Important Notice

The members of the Intel® Cluster Ready development team will carefully consider any and all feedback from responsible parties. Provided feedback will be deemed to be public information which Intel Corporation is free to use for any and all purposes.

Please do not provide feedback which you do not wish to be used and included in a future version of the specification. This document is intended for wide distribution and implementation by third parties.

Please send your feedback to cluster@intel.com.
Legal Notices

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Requires a system with a 64-bit enabled processor, chipset, BIOS and software. Performance will vary depending on the specific hardware and software you use. Consult your PC manufacturer for more information. For more information, visit http://www.intel.com/info/em64t

Intel® Virtualization Technology requires a computer system with an enabled Intel® processor, BIOS, and virtual machine monitor (VMM). Functionality, performance or other benefits will vary depending on hardware and software configurations. Software applications may not be compatible with all operating systems. Consult your PC manufacturer. For more information, visit http://www.intel.com/go/virtualization

Intel, Intel Xeon, Intel Xeon Phi, Intel Core microarchitecture, Intel64, Intel Pentium, Intel Pentium-D, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2006-2014, Intel Corporation. All rights reserved.
# Contents

1. **The Intel® Cluster Ready Architecture Overview** .......................................................... 3
   1.1 Key Architectural Characteristics .............................................................................. 4
   1.2 Conformance with Industry Standards and Common Practices ................................. 6
   1.3 IP Addressing ........................................................................................................... 6

2. **Definitions** ...................................................................................................................... 7

3. **Terminology** .................................................................................................................... 9

4. **Base Requirements** ...................................................................................................... 15
   4.1 Base Cluster Requirements ....................................................................................... 15
   4.2 Base Software Requirements .................................................................................... 15
   4.3 Base Hardware Requirements ................................................................................... 17

5. **Pre-Boot Requirements** ............................................................................................... 18
   5.1 Network Boot ............................................................................................................ 18
   5.2 Remote Access ......................................................................................................... 18

6. **Cluster Functions** ......................................................................................................... 19
   6.1 OS Kernel .................................................................................................................. 19
   6.2 OS Interface and Basic Runtime Environment .......................................................... 19
   6.3 Command System and Tools .................................................................................... 21
   6.4 Message Passing Layer ............................................................................................. 21
   6.5 Management Fabric .................................................................................................. 22
   6.6 Message Fabric ........................................................................................................ 22
   6.7 Remote Execution Environment ................................................................................ 22
   6.8 Storage and File System Interface .......................................................................... 22
   6.9 Cluster Provisioning System ...................................................................................... 23

7. **Application Requirements** ............................................................................................ 24
   7.1 Installation ................................................................................................................ 24

8. **Optional Features** ....................................................................................................... 25
   8.1 Pre-Boot Requirements ............................................................................................. 25
   8.2 Remote Execution Environment ................................................................................ 25
   8.3 Cluster-Wide Resource Manager .............................................................................. 25
   8.4 Cluster Wide Job Scheduler ..................................................................................... 27
   8.5 General purpose coprocessor ................................................................................... 27

9. **Advisories** ..................................................................................................................... 29
   9.1 Base Cluster Requirements – advisory statement ...................................................... 29
   9.2 Base Software Requirements – advisory statement .................................................. 29
   9.3 Base Hardware Requirements – advisory statement .................................................. 30
   9.4 OS Interface, Basic Runtime Environment – advisory statement ............................. 30
   9.5 Cluster Boot and Provisioning – advisory statement .................................................. 30
   9.6 Command System and Tools – advisory statement ................................................... 30
   9.7 Message Passing Layer – advisory statement ............................................................ 31
   9.8 Message Fabric – advisory statement ...................................................................... 31
   9.9 Remote Execution Environment – advisory statement ............................................. 31
   9.10 Storage and File System Interface – advisory statement .......................................... 32
   9.11 Cluster Installation System – advisory statement ..................................................... 32
   9.12 Cluster-Wide Resource Manager – advisory statement ............................................ 32
   9.13 Cluster-Wide Job Scheduler – advisory statement ................................................... 32
   9.14 Environment Variables – advisory statement ........................................................... 33
   9.15 Application Requirements – advisory statement ..................................................... 33
   9.16 Management, Monitoring, and Performance – advisory statement ....................... 33
   9.17 General purpose coprocessors – advisory statement ................................................ 33

10. **Assumptions** ............................................................................................................... 35
    10.1 Relationship to ISO/IEC 9945 POSIX* ..................................................................... 35
    10.2 Relationship to Linux* Standard Base ..................................................................... 35
    10.3 Licensing and Activation ......................................................................................... 35

*Other names and brands may be claimed as the property of others
11. References ........................................................................................................................................ 36
    11.1 Normative References .............................................................................................................. 36
    11.2 Bibliographic References ........................................................................................................... 37

12. Change History ................................................................................................................................ 38
    12.1 Version 1.0 (May 24, 2007) ....................................................................................................... 38
    12.2 Version 1.0.1 (June 28, 2007) ................................................................................................... 38
    12.3 Version 1.0.2 (October 23, 2007) ............................................................................................. 38
    12.4 Version 1.1 (June 27, 2008) ....................................................................................................... 38
    12.5 Version 1.2 (March 24, 2011) ................................................................................................... 38
    12.6 Version 1.3 (July 26, 2013) ....................................................................................................... 39
    12.7 Version 1.3.1 (June 20, 2014) .................................................................................................. 41
1. The Intel® Cluster Ready Architecture Overview

This document describes the collection of hardware and software components that together define the Intel® Cluster Ready platform and implementations of some of the components within that architecture. It is meant to describe a collection of hardware and software components that together produce a cluster computing platform that an end user can deploy. While other operating systems and environments may be considered in the future with similar or different bindings, this particular document is bound to clusters implemented with Linux* systems described by the Linux* Standard Base (LSB*).

At different levels of detail, the architecture describes the view of that platform as seen by an end-user, an administrator, a software developer, or a hardware engineer. The architecture encompasses different instances of specific components. A primary objective of the Intel® Cluster Ready architecture is that an application written on one implementation of the Intel® Cluster Ready architecture can be moved to another implementation with confidence that it will function correctly, as well as meet some nominal performance expectations. The design criterion for Intel® Cluster Ready is to go beyond pure portability, i.e., recompiling and linking, and increase target application binary mobility, i.e., running the same binary on multiple platforms by more precisely defining the target system.

Individual hardware providers, original equipment manufacturers (OEMs), and system integrators are able to take the Intel® Cluster Ready Architecture and implement cluster platform instances that have a high degree of commonality, interoperability, and most importantly, application binary mobility. Furthermore, the Intel® Cluster Ready architecture is more than just a collection of application programming interfaces (APIs). While an API defines system calls and services used by an application, the Intel® Cluster Ready architecture includes the entire environment, including management of nodes, resources, node provisioning, etc.

The Intel® Cluster Ready architecture is made up of ingredients such as the OS (operating system) kernel, runtime environment, and installer (see Figure 1). Another objective of the Intel® Cluster Ready architecture is to facilitate the replacement of different components. Specific components may have their own architectures and implementations. For instance, the OS kernel could be a Linux* 2.6-based kernel, but the OS interface, commands and systems runtime environment might be a distribution from SUSE, Red Hat, or Ubuntu.
1.1 Key Architectural Characteristics

The basic hardware building block is called a cluster node. Each node is an instance of a standard volume server that is a complete computer. Unlike SMPs, a cluster has a disjoint memory space. This is formally known as no remote memory access (NORMA) hardware architecture\(^1\). In fact, a cluster node can be a shared memory multiprocessor system. A program running on separate cluster nodes must communicate over a network to share data, whereas in shared memory architecture, the data is directly available to all processors.

1. Fabric and Message Layer Independence

Two key elements of the Intel® Cluster Ready architecture are the hardware message fabric and software messaging layer. The hardware components that make up the message fabric have characteristics that can be balanced between performance and price. However, the device-level APIs for two interconnect fabrics may be fundamentally different. Until recent years, clusters exposed these differences at the application layer.

The initial scope of the Intel® Cluster Ready architecture is high performance technical computing. As such, the Intel® Cluster Ready architecture uses a message layer based on the Message Passing Interface (MPI, http://www.mpi-forum.org/docs/docs.html). It further adds a requirement on the message layer implementation that differences in the device-level API be hidden from the application code. An example of an implementation of a message layer that meets this requirement is the Intel® MPI Library (http://www.intel.com/go/mpi/). It is possible that the Intel® Cluster Ready scope may expand to encompass additional segments such as enterprise computing.

An important idea in the Intel® Cluster Ready architecture is that applications need not have special programming or alternate binaries for different message fabrics. Thus, Intel® Cluster Ready compliant applications obey the message layer independence property and enable application binary mobility between different Intel® Cluster Ready implementations.

![Diagram of cluster with management, message, and storage networks]

Figure 2: An example of a cluster with all three networks (management, message, and storage).

2. Separate Message, Management and Storage Network Planes

\(^1\) A more detailed description of parallel computer architectures can be found in Chapter 9 of Hennessy and Patterson [Hennessy1994].
Figure 2 reveals another important characteristic of the Intel® Cluster Ready architecture, which is the separation of the three networking systems: the message network, the management network, and the storage network. From an architectural standpoint each of these networks are considered to have separate requirements. Intel® Cluster Ready implementations can choose to combine one or more of the networks in order to trade off some characteristics—typically price—at the expense of performance for some applications. As an example, an implementation might collapse all three networks to a single Gigabit Ethernet fabric, with the expense of sub-optimal performance for message and storage traffic.

3. One to Many

Figure 3 displays the final important idea of the Intel® Cluster Ready architecture: the concept of “one to many.” Starting at the top of the solution stack, different ISVs register applications that run using various APIs. These binaries need to be checked and validated by their developers once on a single Intel® Cluster Ready implementation, and they can have confidence that one binary will run on many implementations. Similarly, the certified cluster platform providers develop and certify a hardware and system software cluster configuration. These platform integrators are free to build implementations of that configuration and have confidence that many different ISV codes will run on their implementations of the configuration. There is no expectation that ISVs will need to port and verify their codes for each possible implementation.

These two “one-to-many” qualities are provided by the components that make up the Intel® Cluster Ready stack. System components can range from a single vendor’s tightly-integrated software suite to inter-operational selections from different vendors and projects. Implementations may extend or add components if they do not adversely impact functionality or compliance of the rest of the stack. The stack is a certified cluster platform, providing a well-defined interface to applications. Applications written and verified to use this stack comprise the final components of a complete cluster solution.
1.2 Conformance with Industry Standards and Common Practices
The Intel® Cluster Ready specification calls for conformance to many industry standards as the means to create a common cluster ecosystem. It is beyond the scope of this specification to recite or maintain those standards, such as ANSI® C, ISO® Fortran 95, POSIX®, or LSB®, or to examine and specify all facets of those standards. Each component in the cluster ecosystem will need to make its own conformance statements where applicable.

This specification is not intended to enforce strict adherence to those industry standards. A key point is the use of the words “materially compliant” where “materially” is defined in the sense of non-compliance not causing any substantial loss. For example, the intent of requiring a POSIX*-compliant operating system is not meant to exclude Linux* distributions that are considered “mostly” POSIX*-compliant. It is the intention to use POSIX* compliance as the method to enforce a common ecosystem that allows for healthy growth of cluster technologies and the industry. Similarly, many cluster components are developed and maintained by the free and open source software communities and may not be formally verified against some of those standards. This specification is not meant to automatically exclude such components that may fall short of full compliance with standards; the authors reserve the right of judgment to reject such components based on non-compliance to industry standards when non-compliance causes unexpected behaviors or incorrect results.

1.3 IP Addressing
Most computing systems can support both the IPv4 and IPv6 protocols. Linux distributions typically enable both IPv4 and IPv6 addressing by default, but permit administrators to disable either when a system is deployed. The infrastructure transition from IPv4 to IPv6 is an independent action from cluster deployment, although it does present a challenge to providing a broad Intel® Cluster Ready ecosystem.

Many of today’s applications exclusively implement IPv4 addresses. Since most operating environments for HPC have operating systems that support both IPv4 and IPv6, Intel recommends application developers permit IP addresses to be specified using either protocol and tested against both. If an application has a preference, then the application should provide a mechanism for that particular protocol to be specified.
2. Definitions

For the purposes of this document, the following definitions apply²:

Can

Be able to; there is a possibility of; it is possible to

Cannot

Be unable to; there is not a possibility of; it is not possible to

Gibibit

A gibibit is the contraction of “giga” binary bit (binary digit) and is a unit of information

1 gibibit = 2³⁰ bits = 1,073,741,824 bits

Gibibyte

A gibibyte is the contraction of “giga” binary byte and is a unit of information

1 gibibyte = 2³⁰ bytes = 1,073,741,824 bytes

May

Is permitted; is allowed; is permissible

Need not

[Omission] is permitted; is allowed; is permissible

Provided

Must be supplied by an implementer in the default installation with a properly configured runtime environment and available to all users and developers

Note: This does not require that a private copy of software or files exist in local storage on every node of the cluster. Shared file storage, such as a global /opt or /usr/local/, is sufficient.

Shall

Is to; is required to; it is required that; has to; only...is permitted; it is necessary

Shall not

Is not allowed; is required to be not; is required that...be not; is not to be

Should

It is recommended that; ought to

Should not

It is not recommended that; ought not to

Undefined, unspecified

Describes a value or behavior that is not defined by this document which results from use of an invalid program construct or invalid data input. The value or behavior may vary among implementations that conform to this document. Thus application programmers should not rely on the existence of the value or behavior. An application writer that relies on such a value or behavior cannot be assured that the application will properly execute across conforming implementations.
3. **Terminology**

When discussing cluster procedures the following terms apply.

**Application Binary Interface (ABI)**

The binary-level interface between an application program and the operating system, between an application and its libraries, or between component parts of the application (except where explicitly noted) defined in terms of 64-bit Intel® 64 Processor family.

**Beowulf-style cluster**

A form of clustered or distributed compute solution using unique operating systems that do not appear as a single computing system; contrast this to single system image (SSI). Explicit communications are used to interact with the various nodes of the systems, allowing each node to operate independently.

**Application Programming Interface (API)**

The interface between source code and libraries.

**Cluster design**

A defined set of hardware and software components and the build method that integrates those components into a functional cluster solution.

**Compute node**

The primary computational resource of a cluster that is provisioned by the cluster installation system and managed by the cluster management system. Users do not typically access a compute node directly but rather the resource manager—or in a SSI system, the cluster operating system—places work on the node as appropriate.

**Direct Access Provider Library (DAPL)**

A transport- and platform-independent API for using the Remote Direct Memory Access (RDMA) capabilities of modern fabrics such as InfiniBand* and Ethernet.

**Direct access storage**

Any storage directly accessible to a node in the pre- and post-boot environments. This typically means storage devices that are supported by the BIOS and locally attached such as IDE/ATA, SCSI, SATA and FC style disks, but may also include iSCSI, iSER, SRP and other virtual disk technologies.

**Distribution or system software distribution**

A system software distribution comprises an operating system kernel, user-space tools and libraries, as well as the documentation. Client oriented distributions may also include a desktop environment and associated window system and support. Distributions may define an installation system specific to that distribution. The components of the distribution may be supplied by different developer teams, but packaged together and released by the distribution supplier.

**External node**

An external node provides resources to the cluster but is not provisioned by the cluster installation system nor managed by the cluster management system.

*Other names and brands may be claimed as the property of others*
Fat node
A node with a hardware superset relative to other nodes used for the same purpose, e.g., extra memory or additional secondary storage compared to a 'standard' configuration of that node type.

General Purpose (GP) coprocessor
A secondary, independent, and programmable processing unit that requires a cluster node to host it. The cluster node has its own general purpose processor and is responsible for the startup and management of the GP coprocessor. The GP coprocessor is capable of directly executing application binary code in parallel with the host CPU.

Hardware
General term that is used to describe physical examples of a specific technology, e.g., processors, chipsets, memory, boards, and systems.

Head node or master node
The administrative node of the cluster, including the cluster installation system and the cluster management system. It is typically the node for interactive user sessions, although this task may be fulfilled by a login node.

Identical files
Files with the same file size, permissions, and checksum.

Implementation-specific
A value or behavior that is not defined by this document, but must be selected by an implementer. The value or behavior may vary among implementations that conform to this document. An application writer that relies on such a value or behavior cannot be assured that the application will be portable across conforming implementations.

Ingredient
A hardware, software, or procedural component that can be combined together with other components to make something new, either another component or a complete system.

Int/Long/Pointer 32 bits (ILP32)
The C language programming model where C data types int, long, and pointers are all 32 bits in length.

Intel® Cluster Ready certified cluster design
A cluster design that passed the formal Intel® Cluster Ready certification process and has an associated certificate of compliance to this specification.

Intel® Cluster Ready deployment
A physical instantiation (installation) of an Intel® Cluster Ready certified cluster design that results in a system compliant with this specification.

IP address
Internet Protocol address, a computer network address expressed according to either IPv4 or IPv6 protocols.
Job execution data
For jobs submitted through a resource manager, job execution data consists of the user account information of the submitting user as well as the data describing the specific resources needed for execution.

Job requirements
The minimum hardware and software required to complete a task.

Login node
The primary point for user interaction with the cluster. Users start jobs from this node, either by submitting jobs to the resource manager or directly starting job execution, and stage input and output files. The function of the login node may be supplied by the head node.

Long Long/Pointer 64 bits (LLP64)
The C language programming model where C data type long long and pointers are 64 bits in length and a long is 32 bits in length.

Long/Pointer 64 bits (LP64)
The C language programming model where C data type long and pointers are 64 bits in length and an int is 32 bits in length.

Message layer independence
Message layer independence is the abstraction of the physical message fabric from the application program.

Node
An independent computer connected to the management network. It has a processor, memory and at least one network connection with a distinct MAC address. Each node must have access to storage, although that storage need not be direct access storage.

OpenFabrics* Enterprise Distribution (OFED*)
OFED is a stable version of the open-source OpenFabrics Alliance* software stack supporting connectivity using RDMA-based InfiniBand* and iWARP fabrics for the Linux* and unix environments.

Out-of-Band (OOB)
The ability to manage a node regardless of the state of the node’s primary power and regardless of the state of the node’s online operating system.

Persistent storage
Non-volatile storage that is unaltered after power-off or reboot of a node. It is capable of remaining unaltered after reprovisioning of a node or reinstallation of the cluster head node.

Platform
An integrated set of ingredients that enables targeted usage models and delivers greater end-user benefits than the sum of its parts. Platforms include 5 ingredients: Hardware, Software, Technologies, Services, and Standards & Initiatives.
Primary node
The single node where a job is instantiated upon its creation.

PXE
The Pre-boot Execution Environment, an environment to bootstrap computers using a network interface card independently of storage or the operating system.

Recipe
The documentation that includes both the list of ingredients and descriptions of the processes and procedures that combine those ingredients into an instance of a cluster.

Runtime
The set of libraries and machine state which provides software services for processes or programs.

Script
A file that is read and executed by a command interpreter.

Serial-over-LAN (SOL)
A feature that enables advanced remote management by providing the ability to redirect traditional console server serial data over a local LAN by using the available baseboard management controller (BMC.) The Serial-over-LAN feature also includes redirection of both BIOS and OS console to a remote client console and is designed to work with existing standard telnet consoles.

Service node
A service node provides non-computational resources to the cluster, is provisioned by the cluster installation system and managed by the cluster management system. Users do not typically access a service node directly, but instead through the service that the node supplies.

Services
A software system designed to support machine-to-machine or machine-to-human interaction either on the local system or over a network, e.g., digital media distribution.

Shared file space
A collection of files of any type that are placed in a persistent storage such that the path name is the same on all nodes and refers to a single instance of the file.

Shell
An application program that provides a user interface to the services of operating system kernel and associated application programs. Traditional command-line shells operate in an interactive mode, where commands are entered ad-hoc to a command-line prompt, and a non-interactive mode, where a script of shell commands (shell script) is run to completion. The execution of a shell command or a shell script should be indistinguishable from an application program. The term shell may also be applied to GUI desktop environments, such as KDE* and Gnome*.
Single System Image (SSI)
A form of clustered or distributed computing solution using a common interface to multiple networks, distributed databases, or servers that appears to the user as one computing system. In concept, the operating system environment is shared by all nodes in the system and the implementers strive to remove the physical barriers to make the logical distinctions between nodes in the clusters transparent to the user, the administrator, and the programmer.

Software
A program that enables a computer to perform a specific task. For example, software includes the OS, user-mode applications, tools, and firmware.

Sparing
The use of additional hardware components to ensure a certain level of service or performance is maintained. A spare (also called a secondary or backup) is a materially identical component that is left unused during normal operation. When a failure of the main or primary component is detected, the primary component is disabled and secondary or spare component takes over the function of the primary. This specification does not define how a component failure is detected, nor the processes and procedures needed by the system to disable the primary and activate the spare.

Standards and initiatives
A practice or set of documented requirements that is widely recognized or employed, e.g., Ethernet, PCI*, SATA, USB.

Sub-cluster
A group or subset of similar nodes within a cluster. The cluster installation software and cluster management software are common for all sub-clusters in a cluster, but the configuration and capabilities of each sub-cluster may vary. Sub-clusters enable a cluster to partition resources for specific uses while providing a single space for ease of operation, management, and/or administration.

Technologies
A general term referring to product features implemented in hardware, software, or both, e.g., Intel® Hyper-Threading Technology (Intel® HT Technology), Intel® Virtualization Technology (Intel® VT), Intel® I/O Acceleration Technology (Intel® I/OAT), and Intel® Active Management Technology (Intel® AMT).

Unix or UNIX*
Unix follows the general convention as the term used to define the family of operating systems that were originally based on technology developed at Bell Laboratories; and now formally defined by IEEE and Open Group with the POSIX* definition documents. This can be conceptually replaced by the reader for the more formal term UNIX* which is a registered trademark.

User account information
User account information includes, but is not limited to, the information maintained for each user by the local OS, as defined by POSIX*, e.g., the user’s login name, numerical user and group IDs, the initial working directory and home directory, and the program to use as a command shell.
x86_64

The Linux* processor architecture that describes the Intel® 64 Processor family or compatible processor.

Other terms and definitions used in this document shall have the same meaning as defined in Chapter 3 of the *Base Definitions* volume of the POSIX* (2008) standard.
4. **Base Requirements**

This section defines the basic framework that all conforming Intel® Cluster Ready clusters must provide. More details for some of the requirements may be found in other sections as appropriate.

In a non-SSI cluster environment a distinction is made between different types of nodes in the cluster. This section also identifies the roles and requirements for those different types of nodes.

4.1 **Base Cluster Requirements**

1. All deployed clusters shall be based on a certified cluster design.
2. A cluster is defined as containing one or more head or login nodes, multiple compute nodes grouped logically into sub-clusters, and optionally, service nodes.
   a. A cluster shall contain at least one head or login node.
   b. A cluster shall contain multiple compute nodes.
      i. All Intel® Cluster Ready design certifications must be performed on a cluster that contains at least four compute nodes per sub-cluster.
      ii. All Intel® Cluster Ready deployments shall contain a total of two or more compute nodes.
   c. Each node may provide multiple capabilities, e.g., a head node also used as a compute node.
      i. A node providing multiple capabilities shall satisfy all the requirements of each capability it provides.
      ii. A node providing multiple capabilities shall be counted once per capability for the purpose of satisfying the minimum number of nodes requirements in this section.
3. A cluster is defined to contain at least one sub-cluster.
   a. The cluster shall contain one or more sub-clusters.
   b. Enhanced capability nodes used in conjunction with other nodes in a sub-cluster are considered fat nodes within the sub-cluster.
   c. No more than three fat nodes are allowed per sub-cluster.
   d. Sub-clusters in an Intel® Cluster Ready certified design may be omitted entirely in a deployment, provided that all other requirements are met.

4.2 **Base Software Requirements**

1. The head node shall be the primary point of entry to the cluster and primary point of administrative control unless there are specific login or administrative nodes in the cluster.
   a. A head node shall run a conforming distribution, but is permitted to run a different distribution than its compute nodes, provided that the head node is not also a compute node.
   b. For head nodes used as compute nodes, the file tree of a head node shall be a superset of the file tree for a compute node.
2. Compute nodes provide the primary computational capabilities of the cluster.
a. Compute nodes shall present a materially identical operating environment to all users.
   i. The file tree of a compute node shall be materially identical to other compute nodes except for files that specify unique node or sub-cluster identification or configuration
   ii. All compute nodes shall run the same software image (kernel, libraries, and commands).
   iii. All provided system-wide software shall have a properly configured and consistent runtime environment; this includes a consistent version of the GNU* runtime, and a consistent MPI image (e.g., mpirun, mpiexec and libs).

b. While running on a compute node, a user’s job shall be able to access that user’s account information.
c. Users shall be able to access data stored locally on any compute node upon which they have a running job.

3. Clusters shall be built such that:
   a. the users’ home directory shall be maintained in a shared file space called /home;
   b. each network host name shall be consistently resolved to the same IP address on all nodes;
   c. each node shall have access to private persistent storage;
   d. privileged users shall be able to execute commands on all nodes;
   e. all nodes operate under a single authentication domain, i.e., a user authenticates once on a head or login node and is granted privileges to the cluster using that single login.

4. Intel® Cluster Checker version 2.2 or later shall be installed on all head and login nodes.
   a. The cluster shall successfully pass Intel® Cluster Checker in certification mode.
   b. If user jobs do not span sub-clusters, the sub-clusters may be checked individually.

5. Multiple application runtime environments must be able to coexist on the same cluster without adversely affecting each specific runtime. This includes multiple versions of the same runtime environment.

6. All system-wide software shall have its runtime environment configured by default. In case of conflicts between components or versions, it is left to the implementer to choose which runtime environment is the default. In all cases, a user shall be able to select the runtime environment required by an application.

7. Every node shall provide a text file "/etc/intel/icr", formatted using POSIX* shell variable assignment syntax.

---

3 The file trees provided by SSI clusters on its nodes are materially identical "by construction".
a. The file shall define the variable `CLUSTER_READY_VERSION` indicating the specification version or versions of the implementation.
   i. This variable shall represent the sequence of specification versions that applications and utilities may use to determine whether an implementation is conformant to the corresponding version of the specification.
   ii. The versions shall be separated by a colon (`:`).
   iii. A version shall be defined in the form `X.Y` or `X.Y.Z`.
   iv. This variable shall be a non-null value.
   v. The value pertaining to his version of the specification shall be `1.3.1`.

b. Future versions of the specification may require additional content in this file.

4.3 Base Hardware Requirements

Minimal hardware requirements are described to ensure functional clusters are built from this specification. This specification does not guarantee that specific implementations built only to these minimal requirements will offer maximal or optimal application performance. Cluster implementers must assume that additional hardware resources beyond this specification will be required to provide maximal or optimal application performance in specific cases.

1. Each node shall have at least:
   a. one Intel® 64 processor;
   b. a minimum of 8 gibibyte of random access memory or 0.5 gibibyte per processor core, whichever is greater;
   c. a connection to a management fabric that permits the functionality and requirements defined in sections 5. Pre-Boot Requirements and 6.5 Management Fabric.

2. Head nodes shall have at least 200 gibibytes of direct access local storage.

3. Compute nodes shall have access to at least 20 gibibytes of persistent storage per node. The storage shall be implemented as direct-access local to the node or available over a network.

4. Compute nodes within a sub-cluster shall use identical hardware, with the following allowed exceptions:
   a. Nominal, revision-level variation;
   b. Enhanced capability fat nodes; no more than 3 fat nodes are permitted per sub-cluster.
5. **Pre-Boot Requirements**

The node’s online OS instance shall not be required for any of the following capabilities. All of these capabilities shall be available to each node before the OS is provisioned, booted, and takes over operation of the node.

5.1 **Network Boot**

1. Nodes shall support download and initiation of arbitrary payloads from network servers, including but not limited to: diskless kernels, BIOS, or firmware updates.
2. A network download shall be initiated via each of the following:
   a. Power-on
   b. Hard reset
3. The arbitrary payload shall be downloaded using an IP-capable network fabric.
4. The arbitrary payload shall be downloaded using one or more of the following protocols:
   a. Trivial File Transfer Protocol (TFTP)
   b. Hypertext Transfer Protocol (HTTP)
   c. File Transfer Protocol (FTP)
   d. Secure Copy (SCP)

5. A node providing PXE boot shall be deemed sufficient.
6. Network download need only occur in a power-on state.

5.2 **Remote Access**

1. The console of each node in the cluster shall be available via a TCP/IP network connection.
   a. A single serial console stream shall be sufficient.
   b. Physical access to the console shall not be required. Serial-over-LAN capabilities shall be sufficient.

---

4 While SSH is well specified in the Internet RFC’s, the authors of this document do not know of a formal definition for SCP. It is based on the UC Berkeley’s rcp(1) program. The C programming language source code is widely available online which make up the definition of each program.

5 For low-cost/entry-level “personal clusters” with minimal node counts, where the usage model is extremely localized and the console information from each node can be demonstrated to be available to the user in a “reasonable manner,” Intel may consider an exception to this requirement for specific implementation families.
6. Cluster Functions

6.1 OS Kernel

The kernel supplies all user-mode access to kernel APIs needed to support other sub-systems. Platform integrators should note that some kernels supplied in off-the-shelf distributions may require patches to support some optional features, such as OFED*.

1. The kernel shall materially conform to the Linux* Standard Base.
2. The kernel version shall be 2.6.32 or later.
3. Any Linux* distribution based on kernel version 2.6.32 or later satisfies this requirement. The following kernel versions included with specific Linux* distributions, including those compiled from the same source code, are sufficient despite formally not having a version greater than 2.6.32.

<table>
<thead>
<tr>
<th>LINUX* DISTRIBUTION</th>
<th>KERNEL VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat* Enterprise Linux* 5 Update 6</td>
<td>kernel 2.6.18-238</td>
</tr>
<tr>
<td>Red Hat* Enterprise Linux* 5 Update 7</td>
<td>kernel 2.6.18-274</td>
</tr>
<tr>
<td>Red Hat* Enterprise Linux* 5 Update 8</td>
<td>kernel 2.6.18-308</td>
</tr>
<tr>
<td>Red Hat* Enterprise Linux* 5 Update 9</td>
<td>kernel 2.6.18-348</td>
</tr>
<tr>
<td>Red Hat* Enterprise Linux* 5 Update 10</td>
<td>kernel 2.6.18-371</td>
</tr>
</tbody>
</table>

6.2 OS Interface and Basic Runtime Environment

This section defines the required minimum runtime environment that each Intel® Cluster Ready implementation must provide. Some ISVs may require additional or newer versions of these runtimes for compatibility. Intel® Cluster Ready implementers need to allow multiple runtime versions to co-exist.

1. All APIs shall be defined using the LP64 programming model. Where explicitly required for compatibility with existing practice or current program binaries, APIs using the ILP32 programming model may also be provided.
2. The ABI behavior shall conform to the Intel® 64 architecture. Where explicitly required for compatibility with existing practice or current program binaries, the IA32 ABI may also be provided.
3. Implementations shall allow multiple versions of libraries to be installed and allow applications to choose the specific version they require.
4. A materially POSIX*-conformant OS runtime binding shall be provided on all nodes.
5. The following libraries that expose interfaces for accessing the operating system, processor, other hardware, and software subsystems shall be provided and recognized by the dynamic loader on all nodes for both the LP64 and ILP32 data models:

| libacl.so.1       | libncurses.so.5     |
| libattr.so.1      | libncursesw.so.5    |
| libbz2.so.1       | libpam.so.0         |
| libcap.so.2       | libz.so.1           |

6. Both LP64 and ILP32 versions of the following runtime libraries shall be provided on all nodes:

*Other names and brands may be claimed as the property of others.
a. ANSI* standard C/C++ language runtime of the GNU* C Compiler version 4.1 or later
b. ANSI* standard C/C++ language runtime of the Intel® C++ Compiler version 2013.1. or later
c. Standard Fortran language runtime of the Intel® Fortran Compiler version 2013.1 or later
d. Intel® Math Kernel Library version 11.01 or later
e. Intel® Threading Building Blocks version 4.1.1 or later

7. The LP64 version of the Intel® MPI Library Runtime Environment version 4.1 or later shall be provided on all nodes

8. The runtime libraries for the Intel tools shall be provided in the directory
   `/opt/intel/PROD/VERSION/
   a. The `VERSION` string shall be in the form: `X.Y` or `X.Y.Z`
   b. The `PROD` string shall correspond to:

<table>
<thead>
<tr>
<th>INTEL TOOL</th>
<th>PROD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® C++ Compiler (ILP32)</td>
<td>cc</td>
</tr>
<tr>
<td>Intel® C++ Compiler (LP64)</td>
<td>cce</td>
</tr>
<tr>
<td>Intel® Fortran Compiler (ILP32)</td>
<td>fc</td>
</tr>
<tr>
<td>Intel® Fortran Compiler (LP64)</td>
<td>fce</td>
</tr>
<tr>
<td>Intel® Math Kernel Library</td>
<td>cmkl</td>
</tr>
<tr>
<td>Intel® MPI Library</td>
<td>impi</td>
</tr>
<tr>
<td>Intel® Threading Building Blocks</td>
<td>tbb</td>
</tr>
</tbody>
</table>

c. Symbolic links are acceptable for these directories as paths may change.

9. Both LP64 and ILP32 versions of the following runtime libraries shall be provided and recognized by the dynamic loader on all nodes:

<table>
<thead>
<tr>
<th>lib- linux.so.2 (ILP32 only)</th>
<th>libnss_files.so.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>lib- linux-x86-64.so.2 (LP64 only)</td>
<td>libnss_hesiod.so.2</td>
</tr>
<tr>
<td>libBrokenLocale.so.1</td>
<td>libnss_idap.so.2</td>
</tr>
<tr>
<td>libSegFault.so</td>
<td>libnss_nis.so.2</td>
</tr>
<tr>
<td>libanl.so.1</td>
<td>libnss_nisplus.so.2</td>
</tr>
<tr>
<td>libc.so.6</td>
<td>libnuma.so.1</td>
</tr>
<tr>
<td>libCldn.so.1</td>
<td>libpanel.so.5</td>
</tr>
<tr>
<td>libcrypto.so.1</td>
<td>libpanelw.so.5</td>
</tr>
<tr>
<td>libcrypto.so.6*</td>
<td>libpthread.so.0</td>
</tr>
<tr>
<td>libdl.so.2</td>
<td>libresolv.so.2</td>
</tr>
<tr>
<td>libgcc_s.so.1</td>
<td>librt.so.1</td>
</tr>
<tr>
<td>libjpeg.so.62</td>
<td>libstdc++.so.5</td>
</tr>
<tr>
<td>libm.so.6</td>
<td>libstdc++.so.6</td>
</tr>
<tr>
<td>libnsl.so.1</td>
<td>libthread_db.so.1</td>
</tr>
<tr>
<td>libnss_compat.so.2</td>
<td>libutil.so.1</td>
</tr>
<tr>
<td>libnss_dns.so.2</td>
<td>libuuid.so.1</td>
</tr>
</tbody>
</table>

10. On each node in the cluster, an implementation of the Java* Runtime Environment (JRE) that conforms to OpenJDK version 6 runtimes or later shall be provided.

11. LP64 versions of the following X11 client runtime libraries shall be provided and recognized by the dynamic loader on all nodes:

6 libcrypto.so.0.9.8, revision a or b, is equivalent to libcrypto.so.6. A symbolic link is acceptable to fulfill this requirement. Symbolic links may also need to be made for other libraries distributed in the same package as libcrypto (e.g., libssl.so.6 -> libssl.so.0.9.8).
6.3 Command System and Tools

1. A complete and materially POSIX*-conformant command system shall be provided on all cluster nodes.

2. The Perl* scripting language version 5.8.8 or later shall be provided on all cluster nodes.

3. The Python* scripting language version 2.4.3 or later shall be provided on all cluster nodes.

4. The Tcl scripting language version 8.4.13 or later shall be provided on all cluster nodes.

5. The following X11R7.1 or later executable binaries shall be provided on the head node:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>glxinfo</td>
<td>xwininfo</td>
</tr>
<tr>
<td>xev</td>
<td>xvinfo</td>
</tr>
<tr>
<td>xkill</td>
<td>xterm</td>
</tr>
<tr>
<td>xauth</td>
<td></td>
</tr>
</tbody>
</table>

6. A web browser that supports HTML 4.01 shall be provided on the head node and enabled with a plugin implementation of the Java* Runtime Environment (JRE) that conforms to OpenJDK version 6 runtimes or later.

7. The head node shall provide a viewer for the Portable Document Format (PDF) and PostScript* (PS) documentation formats.

6.4 Message Passing Layer

1. All MPI implementations installed system-wide shall:
   a. conform to the MPI-2.0 standard.
   b. be network fabric independent.

2. The physical network fabric shall be hidden from an application and no fabric-specific programming shall be required from the application.

3. Regardless of whether the MPI library is statically or dynamically linked to an application, a binary shall run on any network fabric supported by the MPI implementation, requiring only runtime environment modifications to achieve fabric independence.

4. Multiple network fabric interfaces, drivers, and libraries shall coexist without conflicting

5. An MPI implementation must support at least two network fabrics, one of which shall be IP-based Ethernet.

*Other names and brands may be claimed as the property of others
6. Support Secure Shell (ssh) as an underlying transport for MPI startup.

7. Multiple MPI implementations shall coexist without conflicting. A user shall be able to select the MPI runtime environment required by an application.

6.5 Management Fabric
1. The management fabric shall support the requirements defined by Section 5.
2. The management fabric shall be accessible using the standard IP network stack after OS boot is complete.

6.6 Message Fabric
1. The message fabric shall be accessible using at least the standard IP network stack post OS boot.
2. Message fabrics that are not based on Ethernet shall provide a “uDAPL provider interface” compliant with version 2.0 or later.

6.7 Remote Execution Environment
1. All clusters shall provide an SSH implementation, or a functional replacement, on all nodes that supports at least the following use cases:
   a. `ssh host cmd` executes the shell command, `cmd`, on the node, `host`, and redirects output to stdout and stderr.
   b. `ssh user@host cmd` executes the shell command, `cmd`, on the node, `host`, as user, `user`, and otherwise behaves identically to the preceding case.
   c. `ssh -t host cmd` allocates a pseudo-tty on the node, `host`, and otherwise behaves identically to the preceding cases.
   d. `ssh -X host cmd` enables X11 forwarding between the local node and the node, `host`, and otherwise behaves identically to the preceding cases.
   e. `scp localfile host:remotefile` copies the file, `localfile`, from the local node to a file named `remotefile` on node `host`. `localfile` and `remotefile` may be relative or absolute paths.
   f. `scp host:remotefile localfile` copies the file, `remotefile`, from the node `host` to a file named `localfile` on the local node. `localfile` and `remotefile` may be relative or absolute paths.
2. All nodes shall be accessible via SSH, or a functional replacement, over the management network.
3. All compute nodes shall be accessible via SSH, or a functional replacement, over the message network.

6.8 Storage and File System Interface
1. Each node shall have access to private, persistent storage.
2. The cluster shall provide a single, shared file space for users' home directories.
3. The cluster file system shall provide the same file semantics as the local file system.
4. The cluster file system shall provide at least the consistency and access guarantees provided by NFS Version 4.0.
6.9 Cluster Provisioning System

1. The cluster provisioning system shall provide:
   a. an automated method to provision the compute node images.
   b. automated functionality to expand an existing cluster.
   c. automated functionality to contract an existing cluster.

2. The cluster provisioning system may require manual interaction to provision or modify the size of a cluster; the method shall not employ manual provisioning and configuration of the compute node images.

3. The method of adding or removing compute nodes shall update the required services and files on all nodes. Consequently the cluster will operate as if it had been initially constructed of the new size. This functionality must include updates on all nodes for the following:
   a. Network address resolution
   b. User access
   c. Cluster-wide execution services
   d. Resource management
7. **Application Requirements**

Requirements and advisories applicable to applications may also be found in other sections, e.g., Section 4.2.

7.1 **Installation**

1. An application shall completely install in `/opt/`, `/usr/local/`, or in some unique location in the shared file space.\(^7\)
   a. Configuration files may be installed in `/etc/` or `/usr/local/etc/` as required, but shall be named to distinguish the application.
   b. If the suggested LSB\(^*\) paths of `/opt/` and `/usr/local/` are not in the shared file space, and are used by the application, the application must ensure that all nodes contain identical data.
   c. Unix- and POSIX\(^*\)-based applications are permitted to treat `/usr/opt` the same way LSB\(^*\) defines `/opt`.

2. Except for the specific case of files in `/etc/` or `/usr/local/etc/`, all runtime components provided with an application shall be installed in the same installation tree as the application and not in general use locations. In particular, an application providing private versions of 3rd-party files shall install those files within its private installation tree.

3. If an application is not installed using a standard installer (e.g., `rpm`, `apt`) and is attempting to overwrite files already on the system, the installer shall have the following behaviors:
   a. Non-interactive installers by default shall log the error and not replace the existing files. It is permissible to override this logging behavior with a negation option.
   b. Interactive installers by default shall prompt whether to overwrite the existing files. It is permissible to override the replacement behavior with a different parameter or negation option.

---

\(7\) Note: if the shared file space is used, then the application provider shall ensure that the name used is unique.
8. **Optional Features**

These features are suggested but not required. If an optional feature listed in this section is provided, it shall adhere to the requirements specified below.

8.1 **Pre-Boot Requirements**

All nodes in the cluster should have Out-of-Band (OOB) capabilities. If OOB capabilities are provided, the following requirements must be met.

1. OOB controls shall include all of the following:
   a. Power-on
   b. Power-off
   c. Hard reset
2. All nodes shall be available on an IP based network, although the network is not required to be physically or logically distinct from other management network traffic.
3. All nodes shall be selectable by one or more of the following:
   a. IP address
   b. MAC address
4. OOB controls shall function regardless of power state.
5. OOB controls shall function regardless of the OS or its state.

8.2 **Remote Execution Environment**

A cluster should have a secure, scalable method for executing commands and synchronizing files on all nodes or any subnet of nodes. If this capability is provided, the following requirements must be met.

1. The scalable execution method shall report all execution failures on all failing nodes.
2. The scalable execution method shall provide access to the C language file streams, stdin, stdout, and stderr on all nodes.
3. The scalable synchronization method shall report all operation failures on all failing nodes.

8.3 **Cluster-Wide Resource Manager**

The resource manager is responsible for running jobs across the cluster. If a resource manager is provided, the following requirements must be met. A list of resource managers known to meet these requirements can be found at [http://www.intel.com/go/cluster](http://www.intel.com/go/cluster).

1. Upon startup, a job shall be instantiated on its primary node.
2. The names of the allocated nodes shall be made available to the job via an environment variable or a file.
3. The resource manager shall start a job only if sufficient resources are available to satisfy all resource requirements known at job launch.
4. The resource manager shall return an error to the scheduler (see Section 8.4) if it is unable to start a job for any reason.
5. Upon job startup, the resource manager shall propagate one of the following user’s environments to the job’s primary node:
a. The environment variables and values at the time of job submittal.
b. The environment variables and values at normal login.

6. During execution, the resource manager may make the stdout and stderr streams from the primary node of the job available to the submitting node.

7. Upon completion or termination of a job, the resource manager shall return the stdout and stderr streams from the primary node of the job to the submitting node.

8. Job execution data shall include, but not be limited to
   • the user account information of the submitting user
   • job execution status (e.g., queued, running, exited)
   • time stamps of changes to status state (e.g., created, queued, started)
   • nodes needed or used
   • priority
   • total amount of CPU time allocated or consumed
   • the amount of wall clock time allocated or consumed

9. Restarting the resource manager after a crash shall not interrupt currently running jobs.

10. For Beowulf-style clusters, the resource manager should use authenticated and encrypted methods to communicate with compute nodes.

11. If a resource manager is provided on the cluster, interactive commands shall be executable on the head node.

12. Privileged users shall be able to perform any function that an unprivileged user can perform, as well as:
   a. add and delete nodes from the pool of available system resources.
   b. establish resource limits on a per-node basis, including but not limited to the number of “allocatable” CPUs per node, maximum “allocatable” memory, number of jobs, etc.
   c. establish resource limits on a per-queue basis, including but not limited to the number of “allocatable” ranks per queue, maximum “allocatable” memory, number of jobs, etc.
   d. view the status of any queued, running, or recently completed job.
   e. delete any queued job.
   f. alter the priority of any queued job.
   g. terminate any running job.
   h. restart any running job.
   i. preempt any job only if checkpoint-restart is available.

13. An unprivileged user shall be able to:
   a. communicate job requirements via the job submission command or the job script.
   b. view the status of any queued, running, or recently completed job owned by the user.
   c. hold any queued job owned by the user and subsequently release that hold.
   d. delete any queued job owned by the user.
   e. restart any running job owned by the user.
   f. terminate any running job owned by the user.
   g. access a node only if the node is assigned to a currently running job owned by the user.
   h. preempt any job owned by the user only if checkpoint-restart is available.

14. An unprivileged user shall not be able to:
a. hold any queued job not owned by the user or release any held job not owned by the user;
b. delete any queued job not owned by the user.
c. terminate any running job not owned by the user.
d. restart any running job not owned by the user.
e. release a job held by a privileged user, even if it is this user's job.

8.4 Cluster Wide Job Scheduler
The scheduler is responsible for the order in which jobs run on the cluster. If a scheduler is provided, the following requirements must be met.

1. The resource manager shall communicate user-specified job requirements to the scheduler.

2. The scheduler shall support at least the following scheduling policies:
   a. First-In, First-Out (FIFO)
   b. Priority
   c. Backfill

3. Priority scheduling shall support prioritization by queue, user name, user group membership, job "application," and resource requirements.

4. The scheduler shall permit arbitrary limits to the number of currently running jobs, number of queued jobs, and queue priority on a per-queue basis.

5. Restarting the scheduler after a crash shall not interrupt currently running jobs.

6. There shall be a way for both privileged and non-privileged users to communicate with the scheduler. This interaction may be direct or via requests from the resource manager.

7. Privileged users shall be able to:
   a. add and delete nodes from the pool of schedulable system resources.
   b. view the requirements of any queued job.
   c. change the requirements of any queued job.
   d. define policies to enable, disable, and control the scheduler's use of job preemption if the resource manager supports job preemption.

8. An unprivileged user shall be able to:
   a. view the requirements of any queued job owned by the user.
   b. change the requirements of any queued job owned by the user.

9. An unprivileged user shall not be able to:
   a. view the requirements of any queued job not owned by the user.
   b. change the requirements of any queued job not owned by the user.

8.5 General purpose coprocessor
A general purpose (GP) coprocessor operates independently from the CPU on a host platform, and provides the ability to execute arbitrary binary application code in parallel with the host platform CPU. It supports one or more models of operation, or a hybrid of these:
   • Offload execution: execution of subroutines on the GP coprocessor, with the main binary executed on the host CPU.
   • Native execution: the main binary is executed directly on the GP coprocessor.

1. The host platform containing the GP coprocessor shall be compliant with this specification.
2. A device with the ability to run applications without an active, running operating system on the host platform does not fall under the definition of GP coprocessor.

3. The GP coprocessor shall provide a worldwide unique identifier. A unique MAC address on the device will satisfy this requirement.

4. A GCC-compatible API shall be provided on the host platform, and shall include the following functions at a minimum:
   a. Read the worldwide unique identifier
   b. Terminate processes running on the GP coprocessor
   c. View total GP coprocessor and memory utilization on the device
   d. View device power state and readiness status
   e. Activate, disable, and reset GP coprocessor devices

5. Operating software for the coprocessor device shall:
   a. Support assignment of code execution to specific GP coprocessor devices within a host platform.
   b. Be installable on host platform without GP coprocessor devices present.
   c. Be installable and executable alongside software for other GP coprocessor devices without interfering with the operation of other devices.
   d. Meet all requirements defined in Section 7 Application Requirements.

6. If the GP coprocessor implements users and groups, it shall:
   a. Use the same authentication domain as the cluster.
   b. Require no password when accessing the device with the same user account.
   c. Provide the ability to limit access to the GP coprocessor by user and group.
   d. Provide the ability for users to view status and terminate running processes on the GP coprocessor that they own.
9. **Advisories**

This section is not part of the formal specification. It is intended to help implementers understand the intentions of the authors and anticipate future modifications to the specification.

9.1 **Base Cluster Requirements – advisory statement**

1. The head node functionality may be provided by a single machine or more than one in a redundant configuration.
2. Head nodes should be able to access all cluster networks.
3. The head and login node may be accessible to other networks.
4. Unprivileged users may not be able to access a compute node unless they have been allocated resources on that node.
5. Clusters may include a head node or one or more compute nodes which are implemented as virtual machines.
    a. Virtualized nodes must implement all requirements in this specification, i.e. each virtual node must present a materially-identical operating environment to users and use the same software image as other compute nodes. There are no requirements for the host platform or for the hypervisor, provided that the host platform itself is neither a head or compute node.
    b. Virtual nodes must be implemented as subclusters if used in the same cluster with physical nodes.
    c. If more than one virtual node is implemented on the same physical hardware, enough capacity must be provided to comply with requirements for memory and persistent storage. It is recommended that all virtual storage be fully allocated when the virtual node is created.

9.2 **Base Software Requirements – advisory statement**

1. The hardware and software components of all nodes in a sub-cluster should be exactly identical. Files that specify unique identification or configuration of the node may be different as needed.
2. File differences between nodes in a sub-cluster should be as minimal as possible. Practices that force differences between nodes, such as including inline meta-data is discouraged. Such practices increase the difficulty in verifying correct cluster operation and adherence to the specification.
3. When a user selects the runtime environment required by an application, the following capabilities should be provided:
   a. All applications and tools that require a custom environment should provide a single method, such as a command script, that clears any previous runtime environment and sets the new environment.
   b. Naming conventions for these methods are currently undefined. However, future versions of this specification may enforce a specific naming convention.
4. All performance thresholds appropriate to the cluster should be defined in the Intel® Cluster Checker configuration file.
5. The path `/etc/intel/clck/config.xml` should be used for the Intel® Cluster Checker configuration file.
6. The path `/var/log/intel/clck` should be used for Intel® Cluster Checker output logs.
7. Application programmers should search CLUSTER_READY_VERSION from beginning to end (left to right) until a match is found.

9.3 Base Hardware Requirements – advisory statement
1. Each node should have:
   a. a minimum of 16 gibibytes of random access memory per installed processor. Some applications may have higher requirements.
   b. a network fabric supporting uDAPL version 2.0 or later.
2. The message fabric should be implemented using “cut through” network technology.
3. The message fabric should be physically distinct from all other fabrics.
4. The storage fabric should be physically distinct from all other fabrics.
5. Each compute node should have access to at least 100 gibibytes of private, persistent storage.
6. All nodes should implement extensions in the processor and chipset for hardware-assisted virtualization. This should include support for processor virtualization extensions (VT-x), IOMMU virtualization (VT-d), and PCI-SIG* I/O virtualization (IOV).
7. Base hardware requirements and recommendations apply to nodes; they do not apply to add-in cards or general purpose coprocessor devices.

9.4 OS Interface, Basic Runtime Environment – advisory statement
1. Both LP64 and ILP32 versions of the following runtime libraries should be provided on all nodes:
   a. Standard Fortran language runtime of the GNU* gfortran compiler version 4.7 or later.
   b. Standard C and C++ language runtimes of the GNU* gcc and g++ compilers version 4.7 or later.
   c. The libreadline.so.4, libreadline.so.5, and libreadline.so.6 libraries should be provided on all nodes of the cluster.
2. Standard GNU* C compiler (gcc) version 4.7 or later should be provided on head nodes.
3. The latest available versions of the Intel runtimes should be provided on all nodes of the cluster.

9.5 Cluster Boot and Provisioning – advisory statement
1. A head node should be able to boot from direct access storage or over the management fabric.
2. Compute nodes should be able to boot over the management or message fabric.

9.6 Command System and Tools – advisory statement
1. A complete, materially conformant LSB* command system should be provided on all cluster nodes.
2. All nodes should be provided with the following shell command interpreters. It is permissible for implementations that are a true superset of another to provide more than one command interpreter implementation, e.g., a Korn Shell implementation that is a true superset of the POSIX* Shell or a TENEX Shell being a super set of the UC Berkeley C Shell. These implementations may be linked together.
a. POSIX* shell (/bin/sh)
b. Korn Shell (/bin/ksh)
c. GNU* Bourne-Again Shell (/bin/bash)
d. UC Berkeley* C Shell (/bin/csh)
e. TENEX C Shell (/bin/tcsh)

3. The Perl* scripting language version 5.18 or later should be provided on all cluster nodes.
4. The Python* scripting language version 3.3.4 or later should be provided on all cluster nodes.
5. X11/R7.7 clients or later should be provided on all compute nodes.

9.7 Message Passing Layer – advisory statement
1. The message passing library should conform to the MPI-2.2 specification.
2. The MPI profiling interface should be supported.
3. The message passing library should be thread safe and provide at least MPI_THREAD_FUNNELED thread support.
4. An MPI implementation should provide a single method, such as a command script, that clears any previous MPI runtime environment and sets the new environment.
5. For Beowulf style clusters, the default underlying transport for MPI start-up should be SSH.
6. Although SSI-based clusters may not need SSH for routine operations, ISV applications typically rely on SSH utilities for program startup. A functional SSH replacement on SSI-based clusters should present itself to the application with the same interface and behavior as the actual SSH utilities.
7. The MPI start-up command should be compatible with cluster-wide resource management.

9.8 Message Fabric – advisory statement
1. Implementations using InfiniBand* or iWARP should use OFED* release 3.5 or later.
2. The test for uDAPL conformance shall be defined as a successful completion of:
   a. RDMA Consortium’s dapijkstra
   b. Intel® MPI Benchmarks over the Intel® MPI Library
   c. Intel® MPI Library test suite.
3. For OFED*-based implementations, the message fabric software interfaces shall be defined by the OFED* implementation, including but not limited to:
   a. TCP/IP
   b. SDP
   c. uDAPL
   d. OpenFabrics* Verbs API

9.9 Remote Execution Environment – advisory statement
1. The cluster-wide execution method should report every instance where a remote command does not complete successfully.
2. The cluster-wide file synchronization method should report every instance where a remote file operation does not complete successfully.
3. OpenSSH version 5.3 or later should be provided.
4. The **sftp** command should be provided over the management and message networks.

9.10 Storage and File System Interface – advisory statement

1. The cluster file system should provide transparent capabilities matching the local file system with regards to:
   a. access
   b. data consistency
   c. file and record locking

2. The cluster file system should allow an SSI to use a single common file namespace for all system files on all nodes.

3. The cluster file system should allow all storage objects to be named and accessed anywhere in the cluster.

4. Standard file operations should be supported by the cluster file system including:
   a. traditional files and directories
   b. device files
   c. pseudo files (/proc)
   d. UNIX* named pipes
   e. Berkeley* named sockets
   f. Symbolic links

9.11 Cluster Installation System – advisory statement

1. Methods that change some subset of nodes in the cluster should not require a reboot of any other node in the cluster.

9.12 Cluster-Wide Resource Manager – advisory statement

1. The resource manager should provide:
   a. record of actual system resource consumed by each job;
   b. quota management system with soft and hard quotas.

2. Job execution data should also include the execution status (*i.e.*, POSIX* exit status), a time stamp for when the job became eligible to run, and disk and memory resources needed or used.

3. The resource manager should be highly available with at least active-passive sparing configuration.

4. The resource manager should support job pre-emption via checkpoint-restart.

5. If a quota management system is provided, it shall support sending a signal to any job exceeding either a soft or hard quota.
   a. After delivery of a hard quota signal, the job shall be terminated within some publicly available pre-determined interval.

9.13 Cluster-Wide Job Scheduler – advisory statement

1. The scheduler should be able to employ pre-emption to manage service level agreements if the resource manager supports job pre-emption.
9.14 Environment Variables – advisory statement

Note to implementers: items 1 and 2 in this section are expected to be required in future versions of this specification.

1. Applications should provide application-specific environment initializations scripts for inclusion within /etc/profile.d; both "sh" and "csh" versions shall be provided for use by the POSIX* and C-Shell compatible shells, respectively.

2. Applications should not rely on the user to manually establish a working environment, e.g., via a script that is manually run at login time.

3. Applications that rely on the user to manually establish a working environment, e.g., via a script that is manually run, must ensure the script is idempotent, that is, repeated execution results in the identical result.

4. Cluster provisioning software should ensure that application-specific environment initializations scripts are uniformly available on the head node and all compute nodes unless role-specific variants are provided by an application.

5. Applications that permit a small volume and low bandwidth temporary directory to be set via the environment should use the POSIX* “TMPDIR” environment variable.

6. Applications that permit a large volume and/or high bandwidth temporary directory to be set via the environment should use the “SCRATCHDIR” environment variable.

9.15 Application Requirements – advisory statement

Advisories applicable to applications may also be found in other sections, e.g., Section 9.2.

1. During installation or uninstallation, applications should generate a text log file of all installed (created), uninstalled (deleted), or modified files.

2. All system configuration files outside the application’s private installation tree should be backed up prior to modification.

3. Applications should read the contents of the file /etc/intel/icr to obtain versioning- and implementation-specific information.

9.16 Management, Monitoring, and Performance – advisory statement

Future Intel® Cluster Ready specifications may include management, monitoring, and performance requirements, such as in-band and out-of-band tools. Ganglia, System Tap, and Perfmon are specific examples of such tools.

9.17 General purpose coprocessors – advisory statement

1. A device used for offload execution of non-arbitrary tasks, such as encryption or network packet generation, is not considered a general purpose (GP) coprocessor for purposes of this specification. A general purpose coprocessor must have some capability for integer and floating-point general programming.

   Examples of these include:
   - Intel® Many Integrated Core (MIC) Architecture coprocessors
   - General purpose graphic processing units (GPGPUs)
   - High performance computing add-in accelerators

2. Devices that are capable of independent execution of arbitrary code are not GP coprocessors if they:
a. Are stand-alone
b. Require a host platform, but do not require the host platform to be operational
c. Use a host platform that is not independent, programmable, and capable of executing arbitrary code.

In these cases, the device shall be considered a cluster node and must satisfy all other requirements in this specification.
10. Assumptions

10.1 Relationship to ISO/IEC 9945 POSIX*

This specification includes interfaces and tools described in ISO* POSIX* (2008). Unless otherwise specified, these should behave exactly as described in that specification. Any conflict between the requirements described here and the ISO* POSIX* (2008) standard is unintentional, except as explicitly noted otherwise.

10.2 Relationship to Linux* Standard Base

This specification includes interfaces and tools described in Linux* Standard Base (LSB*) v4.1. Unless otherwise specified, these should behave exactly as described in that specification. Any conflict between the requirements described here and the LSB* v4.0 standard is unintentional, except as explicitly noted otherwise.

10.3 Licensing and Activation

Each node on the cluster contains the proper licenses required to run all software. Tools and runtimes that require activation may be provided on the cluster, but may not be used without activation. The user of those tools or runtimes is responsible for obtaining licenses.

Note that all of the runtimes currently specified in this document contain zero cost licenses and it is not the intention of this specification to require any specific activation or license fee. However, some of the tools and application-level programs may require specific licenses from their distributors.
11. References

11.1 Normative References

The following documents are helpful for understanding this specification.

An attempt has been made to state where Intel® Cluster Ready diverges from existing norms. These norms are the basic standards upon which Intel® Cluster Ready is built. The reader should consider this specification as the first standard of reference.

<table>
<thead>
<tr>
<th>TITLE</th>
<th>KNOWN URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® 64 Architecture</td>
<td><a href="http://developer.intel.com/technology/intel64/index.htm">http://developer.intel.com/technology/intel64/index.htm</a></td>
</tr>
<tr>
<td>Filesystem Hierarchy Standard (FHS) 2.3</td>
<td><a href="http://www.pathname.com/fhs/">http://www.pathname.com/fhs/</a></td>
</tr>
<tr>
<td>FILE TRANSFER PROTOCOL (FTP)</td>
<td><a href="http://www.ietf.org/rfc/rfc0959.txt">http://www.ietf.org/rfc/rfc0959.txt</a></td>
</tr>
<tr>
<td>Hypertext Transfer Protocol (HTTP v1.1)</td>
<td><a href="http://www.ietf.org/rfc/rfc2616.txt">http://www.ietf.org/rfc/rfc2616.txt</a></td>
</tr>
<tr>
<td>Message Passing Interface (MPI)</td>
<td><a href="http://www.mpi-forum.org/docs/docs.html">http://www.mpi-forum.org/docs/docs.html</a></td>
</tr>
<tr>
<td>Secure Shell (ssh)</td>
<td><a href="http://www.ietf.org/rfc/rfc4251.txt">http://www.ietf.org/rfc/rfc4251.txt</a></td>
</tr>
</tbody>
</table>

*URLs are ephemeral and may be updated and changed independently to this publication.*
11.2 Bibliographic References

Hennessy 1994


Pfister 1997


Popek 1986

12. Change History

This section contains a summary of the substantive changes to this specification. Changes to the advisory sections, clarifications, formatting, section placement, and other minor changes are not included in this summary unless they result in a material change. The specification documents themselves are the first standard of reference.

12.1 Version 1.0 (May 24, 2007)
Initial version.

12.2 Version 1.0.1 (June 28, 2007)
No substantive changes.

12.3 Version 1.0.2 (October 23, 2007)

1. OS Interface and Basic Runtime Environment
   • Dropped requirement for libttf.so.2, libXft.so.1, and libXxf86rush.so.1.
     Rationale: Work with LSB* and ISVs showed that none of these libraries were used and/or their functionality was supplanted by other libraries.

2. Command System and Tools
   • Dropped requirement for beforelight, ico, xcalc, and xgc.
     Rationale: Work with LSB* and ISVs showed that none of these clients were required by known ISV application programs.

12.4 Version 1.1 (June 27, 2008)

1. Base hardware requirements
   • Added the concepts of sub-clusters and fat nodes.

12.5 Version 1.2 (March 24, 2011)

1. Base hardware requirements
   • Redefined the minimum cluster size when deploying a certified cluster design
     Rationale: Provided a cluster design is certified (i.e., tested) at a minimum cluster size, instances may be confidently deployed at lesser size.

2. Base Software Requirements
   • Clarified runtime environment requirements.
   • Changed Intel® Cluster Checker version to 1.7 or later.
     Rationale: The changes in this version of the specification require a revised tool to verify compliance.
   • Changed the definition of CLUSTER_READY_VERSION to allow implementations to comply with multiple specification versions.
     Rationale: An implementation may be compliant with multiple versions of the specification and should be able to indicate that to application software.

3. Remote Execution Environment
   • Allowed functional SSH replacements.
     Rationale: Work with ISVs and cluster implementers showed that functional replacements, such as the remote shell used in SSI clusters, were sufficient.

4. Development Clusters and Tools
5. **Message Fabric**
   - Changed uDAPL version to 2.0 or later.
   - Rationale: Reflect the evolution of DAPL.

6. **OS Interface and Basic Runtime Environment**
   - Added requirement for `libstdc++.so.5`.
   - Rationale: Work with ISVs showed that this library was still commonly used.
   - Added requirement for `libjpeg.so.62, libpanel.so.5, libselinux.so.1, libsepol.so.1, libdrm.so.2, libGL.so.1, libSM.so.6, libXau.so.6, and libXdmcp.so.6`.
   - Rationale: Work with ISVs showed that these libraries were commonly used.
   - Changed Intel® C++ Compiler runtime version to 11.0 or later, Intel® Fortran Compiler runtime version to 11.0 or later, Intel® Math Kernel Library runtime version to 10.1 or later, Intel® MPI Library Runtime Environment version to 3.2 or later, and Intel® Threading Building Blocks version to 2.1 or later.
   - Rationale: Updated the Intel® MPI Library version based on support for DAPL 2.0 and the other versions based on the tools bundled with the Intel® MPI Library version in the Intel® Cluster Toolkit.
   - Dropped requirement for `libtermcap.so.2`.
   - Rationale: Work with LSB* and ISVs showed that this library was not used and was no longer provided by Linux* distributions.
   - Dropped requirement for `libelf.so.0, libgdm.so.2, libGL.so.6, libGLw.so.1, libMrm.so.3, libOSMesa.so.4, libUil.so.3, libXevie.so.1, libXm.so.3, and libXTrap.so.6`.
   - Rationale: Work with LSB* and ISVs showed that these libraries were not used and were no longer provided by Linux distributions.

7. **Command System and Tools**
   - Dropped requirement for `listres, mkhtmlindex, showfont, and viewres`.
   - Rationale: Work with LSB* and ISVs showed these programs were not used and were no longer provided by Linux distributions.

8. **Compute Node File Trees**
   - Removed this section.
   - Rationale: This guidance is embedded in Intel® Cluster Checker.

9. **Message Passing Layer**
   - Changed the version of the MPI Standard to 1.3 or later
   - Rationale: Reflect the evolution of the MPI standard.

### 12.6 Version 1.3 (July 26, 2013)

1. **Base Software Requirements**
   - Changed Intel® Cluster Checker version to 2.1 or later.
   - Rationale: The changes in this version of the specification require a revised tool to verify compliance.

2. **Base Hardware Requirements**
   - Changed minimum memory requirements to 8 gibibits.
   - Rationale: changed to reflect increased memory requirements for many applications and current memory technologies.
   - Changed head node storage requirement to 200 gibibytes and compute node storage to 20 gibibytes.

*Other names and brands may be claimed as the property of others.*
Rationale: changed to reflect increased storage requirements for OS and applications and current storage availability.

3. OS Kernel
   • Changed minimum Linux* kernel to 2.6.32.
     Rationale: This is a minimum kernel required to support many new hardware technologies, and was a standard kernel for many Linux distributions as of December, 2010. Exceptions were added to reflect inclusion of new patches by Red Hat in an older kernel.

4. OS Interface and Basic Runtime Environment
   • Updated version of libcap.so to 2.
     Rationale: Work with Linux* distributions showed that this library was current for most and readily available for others.
   • Added requirement for libcursesw.so.5 and libpanelw.so.5
     Rationale: Work with ISVs showed that these libraries were commonly used and were normally installed with their required non-wide counterparts.
   • Added requirement for libnss_ldap.so.2
     Rationale: Add LDAP to match security libraries for other authentication already required in Specification 1.2.
   • Removed requirement for libselinux.so.1, libsepol.so.1.
     Rationale: Work with ISVs showed that these libraries were not used and were not available on some Linux* distributions.
   • Changed Intel® C++ Compiler runtime version to 2013.1 or later, Intel® Fortran Compiler runtime version to 2013.1 or later, Intel® Math Kernel Library runtime version to 11.01 or later, Intel® MPI Library Runtime Environment version to 4.1.0 or later, and Intel® Threading Building Blocks version to 4.1.1 or later.
     Rationale: All tools were updated to support development of applications for new technologies available as of December, 2011.
   • Added requirements for libcrypto.so.6, libnuma.so.1, libuuid.so.1, and libGLU.so.1
     Rationale: Work with ISVs showed that these libraries were commonly used by registered applications.
   • Sun* Java* JRE version 1.4.2 updated to OpenJDK version 6 runtimes.
     Rationale: Move of primary development effort for Java* moved from Sun Microsystems to .open source project. Compliant runtime environments are available from a number of sources. Version 6 represents a commonly used version.

5. Command System and Tools
   • Perl* version updated to 5.8.8, Python* version updated to 2.4.3, Tcl version updated to 8.4.13, and X11 version updated to R7.1.
     Rationale: These versions were in common use as of December 2010. This updates requirements to reflect current versions, while maintaining reasonable backward-compatibility.
   • Browser plugin for Sun* Java* version 1.4.2 updated to OpenJDK version 6; requirement for support of HTML 4.0.1 explicitly stated.
     Rationale: Move of primary development effort for Java* moved from Sun Microsystems to .open source project. Compliant runtime environments are available from a number of sources. Version 6 represents a commonly used version. Explicit HTML requirement prepares future specification updates to require newer HTML version support.
   • Added requirement for xauth.

6. Message Passing Layer
   • Changed the version of the MPI Standard to 2.0 or later
     Rationale: Reflect the evolution of the MPI standard.
7. Storage and File System Interface
   - The consistency and access guarantees minimum NFS version changed from 3 to 4.0.
     Rationale: Reflect update storage requirements and expected features implemented in NFS v4.

8. General Purpose Coprocessors
   - A new section covering requirements for add-in coprocessors, accelerators, and GPGPUs was added to define requirements for these devices.
     Rationale: With the growth of GPGPUs, accelerators, and coprocessors, it is necessary to address requirements in order to maintain the Intel Cluster Ready promise. It is important to consider, from a vendor-neutral standpoint, that registered applications and resource managers now optionally utilize these devices.

12.7 Version 1.3.1 (June 20, 2014)

1. Base Software Requirements
   - Changed Intel® Cluster Checker version to 2.2 or later.
     Rationale: The changes in this version of the specification require a revised tool to verify compliance.

2. OS Kernel
   - Added exception for Red Hat® Enterprise Linux® 5.10.
     Rationale: Exceptions were updated to reflect inclusion of new patches by Red Hat in an older kernel.

3. OS Interface and Basic Runtime Environment
   - Removed requirement for libI810XvMC.so.1.
     Rationale: Work with ISVs showed that this library was not used and was not available on some Linux* distributions.
   - Removed requirement for ILP32 version of Intel® MPI Library runtime environment.
     Rationale: Work with ISVs showed that the ILP32 version was not used and may not be available, depending on the Intel® MPI Library version used.

4. Command System and Tools
   - Removed requirements for xfontsel, xload, xfd, x11perfcomp, xmmessage, xclipboard, and x11perf.
     Rationale: Work with ISVs showed that these tools were not used and were not available on some Linux* distributions.