   Advisory

   BASH (Bourne Again SHell) and POSIX shells meet this requirement.

3.3.3. Managed system requirements

A. There shall be 1 primary channel on each Managed System.

Advisory

There may be additional management channels on a Managed System.

A Managed System should provide direct console access, either for redirection of KVM (keyboard/video/mouse) or redirection of a serial (RS232) interface.

B. The primary channel on all Managed Systems shall exist on a single, logical TCP/IP v4 network.

C. The primary channel on a Managed System shall be accessible using the TCP/IP v4 protocol through at least one of the following:

   a. A direct Ethernet or fabric interface on that system

   b. An indirect Ethernet or fabric interface via a system acting as a proxy for the target Managed System. The proxy system shall also comply with the requirements in this section.

Advisory

Managed Systems should implement a dedicated network interface used only for out-of-band management traffic.

D. The primary channel on all Managed Systems shall provide the capability to configure each of the following parameters using the DHCP protocol.

   a. IP address

   b. subnet mask

   c. default gateway

E. The primary channel on all Managed Systems is available when the system is attached to a power source, regardless of the power state of the system.

F. The primary channel on all Managed Systems shall provide the capability to open a session with the ability to perform all commands defined in this section.

Advisory

A Managed System should provide a minimum of three user accounts with the ability to set channel privilege level per user. Available privilege levels should include administrator, read-only (RO), and power-control-only.

The Managed System should support at least 4 simultaneous sessions.

The Managed System should support security options, including communication payload encryption, SHA256 authentication algorithms, and a firmware firewall.

G. A Managed System, through its primary channel, shall provide these capabilities to Management Control Services:

   a. Read identification of the managed system.

Advisory

Identification of the managed system may be any cluster-wide unique value, including hardware serial numbers or user-assigned values. The specific definition of identification is system dependent.

b. Initiate a hard reset
c. Initiate a soft reset
d. Initiate a Power off (S5)
e. Initiate a Power on (S0)
f. Read the current ACPI state
g. Read current power state
h. Activate a visual or audio indicator that allows physical identification of the managed system
i. Configure the setting for the default system state after a power loss (power restore policy)
j. Set the default media option to be used the next time the managed system is booted
k. Read the managed system firmware, uEFI, or BIOS event log.

Advisory

The Managed System should provide the capability to send SNMP (Simple Network Management Protocol) alerts over a management channel.

H. Each Managed System shall provide a report that includes minimally the following:
a. An inventory of baseboard, processors, and system memory.
b. Baseboard firmware versions, including BIOS, uEFI, BMC, and on-board Ethernet.

Advisory

The Read System Information capability should have the capability to report:

- An inventory of all Field-Replaceable Units, including storage devices, power supplies, and add-in cards.
- Active operating system name and version

I. Each Managed System shall provide the capability to read all of the following information for each managed sensor:
a. Sensor Identification
b. Sensor type and capabilities
c. Current measurement, with units
d. Current thresholds set if thresholds are supported
e. Sensor self-test results if the sensor supports self-tests.

J. All functions on the primary channel shall be available on the Managed System that is powered on (S0), in sleep or hibernate state (S1-S4), or soft off (S5) regardless of operating system status.

Advisory

All configuration options for the Managed System primary channel should be readable and configurable directly through a command line shell on that Managed System.

Managed Systems should implement the capability to read a complete list of its supported management commands and capabilities.

Appendices

A. Appendix – Core – Operating System

A node that meets the “Operating System” requirements of the `core-2.0` specification layer will have the following packages installed that provide the libraries that the respective family of operating systems consider a minimal set of required libraries. The table does not contain all packages that the respective operating system considers to be required for a minimal installation.

Table 5 Showing the Operating System package that provides a given runtime library.

Library	RHEL 7 related	RHEL 8 related	SLES 12 related	SLE 15 related	Ubuntu related
libacl.so.1	libacl	libacl	libacl1	libacl1	libacl1
libanl.so.1	glibc	glibc	glibc	glibc	libc6
libattr.so.1	libattr	libattr	libattr1	libattr1	libattr1
libblkid.so.1	libblkid	libblkid	libblkid1	libblkid1	libblkid1
libBrokenLocale.so.1	glibc	glibc	glibc	glibc	libc6
libbz2.so.1	bzip2-libs	bzip2-libs	libbz2-1	libbz2-1	libbz2-1.0
libcap-ng.so.0	libcap-ng	libcap-ng	libcap-ng0	libcap-ng0	libcap-ng0
libcap.so.2	libcap	libcap	libcap2	libcap2	libcap2
libcom_err.so.2	libcom_err	libcom_err	libcom_err2	libcom_err2	libcom-err2
libcrypto.so.10 or libcrypto.so.1.1 or libcrypto.so.1.0.0	openssl-libs	openssl-libs	libopenssl1_0_0	libopenssl1_1	libssl1.1
libcrypt.so.1	glibc	libxcrypt	glibc	glibc	libcrypt1
libc.so.6	glibc	glibc	glibc	glibc	libc6
libdbus-1.so.3	dbus-libs	dbus-libs	libdbus-1-3	libdbus-1-3	libdbus-1-3
libdbus-glib-1.so.2	dbus-glib	dbus-glib	dbus-1-glib	dbus-1-glib	libdbus-glib-1-2
libdl.so.2	glibc	glibc	glibc	glibc	libc6
libe2p.so.2	e2fsprogs-libs	e2fsprogs-libs	libext2fs2	libext2fs2	libext2fs2
libedit.so.0 or libedit.so.2	libedit	libedit	libedit0	libedit0	libedit2
libelf.so.1	elfutils-libelf	elfutils-libelf	libelf1	libelf1	libelf1
libexpat.so.1	expat	expat	libexpat1	libexpat1	libexpat1
libext2fs.so.2	e2fsprogs-libs	e2fsprogs-libs	libext2fs2	libext2fs2	libext2fs2
libffi.so.4 or libffi.so.6 or libffi.so.7	libffi	libffi	libffi4	libffi7	libffi7
libform.so.5 or libform.so.6	ncurses-libs	ncurses-libs	libncurses5 libncurses6	libncurses6	libncurses6
libformw.so.5 or libformw.so.6	ncurses-libs	ncurses-libs	libncurses5 libncurses6	libncurses6	libncursesw6
libfreetype.so.6	freetype	freetype	libfreetype6	libfreetype6	libfreetype6
libgcc_s.so.1	libgcc	libgcc	libgcc_s1	libgcc_s1	libgcc-s1
libgcrypt.so.11 or libgcrypt.so.20	libgcrypt	libgcrypt	libgcrypt20	libgcrypt20	libgcrypt20

Library	RHEL 7 related	RHEL 8 related	SLES 12 related	SLE 15 related	Ubuntu related
libgio-2.0.so.0	glib2	glib2	libgio-2_0-0	libgio-2_0-0	libglib2.0-0
libglib-2.0.so.0	glib2	glib2	libglib-2_0-0	libglib-2_0-0	libglib2.0-0
libgmodule-2.0.so.0	glib2	glib2	libgmodule-2_0-0	libgmodule-2_0-0	libglib2.0-0
libgmp.so.10	gmp	gmp	libgmp10	libgmp10	libgmp10
libgobject-2.0.so.0	glib2	glib2	libgobject-2_0-0	libgobject-2_0-0	libglib2.0-0
libgpgme.so.11	gpgme	gpgme	libgpgme11	libgpgme11	libgpgme11
libhistory.so.5 or libhistory.so.6 or libhistory.so.7 or libhistory.so.8	readline	readline	libreadline6	libreadline7	libreadline5 libreadline8
libip4tc.so.0 or libip4tc.so.2	iptables	iptables-libs	libiptc0	libiptc0	libip4tc2
libip6tc.so.0 or libip6tc.so.2	iptables	iptables-libs	libiptc0	libiptc0	libip6tc2
libkeyutils.so.1	keyutils-libs	keyutils-libs	libkeyutils1	libkeyutils1	libkeyutils1
libkmod.so.2	kmod-libs	kmod-libs	libkmod2	libkmod2	libkmod2
liblzma.so.5	xz-libs	xz-libs	liblzma5	liblzma5	liblzma5
liblzo2.so.2	lzo	lzo	liblzo2-2	liblzo2-2	liblzo2-2
libmagic.so.1	file-libs	file-libs	libmagic1	libmagic1	libmagic1
libmenu.so.5 or libmenu.so.6	ncurses-libs	ncurses-libs	libncurses5 libncurses6	libncurses6	libncurses6
libmenuw.so.5 or libmenuw.so.6	ncurses-libs	ncurses-libs	libncurses5 libncurses6	libncurses6	libncursesw6
libmount.so.1	libmount	libmount	libmount1	libmount1	libmount1
libm.so.6	glibc	glibc	glibc	glibc	libc6
libncurses.so.5 or libncurses.so.6	ncurses-libs	ncurses-libs	libncurses5 libncurses6	libncurses6	libncurses6
libncursesw.so.5 or libncursesw.so.6	ncurses-libs	ncurses-libs	libncurses5 libncurses6	libncurses6	libncursesw6
libnsl.so.1 or libnsl.so.2	glibc	libnsl2	glibc	glibc libnsl2	libc6
libnss_compat.so.2	glibc	glibc	glibc	glibc	libc6
libnss_dns.so.2	glibc	glibc	glibc	glibc	libc6
libnss_files.so.2	glibc	glibc	glibc	glibc	libc6
libpam.so.0	pam	pam	pam	pam	libpam0g
libpanel.so.5 or libpanel.so.6	ncurses-libs	ncurses-libs	libncurses5 libncurses6	libncurses6	libncurses6
libpanelw.so.5 or libpanelw.so.6	ncurses-libs	ncurses-libs	libncurses5 libncurses6	libncurses6	libncursesw6
libpcre.so.1 or libpcre.so.3	pcre	pcre	libpcre1	libpcre1	libpcre3

Library	RHEL 7 related	RHEL 8 related	SLES 12 related	SLE 15 related	Ubuntu related
libpng15.so.15	libpng				
libpng16.so.16		libpng	libpng16-16	libpng16-16	libpng16-16
libprocps.so.3 or libprocps.so.4 or libprocps.so.6 or libprocps.so.7 or libprocps.so.8	procps-ng	procps-ng	libprocps3	libprocps6	libprocps8
libpthread.so.0	glibc	glibc	glibc	glibc	libc6
libreadline.so.5 or libreadline.so.6 or libreadline.so.7 or libreadline.so.8	readline	readline	libreadline6	libreadline7	libreadline5 libreadline8
libresolv.so.2	glibc	glibc	glibc	glibc	libc6
librt.so.1	glibc	glibc	glibc	glibc	libc6
libSegFault.so	glibc	glibc	glibc	glibc	libc6
libssl.so.10 or libssl.so.1.1 or libssl.so.1.0.0	openssl-libs	openssl-libs	libopenssl1_0_0	libopenssl1_1	libssl1.1
libstdc++.so.6	libstdc++	libstdc++	libstdc++6	libstdc++6	libstdc++6
libthread_db.so.1	glibc	glibc	glibc	glibc	libc6
libtic.so.5 or libtic.so.6	ncurses-libs	ncurses-libs	libncurses5 libncurses6	libncurses6	libtinfo6
libtinfo.so.5 or libtinfo.so.6	ncurses-libs	ncurses-libs	libncurses5 libncurses6	libncurses6	libtinfo6
libudev.so.1	systemd-libs	systemd-libs	libudev1	libudev1	libudev1
libutil.so.1	glibc	glibc	glibc	glibc	libc6
libuuid.so.1	libuuid	libuuid	libuuid1	libuuid1	libuuid1
libz.so.1	zlib	zlib	libz1	libz1	zlib1g

B. Appendix - High Performance Compute Cluster Application Compatibility – Application Environment

In order to meet the requirements put forth in the `compat-hpc-cluster-2.0` specification layer, the packages listed in Table 6 can be installed. When all dependencies of those packages are resolved and installed, the runtime libraries that are required in the `compat-hpc-cluster-2.0` specification layer will be installed.

Table 6 Operating System provided packages that will install all required runtime libraries that are required in the compat-hpc-cluster-2.0 specification layer

RHEL related	SLES related	Ubuntu related
xterm	xterm	xterm
motif	motif	motif-clients
xorg-x11-server-Xorg	xorg-x11-server	xserver-xorg
xorg -x11-utils		

These packages require dependent packages that provide the following runtime libraries that are either required by the `compat-hpc-cluster-2.0` specification layer, are dependencies of required runtime libraries, or are generally considered to be important, yet did not meet the criteria for being a required runtime library in the `compat-hpc-cluster-2.0` specification layer.

Table 7 The list of packages and the runtime library, or libraries, they will provide.

Package names for			Library
RHEL related	SLE related	Ubuntu related	
fontconfig	fontconfig	libfontconfig1	libfontconfig.so.1
libICE	libICE6	libice6	libICE.so.6
libSM	libSM6	libsm6	libSM.so.6
libX11	libX11-6	libx11-6	libX11.so.6
libXau	libXau6	libxau6	libXau.so.6
libXaw	libXaw7	libxaw7	libXaw.so.7
libXcursor	libXcursor1	libxcursor1	libXcursor.so.1
libXdamage	libXdamage1	libxdamage1	libXdamage.so.1
libXdmcp	libXdmcp6	libxdmcp6	libXdmcp.so.6
libXext	libXext6	libxext6	libXext.so.6
libXfixes	libXfixes3	libxfixes3	libXfixes.so.3
libXfont2	libXfont2-2	libxfont2	libXfont2.so.2
libXft	libXft2	libxft2	libXft.so.2
libXi	libXi6	libxi6	libXi.so.6
libXinerama	libXinerama1	libxinerama1	libXinerama.so.1
libXmu	libXmu6	libxmu6	libXmu.so.6

Package names for			Library
RHEL related	**SLE related**	**Ubuntu related**	
libXxf86vm	libXxf86vm1	libxxf86vm1	libXxf86vm.so.1
libXpm	libXpm4	libxpm4	libXpm.so.4
libXrandr	libXrandr2	libxrandr2	libXrandr.so.2
libXrender	libXrender1	libxrender1	libXrender.so.1
libXt	libXt6	libxt6	libXt.so.6
libXtst	libXtst6	libxtst6	libXtst.so.6
libXv	libXv1	libxv1	libXv.so.1
libepoxy	libepoxy0	libepoxy0	libepoxy.so.0
libfontenc	libfontenc1	libfontenc1	libfontenc.so.1
libglvnd	libglvnd	libglvnd0	libGLdispatch.so.0
libjpeg-turbo	libjpeg8	libjpeg-turbo8	libjpeg.so.62 libjpeg.so.8
libxcb	libxcb1	libxcb1	libxcb.so.1
libxkbfile	libxkbfile1	libxkbfile1	libxkbfile.so.1
libxshmfence	libxshmfence1	libxshmfence1	libxshmfence.so.1
libglvnd-glx	libglvnd	libgl1	libGL.so.1
libglvnd-glx	libglvnd	libglx1	libGLX.so.0

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries.
Other names and brands may be claimed as the property of others.