LLNL-TM-7390608

CORAL-2 BUILD STATEMENT OF WORK

April 19, 2018

CORAL: Collaboration of Oak Ridge, Argonne and Livermore National Laboratories

Department of Energy

Office of Science and

National Nuclear Security Administration

RFP No. 6400015092

Changes List

Changes from version 14 & 16 (version 15 was not published)

Section(s)	Description
Section 1	Changed to reflect that ORNL has the flexibility to choose a system proposed for delivery in 2022 and accepted in 2023 should programmatic changes occur
Section 3.2	Reduced aggregate system memory target from 10 PB down to 8 PB
Section 4.0	Benchmarks section completely rewritten – reducing the number of apps required for FOM analysis to four scalable science, four throughput, and two data science.
Sections 5.1.10 and 5.1.11	Both are new TR-2 requirements defining preferences when compute nodes contain both CPUs and accelerators
Section 6.0	Introduction revised to more clearly define why ORNL and LLNL I/O requirements are different and why these requirements are in different sections (6.1 and 6.2)
Section 6.2.4.1.4	Fixed typo in LLNL I/O system requirement, “The I/O subsystem will support 10 concurrent instantiations of the metadata performance tests specified in Section 6.2.4.1.3 without any degradation in performance."
Section 10.1.5	Entirely removed requirement: “The Offeror will provide a facility with a single point of control to analyze and to tune full system performance”
Section 12	Removed the height requirement from all three sites
Section 12.5.3	Modified to replace the one connection per rack with “Offeror will provide a solution that minimizes the number of connections to each rack”
Section 12.8	Clarified that the non-conductive materials requirement refers to cable management external to the cabinets

Changes in version 17

Section 4.0	Section further revised January 18 2018, clarifying throughput rules, and apps descriptions in table 4-1
Section 10.1.3	Tweak to wording to say “The SPMS will allow multiple versions of packaged software to be installed in Laboratory-specified locations . . .”

Changes in version 18

Section 3.4.1	New. Specifically calls out the one mandatory requirement in the NRE proposal which is the establishment of a Center of Excellence to support application porting and tuning.
Section 6.1.3.1.3	Single FEN Aggregate Transactions Performance TR simplified by deleting the phrase “from each FEN” and dropping “read” from the last sentence.
Section 6.1.3.2.9 and 6.1.3.2.10	Deleted these two ORNL IO system requirements regarding Shared File Overlapping Write and Read Performance
Section 6.2.3.3	Reduce requirement from 150% to 100%
Section 6.2.4.1.2 and 6.2.4.1.3	Remove read performance from these two requirements
Section 11.1.1	Replaced 24/7 hardware support with 9x5 hardware support . Left 12/7 as second support option and give Offeror the option to propose other support options

Changes in version 19

Section 4.4.5	Clarified allowed changes for the baseline benchmark runs
Section 6.2.4.2.1 and 6.2.4.2.2	Reduced 2022 system’s I/O bandwidth requirements

Changes in versions 20 and 21

	Editorial changes suggested by DOE Internal Review Board
Change in version 22	Missing TR level on 9.2.1.10 fixed to be TR-2
Change in version 23	Introduction section, page 9, 5th paragraph is revised to include the following: LLNL reserves the right to make an award based on the proposals for the ORNL 2021 system. Section 11.6 is revised to remove reference to “DOE P clearances”. The second sentence of the section is revised as shown.

Table of Contents

List of Figures

List of Tables

List of Equations

This work is performed under the auspices of the U.S. Department of Energy by Oak Ridge National Laboratory under contract DE-AC0500OR22725, Argonne National Laboratory under contract DE-AC02-06CH11357, and Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Requirements Definitions

Particular sections of these technical requirements have priority designations, which are defined as follows:

(a) Mandatory Requirements designated as (MR)

Mandatory Requirements (designated MR) in this draft build Statement of Work (SOW) are performance features that are essential to the Laboratories' requirements, and an Offeror must satisfactorily propose all Mandatory Requirements in order to have its proposal considered responsive.

(b) Mandatory Option Requirements designated as (MO)

Mandatory Option Requirements (designated MO) in this draft build SOW are features, components, performance characteristics, or upgrades whose availability as options to the Laboratories are mandatory, and an Offeror must satisfactorily propose all Mandatory Option Requirements in order to have its proposal considered responsive. The Laboratories may or may not elect to include such options in the resulting subcontract(s). Therefore, each MO shall appear as a separately identifiable item in Offeror's proposal.

(c) Technical Option Requirements designated as (TO-1, TO-2 or TO-3)

Technical Option Requirements (designated TO-1, TO-2, or TO-3) in this draft build SOW are features, components, performance characteristics, or upgrades that are important to the Laboratories, but that will not result in a nonresponsive determination if omitted from a proposal. Technical Options add value to a proposal. Technical Options are prioritized by dash number. TO-1 is most desirable to the Laboratories, while TO-2 is more desirable than TO-3. Technical Option responses will be considered as part of the proposal evaluation process; however, the Laboratories may or may not elect to include Technical Options in the resulting subcontract(s). Each proposed TO should appear as a separately identifiable item in an Offeror's proposal response.

(d) Target Requirements designated as (TR-1, TR-2 or TR-3).

Target Requirements (designated TR-1, TR-2, or TR-3), identified throughout this draft build SOW, are features, components, performance characteristics, or other properties that are important to the Laboratories, but that will not result in a nonresponsive determination if omitted from a proposal. Technical Requirements are prioritized by dash number. TR-1 is most desirable to the Laboratories, while TR-2 is more desirable than TR-3. Target Requirements add value to a proposal. Target Requirements are prioritized by dash number.

(e) Summary of MR, MO, TO and TR

The aggregate of MRs and TR-1s form a baseline system. TR-2s are goals that boost a baseline system, taken together as an aggregate of MRs, TR-1s and TR-2s, into a moderately useful system. TR-3s are stretch goals that boost a moderately useful system, taken together as an aggregate of MRs, TR-1s, TR-2s and TR-3s, into a highly useful system. Therefore, the ideal CORAL system will meet or exceed all MRs, TR-1s, TR-2s and TR-3s. Target Requirement responses will be considered as part of the proposal evaluation process.

MOs are alternative features, components, performance characteristics or system sizes that may be considered for technical and/or budgetary reasons. Technical Option Requirements may also

affect the Laboratories' perspective of the ideal CORAL system(s), depending on future budget considerations.

1.0 Introduction

This document contains the collective technical requirements of CORAL, a Collaboration of Oak Ridge National Laboratory (ORNL), Argonne National Laboratory (ANL) and Lawrence Livermore National Laboratory (LLNL), hereafter referred to as the Laboratories, for an exascale High Performance Computing (HPC) system to be delivered in 2021 to ORNL (accepted in 2022), a second exascale system delivered in 2022 to LLNL (accepted in 2023), and a potential 2022 exascale system at ANL. These systems are required to meet the mission needs of the Advanced Scientific Computing Research (ASCR) Program within the Department of Energy's Office of Science (SC) and the Advanced Simulation and Computing (ASC) Program within the National Nuclear Security Administration (NNSA).

DOE has a requirement that ORNL and ANL systems must always be diverse from one another. ANL has a system planned for acceptance in 2021called "A21", which is outside this RFP. Consequently, the ORNL system must be diverse from the A21 system. The LLNL system can be the same as the ANL system, the same as the ORNL system, or a different system altogether.

Diversity will be evaluated by how much it promotes a competition of ideas and technologies; how much it reduces risk that may be caused by delays or failure of a particular technology or shifts in vendor business focus, staff, or financial health; and how much the diversity promotes a rich and healthy HPC ecosystem.

In response to this RFP, Offerors may submit only one proposal for the ORNL system to be delivered in 2021 and one proposal for the LLNL system to be delivered in 2022 (ANL will select from the latter set of proposals). If the 2021 and 2022 systems are very similar, Offerors may submit a single proposal that covers both systems.

The delivery and acceptance timing provided above represents the current outlook and alignment of programmatic requirements and funding. However, ORNL reserves the right to make an award based on the proposals for the LLNL/ANL system. LLNL reserves the right to make an award based on the proposals for the ORNL 2021 system.

Additional information on proposal preparation will be provided in the *Proposal Evaluation and Proposal Preparation Instructions* (PEPPI).

This draft build SOW describes specific technical requirements (refer to the preceding Requirements Definitions Section for particular technical requirements definitions) related to both the hardware and software capabilities of the desired systems as well as application requirements.

The Laboratories anticipate the following schedule for the DOE 2021-2023 system acquisitions based on calendar years:

1Q 2017 Exascale Computing Project (ECP) released an RFI to gather information about potential systems and related Non-Recurring Engineering (NRE) for the next acquisitions at ORNL, LLNL, and ANL;

3Q 2017-1Q 2018 Vendor meetings to discuss the CORAL draft technical requirements;

1Q 2018 ORNL releases final CORAL RFP;

2Q 2018 proposal responses are due and will be evaluated by the CORAL evaluation team;

2Q 2018 select the ORNL system and the LLNL system;

2Q 2018 potentially select an ANL system;

3Q 2018 NRE contracts put in place for each unique system;

1Q-3Q 2019 separate acquisition contracts: at least 2 and up to three (one from each laboratory);

3Q 2021 exascale system delivered to ORNL;

3Q 2022 exascale system delivered to LLNL;

3Q 2022 exascale system delivered to ANL.

The Laboratories reserve their right to revise any or all of the points reflected in the above schedule based upon the Laboratories' and/or DOE's needs.

2.0 Program Overview and Mission Needs

2.1 Office of Science (SC)

The SC is the lead Federal agency supporting fundamental scientific research for energy and the nation's largest supporter of basic research in the physical sciences. The SC portfolio has two principal thrusts: direct support of scientific research and direct support of the development, construction, and operation of unique, open-access scientific user facilities. These activities have wide-reaching impact. SC supports research in all 50 States and the District of Columbia, at DOE laboratories, and at more than 300 universities and institutions of higher learning nationwide. The SC user facilities provide the Nation's researchers with state-of-the-art capabilities that are unmatched anywhere in the world.

Within SC, the mission of the Advanced Scientific Computing Research (ASCR) program is to discover, to develop, and to deploy computational and networking capabilities to analyze, to model, to simulate, and to predict complex phenomena important to the DOE. A particular challenge of this program is fulfilling the science potential of emerging computing systems and other novel computing architectures, which will require numerous significant modifications to today's tools and techniques to deliver on the promise of exascale science.

2.2 National Nuclear Security Administration (NNSA)

The NNSA, an agency within the DOE, is responsible for the management and security of the nation's nuclear weapons, nuclear nonproliferation, and naval reactor programs. The NNSA Strategic Plan supports the Presidential declaration that the United States will maintain a "safe, secure, and effective nuclear arsenal." The Plan includes ongoing commitments:

- to understand the condition of the nuclear stockpile; and
- to extend the life of U.S. nuclear warheads.

The NNSA's Advanced Simulation and Computing (ASC) Program provides the computational resources that are essential to enable nuclear weapon scientists to fulfill stockpile stewardship requirements through simulation without underground testing. Modern simulations on powerful computing systems are key to supporting this national security mission.

As the stockpile moves further from the nuclear test base, through aging of stockpile components or modifications involving system refurbishment, reuse, or replacement, the realism and accuracy of ASC simulations must increase significantly through development and use of improved physics models and solution methods, which will require orders of magnitude greater computational resources than are currently available.

2.3 Mission Needs

Today, HPC-based modeling and simulation is used extensively in the advancement of DOE missions in science and engineering and in the stewardship of the nation's nuclear weapons stockpile. However, to maintain leadership and to address the future challenges in national security, science, energy, health, and growing security threats, the U.S. must make a strategic move in HPC—a grand convergence of modeling and simulation, data analytics, and deep learning. Arising out of this convergence will be new approaches to predictive analysis for scientific discovery and stockpile stewardship, and solutions to complex data-driven engineering problems.

The exascale systems will need to be designed to address the emerging convergence of data science, machine learning, and simulation science. Further, they will need to run simulations that require the entire platform and take days to weeks to complete. Supervised learning on these systems will be used to do uncertainty quantification and to discover the complex, non-linear relationships in the output of large multi-physics simulations and large scientific experiments. These systems' new capabilities will revolutionize areas such as energy production, materials design, chemistry, and precision health care.

Science Needs: For the past decade, the six science programs in the DOE Office of Science have formulated strategic plans for the disciplines that they steward. Consistently, these plans rely on HPC in ever increasing proportion and, in recent years, the explicit call for HPC at exascale performance levels has been a common and defining theme. Examples include discovery and characterization of next-generation materials; systematic understanding and improvement of chemical processes; analysis of the extremely large datasets resulting from the next generation of particle-physics experiments; and extraction of knowledge from systems-biology studies of the microbiome. Advances in applied energy technologies also are dependent on next-generation simulations, notably whole-device modeling in plasma-based fusion systems.

National Security Needs: The NNSA Stockpile Stewardship Program, which underpins confidence in the U.S. nuclear deterrent, has been successful since its inception in 1995, largely as a result of HPC-based modeling and simulation tools used in the NNSA's Annual Assessment process, as well as solving issues arising from Significant Finding Investigations. In the coming decade, the importance and role of HPC in this area will intensify, and HPC-based modeling and simulation tools will be increasingly called upon to provide the required confidence, using robust uncertainty quantification techniques, in lifetime extensions of warheads in the U.S. nuclear weapons stockpile. HPC tools also will have an increasing role in understanding evolving nuclear threats posed by adversaries, both state and non-state, and in developing national policies to mitigate these threats.

3.0 CORAL High-Level System Requirements

The requirements for the systems are based on the definition of a capable exascale system defined by DOE's Exascale Computing Project. A capable exascale system is defined as a supercomputer that can solve science problems 50x faster (or more complex, for example, more physics, higher fidelity) than the 20PF systems of today can solve comparable problems. Further, a capable exascale system must use a software stack that meets the needs of a broad spectrum of applications and workloads, within a power envelope of 30 MW, and must be sufficiently resilient such that user intervention due to hardware or system faults is required on the order of a week.

This section describes the high-level requirements for the ORNL and LLNL/ANL system proposals. In addition, the PEPPI provides system and node tables for the Offeror to complete and to submit separately as well as to include here.

3.1 Description of the CORAL System(s) (MR)

The Offeror shall provide a concise description of its proposed CORAL system architecture, including all major system components plus any unique features that should be considered in the design. The description shall include:

- A statement whether this is an ORNL system proposal, or a LLNL/ANL proposal, or both;
- An overall system architectural diagram that shows all node types and their quantity, the interconnect(s), and the I/O subsystem as well as the latencies and bandwidths of data pathways between components;
- An architectural diagram of each node type showing all elements of the node along with the latencies and bandwidths to move data between the node elements;
- A high-level overview of the proposed I/O subsystem design, whether the solution is monolithic or tiered, and, if Offeror's solution consists of multiple storage tiers, a description of the interface between tiers and an indication of the extent to which the tiers can function independently of one another.

If the proposal is for both an ORNL system and a LLNL/ANL system then the differences between the two systems should be clearly identified. They should be described at a high level in the system description. They should also be identified wherever they are relevant to a particular subsection. The Offeror is strongly encouraged to use tables and other summary methods to capture these differences.

In order to reduce risks with the long delivery time range, the Offeror may propose late-binding of technology. In this case, the response shall concisely describe the complete system architecture that the Offeror currently considers to include the best choices, and then the response can propose alternative technologies (for example, for processor, memory or interconnect) and the dates when late-binding of these technologies would be decided. Regardless of the choices for these late-binding decisions, the resulting system should meet all CORAL high-level system requirements.

The Offeror shall describe how the proposed system fits into their long-term product roadmap and how the technologies, architecture, and programming model address the emerging convergence of data science, machine learning, and simulation science.

3.2 High Level CORAL System Metrics

A design that meets or exceeds all metrics outlined in this section is strongly desired.

3.2.1 CORAL System Performance (TR-1)

CORAL places a particularly high importance on system performance. The Offeror will provide projected performance results for the proposed system for the four Scalable Science Benchmarks, the four Throughput Benchmarks, and the two Data Science and Deep Learning Benchmarks. The target (speedup) performance requirements will be at least $S_{scalable}$= 50 and $S_{throughput}$= 50, S_{data} = 50 respectively, where $S_{scalable}$ and $S_{throughput}$ and S_{data} are defined in Section 4.0. The Offeror will also provide the performance results for the Skeleton Benchmarks. The Offeror will explain the methodology used to determine all projected results. (Note: the values of S are based on Titan/Sequoia.)

3.2.2 CORAL System Peak (TR-1)

The CORAL baseline system performance will be at least 1,300 petaFLOPS ($1300x10^{15}$ double-precision floating point operations per second).

3.2.3 Total Memory (TR-1)

The system will have an aggregate of at least 8 PiB of memory available for direct application use. This total can consist of any memory directly addressable from a user program. Example memory technologies that may be included in this total are HBM, DDR DIMMs, and NVRAM. The memory configuration of the proposed system will be the memory configuration used to achieve the benchmark results reported in the Offeror's proposal.

CORAL will place high value on solutions that maximize the ratio of "fast and close" memory to "far and slow" memory. In particular, exascale application developers have indicated that they will require 4-6 PiB of HBM.

3.2.4 Memory per MPI Process (TR-2)

A minimum of 1 GiB per MPI process will be provided, with 2 GiB per MPI process preferred, counting all directly addressable memory types available. For systems that provide less than this amount of memory, threading should be used to obtain additional MPI concurrency.

3.2.5 Maximum Power Consumption (TR-1)

The maximum power consumed by the 2021 or 2022 CORAL system and its peripheral systems, including the proposed storage system, will not exceed 40MW, with power consumption between 20MW to 30MW preferred. The maximum power consumption includes all equipment provided by the proposal.

3.2.6 System Resilience (TR-1)

The Mean Time Between Application Failure due to a system fault requiring user or administrator action will be at least 144 hours (6.0 days). The failure rates of individual components can be much higher as long as the overall system adapts, and any impacted jobs or services are restarted without user or administrator action.

This resilience requirement is designed to maximize the different ways that a system could be designed and still perceived to be resilient. At one extreme, a system could be proposed with highly reliable (and possibly redundant) components to the point that a system fault that causes an application failure is only expected every 144 hours. At the other extreme, a system could be proposed with very unreliable components and the system software would have to compensate by being very reliable and adaptable so that no user or administrator action is required for jobs to continue to run. There are many design choices between these extremes that the Offeror is free to consider.

Applications that fail must be restarted. If they have no user checkpointing, they can be restarted from the beginning. If the application uses services that have been affected by a system fault, these services must be automatically restored so that the application can continue to progress.

3.2.7 Application Reliability Overhead (TR-1)

An application with user check-pointing, using at least 90% of the CNs, will complete with correct results without human intervention with less than 50% overhead due to recovery from system faults.

3.3 CORAL High Level Software Model (MR)

Offeror shall include a high-level software architecture diagram. This diagram shall show all major software elements. The Offeror shall also describe the expected licensing strategy for each. The CORAL software model and resulting requirements shall be described from the perspective of a highly scalable system that consists of compute nodes (CNs) that number in the range of tens to hundreds of thousands, and an I/O subsystem that provides a scalable solution that utilizes unit building blocks that allow for modular expansion and deployment, and a Front End Environment (FEE) of sufficient capability to provide access, system management, and programming environment to the system. The system must also provide Linux-like capability and compatibility. The Offeror shall describe how the system software: tracks early signs of system faults; manages power dynamically; collects power and energy statistics; and reports accurate and timely information about the hardware, software, and applications from all components in the system. The Offeror is only required to supply one OS per node type, but the architecture shall not preclude the booting of different node OSs, which will allow system software developers to run jobs that use OS configurations targeted for data-centric, SPMD, or dynamic multithreaded programming models.

3.3.1 Open Source Software (TR-1)

The Laboratories strongly prefer that all offered software components are Open Source.

3.4 CORAL High Level Project Management (MR)

The Offeror's proposal shall include the following:

- Overview of collaboration plan and discussion of any requirements that the Offeror has for CORAL in the management of the project.
- Preliminary Risk Management Plan that describes any aspects or issues that the Offeror considers to be major risks for the system, including management of secondary subcontractors, and planned or proposed management and mitigations for those risks including late-binding.
- Discussion of delivery schedule and how Offeror would manage the large system delivery and deployment, e.g., personnel and communications, factory staging, onsite staging, installation, integration, testing, and bring-up. Any schedule risks shall be clearly identified.
- Discussion of Offeror's general approach for software licensing, e.g., range of licenses and criteria for selection from that range of licenses for a particular package.
- Discussion of Offeror's quality assurance and factory test plans.

3.4.1 Center of Excellence (MR)

The Offeror shall include in the NRE proposal a Center of Excellence task that consists of multiple milestones to support the Laboratories in porting key DOE applications to the CORAL-2 system and improving the performance of DOE applications on the CORAL-2 system. Activities will require the support of experts in the areas of application porting and performance optimization, who will work with laboratory personnel on porting and tuning of key applications, which may include some of the CORAL-2 benchmarks, or full applications—to be determined during contract negotiation—for the proposed system. This task should be run as its own project, with a coordinator/project manager overseeing and coordinating issues across labs as appropriate. Colocation of staff at the laboratory sites is desirable but not necessary. Base support is required from the date of subcontract execution through 2 years after final

acceptance. Laboratories may negotiate an extended period of performance or options for such an extension. This activity shall reflect all terms and conditions of NRE activities including cost sharing.

3.5 Early Access to CORAL Hardware Technology (TR-1)

The Offeror will propose mechanisms to provide the Laboratories with early access to hardware technology for hardware and software testing prior to inserting the technology into the CORAL system. Small early access systems are encouraged, particularly if they are sited at the Laboratories.

3.6 Early Access to CORAL Software Technology (TR-1)

The Offeror will propose mechanisms to provide the Laboratories with early access to software technology and to test software releases and patches before installation on the CORAL system.

3.7 CORAL Hardware Options

The Offeror shall propose the following separately priced options using whatever is the natural unit for the proposed architecture design as determined by the Offeror. For example, for system size, the unit may be number of racks, peak PF, number of blades, or some other unit appropriate for the system architecture. If the proposed design has no option to scale one or more of these features, the Offeror should simply state this in the proposal response.

3.7.1 Scale the System Size (TO-1)

The Offeror will describe and separately price options for scaling the overall CORAL system up or down as different sites may desire different size systems. These options may be larger than the smallest usable compute partition. In addition to scaling options for the primary 2022/2023 CORAL system, Offeror will provide separately priced options for systems that provide approximately 10%, 20% and 30% of the capability of the primary system, which the Laboratories may procure in addition to a primary system.

3.7.2 Scale the System Memory (TO-1)

The Offeror will describe and separately price options for configuring the CORAL system memory with respect to the capacity and the ratio of different memory types as different sites may desire different configurations and ratios of memory types. The Offeror will describe and separately price options for using memory devices of different capacities for each type of memory and will detail any performance implications of using those sizes. The Offeror will include at least one separately priced option for each type of memory that a user program can directly address.

3.7.3 Scale the System Interconnect (TO-1)

The Offeror will describe and separately price options for configuring the high performance interconnect as different sites may prefer cost savings provided by reducing bandwidth or network connectivity.

3.7.4 Scale the I/O Subsystem Capacities and Performance (TO-1)

The Offeror will describe and separately price options for scaling the capacity and performance of the CORAL I/O subsystem as different sites may desire different configurations. If Offeror’s I/O subsystem solution consists of multiple storage tiers, the Offeror will describe and separately price options for scaling each storage tier separately.

3.7.5 Scale the Integrated Telemetry Data Base (TO-1)

The Offeror will provide separately priced options to scale the Integrated Telemetry Data Base to handle additional log data.

3.7.6 CORAL-Scalable Unit System (MO)

The Offeror shall propose, as a separately priced option, a CORAL system configuration called CORAL-SU, that consists of the minimum deployable system. The CORAL laboratories may exercise this option for their respective sites. This option shall include the smallest usable compute partition as well as a minimal front-end environment and an I/O subsystem containing all architectural components of the full solution. Options and costs for scaling the CORAL-SU up to larger configurations shall be provided. Specifically, Offeror shall describe and separately price options for the smallest usable compute partition, as well as options for any infrastructure required to add it to a CORAL-SU or larger system. Further, if the system includes different compute node types; the Offeror shall describe and separately price options to increase each compute node type independently of the other compute node types.

3.7.7 Options for Mid-Life Upgrades (TO-1)

The Offeror will describe and separately price any options for upgrading the proposed CORAL system over its five year lifetime.

4.0 CORAL Application Benchmarks

Future DOE supercomputers will need to solve emerging data science and machine learning problems in addition to the traditional modeling and simulations applications. The CORAL procurement is a major leap in capability to exascale class systems. The preferred solution will be one that provides innovative solutions for hardware with a demonstrable path toward performance portability using a software stack and tools that will ease the transition without sacrificing DOE goals for continued delivery of top predictive science and stockpile stewardship mission deliverables.

The past 15-20 years of computing have provided an almost unprecedented stability in high-level system architectures and parallel programming models, with the MPI, OpenMP, C++, and Fortran standards paving the way for performance portable code. Combined with the application trends toward more coupled physics, predictive capabilities, sophisticated data management, object oriented programming, and massive scalability – each of the DOE applications that CORAL systems will run represents tens or hundreds of person-years of effort, and *thousands* of person-years in aggregate. Thus, there is a keen interest in protecting the Laboratories' investment in the DOE application base by procuring systems that allow today's workhorse application codes to continue to run without radical overhauls. Both SC and NNSA applications seek solutions that minimize disruptive changes to software that are not part of a standard programming model likely to be available on multiple future acquisitions, while recognizing the need that the existing software base must continue to evolve.

The CORAL benchmarks have thus been carefully chosen and developed to represent the broad range of applications expected to dominate the science and mission deliverables on the CORAL systems.

Note: This section describes the CORAL benchmarks list as of January 31, 2018.

4.1 Benchmark Categories

The benchmarks have been divided into four broad categories that represent the envisioned system workloads and targeted benchmarks to allow specific insight into features of the proposed system.

Scalable Science Benchmarks are full applications that are expected to scale to a large fraction of the CORAL system and that are typically single physics applications designed to push the boundaries of human understanding of science in areas such as material science and combustion. Discovery science is a core mission of the Office of Science and predictive science for NNSA; these benchmarks represent important applications that ensure the DOE remains on the forefront of pioneering breakthroughs. Moreover, the primary mission of the Office of Science Leadership Computing Facilities is to enable the most ambitious capability computing at any given moment, where scalability is of singular importance.

Throughput Benchmarks represent particular subsets of applications that are expected to be used as part of the everyday workload of science applications at all CORAL sites. In particular, Uncertainty Quantification (UQ) is a driving mission need for the ASC program, where the CORAL system at LLNL is expected to run large ensembles of 10s or 100s of related calculations, with a priority on minimizing the ensemble's overall throughput time. The throughput benchmarks test how well CORAL machines reduce the time to completion of a small number of larger jobs that are executed simultaneously in UQ workloads.

Data Science and Deep Learning Benchmarks represent the operations and algorithms more common in data analytics and machine learning. They stress lower precision floating point as well as integer operations, instruction throughput, and indirect addressing capabilities, among other system characteristics. The exascale systems must be designed to address the emerging convergence of data science, machine learning, and simulation science. A few examples of this convergence are: supervised learning methods used to capture the complex, non-linear relationships in the output of large multi-physics simulations and large science experiments; unsupervised learning used to guide multi-scale physics runs through a large-scale state-space; in situ analysis and extraction of information while a simulation is running.

Skeleton Benchmarks reproduce the memory or communication patterns of a physics application or package, and make little or no attempt to investigate numerical performance. They are useful for targeted investigations such as network performance characteristics at large scale, memory access patterns, thread overheads, bus transfer overheads, system software requirements, and I/O patterns.

In general, the Scalable Science Benchmarks are full applications, and thus large in terms of lines-of-code. However, the CORAL team will ensure that the portion of the application that is exercised with the benchmark test cases is minimized and well-documented.

The CORAL benchmarks are divided into two tiers. Each tier contains a variety of benchmarks from the four benchmark categories (scalable science, throughput, data science/machine learning, skeleton). For the Tier 1 benchmarks, the Offeror shall provide a detailed performance analysis and projected figure of merits on the proposed CORAL architecture(s). The Tier 2 benchmarks are provided as examples of additional benchmarks of interest to DOE, but the Offeror should not provide a detailed performance analysis in the RFP response. Enhancing the performance of tier 2 benchmarks may be integrated into future Center of Excellence activities between the Offeror and CORAL labs. The tier 1 and tier 2 benchmarks are listed in Table 4-1.

Table 4-1: CORAL Tier 1 and Tier 2 Benchmarks

CORAL Tier 1 Benchmarks									
		Parallelism			Language				
Code	Lines of Code	MPI	OpenMP/ pthreads	GPU	F	Py	C	C++	Description
Scalable Science Benchmarks									
QMCPACK	200,000	X	X	X			X	X	QMCPACK is a many-body ab initio quantum Monte Carlo code for computing the electronic structure of atoms, molecules, and solids. It is written primarily in C++, and its use of template metaprogramming is known to stress compilers. When run in production, the code is memory bandwidth sensitive, while still needing thread efficiency to realize good performance.
HACC	35,000	X	X	X				X	The Hardware Accelerated Cosmology Code (HACC) framework uses N-body techniques to simulate the formation of structure in collisionless fluids under the influence of gravity in an expanding universe. It depends on an external FFT library and is typically compute limited achieving 13.92 Petaflops (69.2% of machine peak) on Sequoia.
NEKbone	48,000	X		X	X		X		Nekbone is a mini-app derived from the Nek5000 CFD code, which is a high order, incompressible Navier- Stokes CFD solver that is based on the spectral element method. The conjugate gradiant solve is compute intensive, contains small messages and frequent allreduces.
LAMMPS	500,000	X	X	X			X	X	LAMMPS is a classical molecular dynamics code. Performance limiters will depend on the problem chosen and could include : compute, memory bandwidth, network bandwidth, and network latency. The LAMMPS problem will utilize a ReaxFF potential.
Throughput Benchmarks									
Quicksilver	10,000	X	X	X				X	Monte Carlo transport benchmark with multi-group cross section lookups. Stresses memory latency, significant branching, and one large kernel that is 1000s of lines.
Kripke	4,000	X	X	X				X	Kripke is a structured deterministic (Sn) transport simulation that uses RAJA. It contains wavefront algorithms that stress memory latency and/or bandwidth, and network latency.
AMG (Solver phase)	65,000	X	X	X			X		AMG is a parallel algebraic multigrid solver for linear systems arising from problems on unstructured grids. AMG is memory-access bound, generates many small messages and stresses memory and network latency. The AMG problem inlcudes only the AMG solver phase, not the setup phase.
PENNANT	3,300	X	X	X				X	PENNANT is a mini-app for hydrodynamics on general unstructured meshes in 2D (arbitrary polygons). It makes heavy use of indirect addressing and irregular memory access patterns.
Data Science and Deep Learning Benchmarks									
Big Data Analytic Suite									The big data analytic suite contains: K-Means benchmark defined by computing the observation labels (class assignments) and centroids for k=2, 3, and 4. The PCA benchmark defined as computing the first and last of the "standard deviations" from a PCA on a distributed matrix. The SVM benchmark consisting of a linear 2-class SVM fit and calculating the feature weights.

Table 4-1: CORAL Tier 1 and Tier 2 Benchmarks (continued)

CORAL Tier 1 Benchmarks									
		Parallelism			Language				
Code	Lines of Code	MPI	OpenMP/ pthreads	GPU	F	Py	C	C++	Description
Deep Learning Suite									The deep learning suite contains: Convolutional Neural Networks (CNNs) that comprises convolutional layers followed by fully connected layers; LSTM recurrent neural network (RNN) architecture; and, distributed training code for classification in ImageNet data set at scale. Finally, the CANDLE benchmark codes implement deep learning architectures that are relevant to problems in cancer. These architectures address problems at different biological scales, specifically problems at the molecular, cellular and population scales. We will use two diverse benchmark problems, namely, a) P1B1, a sparse autoencoder to compress the expression profile into a low-dimensional vector, and, b) P3B1 a multi-task deep neural net for data extraction from clinical reports.
Skeleton Benchmarks									
CORAL MPI Benchmarks	1,000	X					X		Subsystem functionality and performance tests. Collection of independent MPI benchmarks to measure various aspects of MPI performance including interconnect messaging rate, latency, aggregate bandwidth, and collective latencies.
Memory Benchmarks	1,500		X				X		Memory subsystem functionality and performance tests. Collection of STREAMS and STRIDE memory benchmarks to measure the memory subsystem under a variety of memory access patterns.
ML/DL micro-benchmark Suite									Sparse and dense convolutions, FFT, double, single and half precision GEMM and other machine/deep learning math algorithms not included in other CORAL benchmark suites.
I/O Suite	4,000	X					X		IOR, Simul, MDTest, FTree
Pynamic	12,000	X				X		X	Subsystem functionality and performance test. Dummy application that closely models the footprint of an important Python-based multi-physics ASC code.
CLOMP			X				X		Threading benchmark focused on performance of thread overheads evaluation.
RAJA Performance Suite			X	X				X	The RAJA performance suite is designed to explore performance of loop-based computational kernels of the sort found in HPC applications. In particular, it is used to assess, monitor, and compare runtime performance of kernels implemented using RAJA and variants implemented using standard or vendor-supported parallel programming models directly. Each kernel in the suite appears in multiple RAJA and non-RAJA variants using parallel programming models such as OpenMP and CUDA.

Table 4-2: CORAL Tier 2 Benchmarks

CORAL Tier 2 Benchmarks									
		Parallelism			Language				
Code	Lines of Code	MPI	OpenMP/ pthreads	GPU	F	Py	C	C++	Description
Scalable Science Benchmarks									
ACME		X	X	X	X				ACME is a high-resolution climate simulation code for the entire Earth system, containing five major components for the atmosphere, ocean, land surface, sea ice, and land ice along with a coupler. Performance limiters will be network latency, memory bandwidth, kernel launch overheads on accelerators, and accelerator data transfer latency. ACME also stresses Fortran compiler compliance and efficiency.
VPIC	90,000	X	X					X	Vector Particle-In-Cell) is a general purpose particle-in-cell simulation code for modeling kinetic plasmas. It employs a second-order, explicit, leapfrog algorithm to update charged particle positions and velocities in order to solve the relativistic kinetic equation for each species in the plasma, along with a full Maxwell description for the electric and magnetic fields evolved via a second- order finite-difference-time-domain (FDTD) solve.
Throughput Benchmarks									
Laghos	2,000 + MFEM	X						X	Laghos solves the time-dependent Euler equation of compressible gas dynamics in a moving Lagrangian frame using unstructured high-order finite element spatial discretization and explicit high-order time-stepping. It is built on top of a general discretization library (MFEM) and supports two modes: *full assembly*, where performance is limited by the data motion in a conjugate gradient (CG) solve, and *partial assembly*, where performance depends mostly on small dense matrix operations and the CG solve communication
AMG (Setup phase)	65,000	X	X	X			X		AMG is a parallel algebraic multigrid solver for linear systems arising from problems on unstructured grids. AMG is memory-access bound, generates many small messages and stresses memory and network latency. The AMG setup phase has proven challenging to parallelize on accelerators.
Data Science and Deep Learning Benchmarks									
Parallel Integer Sort	2,000	X	X		X		X		Massively parallel integer sort of a large number of 64-bit integers. Data sets can be sized to run in system memory or to exceed aggregate memory and to require IO. The benchmark emphasizes integer operations, IO, and all-to-all commuications.
Havoq		X		X				X	The Havoq massively parallel graph analysis algorithms for computing triangles, edges, vertices. Emphasizes load imbalance and irregular random memory accesses.

4.2 Benchmark Availability

The benchmark source codes are available via the Web at the following URL:

https://asc.llnl.gov/coral-2-benchmarks/

This site will be maintained with updated information throughout the proposal response period, including updates to instructions for build and execution, as well as the rare possibility of a change in the baseline Figures of Merit (FOMs) due to late discovery of a bug or issue with the baselining procedures performed by the CORAL team.

The entire set of benchmarks listed in Table 4-1 have been executed on the existing ASCR Leadership Class or ASC Advanced Technology Systems in DOE (i.e., Titan, Sequoia/Mira, Theta) to provide baseline execution performance. The benchmark website provides the results of these runs as an aid to Offerors.

4.3 Performance Measurements (Figures of Merit) (TR-1)

All performance measurements for the benchmarks are stated in terms of a FOM specific to each benchmark. Each benchmark code defines its own FOM based on the algorithm being measured in the benchmark and represents a rate of execution based on, for example, iterations per second or simulated days per day. FOMs are defined so that they scale linearly (within measurement tolerances) with the delivered performance of the benchmark. For example, running a given benchmark 10x faster should result in a FOM that is ~10x larger. Likewise, running a 10x more complex problem (higher fidelity, more physics, or larger size) in the same amount of time should also result in a ~10x increase in the resulting FOM. It is allowable to accomplish the 50x increase by combining both, for example, running a 5x more complex application 10x faster.

The value of the FOM for each benchmark is described in its documentation and is printed out (usually to stdout) at the end of its successful execution.

Each benchmark projected FOM must be normalized to the measured baseline FOM. The resulting ratio should reflect the CORAL goals specified in Section 3.2.1 of at least 50x improvement on scalable science workloads and throughput workloads, and at least 50x improvement on data science and deep learning workloads relative to current systems. The Offeror simply needs to measure or to project the FOM on the target platform, and then take the ratio of the projected FOM over the supplied FOM baseline.

The normalized performance (S_i) metric for each Science, Throughput, or Data benchmark is then:

$$S_i = \textit{projected FOM}_i \,/\, \textit{baseline FOM}_i$$

Equation 4-1: Calculation of Individualized Normalized FOM

The projected performance (S) metric for the Scalable Science is the geometric mean of all S_i in Scalable Science. Similarly, for the projected performance (S) of Throughput, and Data Science and Deep Learning Benchmarks (see **Equation 4-2**, **Equation 4-3** and **Equation 4-4** below).

4.4 Benchmarking Procedures

Each benchmark includes a brief summary file, and a tar file. Tar files contain source code and test problems. Summary files contain instructions for determining that the code has been built correctly and the CORAL RFP problems to run. RFP problems are usually a set of command line arguments that specify a problem setup and parameterization, and/or input files.

All of the Scalable Science and Throughput, and Data Science benchmarks are designed to use a combination of MPI for inter- or intra-node communication, and some form of threading for additional on-node parallelism within a shared memory coherency domain. The Offeror is allowed to choose the ratio of MPI vs. threading that will produce the best results on their system. Although many of the benchmarks do not require 1 GiB/task as they are written, unless the CORAL benchmark website states that a smaller amount of main memory is needed, the amount of main memory per MPI task for the Scalable Science and Throughput Benchmark calculations should meet requirement 3.2.4.

4.4.1 Scalable Science Benchmarks

The Scalable Science Benchmarks were selected to test the limits of system scalability, and represent how the CORAL institutions generate significant scientific results through increased fidelity. The Offeror should provide a projected FOM for each benchmark problem run at between 90% and 100% scale of the proposed system. The calculation of the geometric mean, $S_{scalable}$, must include the four Tier 1 Scalable Science Benchmarks

$$S_{scalable} = \left(\prod_{i=1}^{4} S_i \right)^{\frac{1}{4}}$$

Equation 4-2: Calculation of Aggregate Scalable Science Benchmark Metric

The goal of scalable science is to push the boundaries of computing by demonstrating the calculation of problems that could not previously be performed. As such, the *preference* for the Scalable Science Benchmarks is that the Offeror achieve the designed increase in FOM by demonstrating the ability to run problems that are at least 5-10x larger than the baseline problems. The individual descriptions of each of the Scalable Science Benchmarks found on the benchmark website provide information regarding how this increase in problem size can be performed. After this measure of weak scalability has been demonstrated, the remainder of the 50x increased performance for the target FOM can be achieved via weak or strong scaling, with weak scaling preferred.

4.4.2 Throughput Benchmarks

The Throughput Benchmarks are intended to provide an example of how CORAL machines will support a small number of larger jobs that are executed simultaneously, e.g., in a large UQ ensemble study or mid-range capability multi-physics simulation. The Throughput Benchmarks do not stress the full scalability of the system for a single job, but are meant to demonstrate how a fully loaded machine impacts the performance of quarter system sized jobs. They require taking into consideration issues such as message contention in the interconnect and how other system resources are shared across job partitions to determine the total amount of work (throughput) that can be accomplished.

A single, aggregated performance metric for the Throughput Benchmark runs will be referred to as $S_{throughput.}$. The Throughput Benchmark performance metric is calculated by projecting the FOM for each benchmark application, accounting for any performance degradation as a result of the system being fully loaded, and then calculating the geometric mean. The calculation of $S_{throughput}$ must include all four Tier-1 Throughput Benchmarks. How to run multiple copies for the baseline FOMs is shown in Figure 4-1.

	¼ of nodes	¼ of nodes	¼ of nodes	¼ of nodes
Application run	AMG	Kripke	Quicksilver	Pennant

Figure 4-1: Shows the $S_{throughput}$ baseline run which requires the four Throughput applications running simultaneously on ¼ of system.

All applications will repeat their calculations long enough for the longest application to finish. Each benchmark is given equal importance in the final $S_{throughput}$. The formula for calculating $S_{throughput}$ is:

$$S_{throughput} = \left(\prod_{i=1}^{4} S_i \right)^{\frac{1}{4}}$$

Equation 4-3: Calculation of Aggregate Throughput Benchmark Metric for Baseline Runs

Unlike the Scalable Science Benchmarks, the Throughput Benchmarks have two sets of rules: one for baseline projections and one for optimized projections. The baseline rules are:

1) The total number of nodes being modeled should be between 90% and 100% of the total nodes on the proposed system.
2) The number of nodes that each individual job uses should be approximately the same
3) For all runs no application should use more than $1/4^{th}$ of the total nodes on the system.
4) The vendor will run a single copy of each benchmark for the four simultaneous running jobs used to calculate $S_{throughput}$ as shown in Figure 4-1.

For $S_{throughput}$ optimized projections, the Offeror is encouraged to demonstrate how to maximize throughput on their system by running jobs on the node counts that highlight the optimal way to maximize the throughput of their proposed system. For optimized throughput projections the Offeror is allowed additional latitude, but the chosen configuration cannot exceed any resource available on the proposed system.

The optimized $S_{throughput}$ equation 4.4 allows the Offeror to account for the ability of a machine to effect more total throughput by running fewer or more than four problems simultaneously and multiple copies of each application if resources are available and aggregate throughput increases.

All optimized $S_{throughput}$ projections must account for the following guidelines and constraints in addition to abiding by rules 1 and 2 from the baseline projection section.

1) Each application contains a CORAL target problem size, and this specified problem size must be used for the FOM runs regardless of the number of copies of each application used or the number of nodes they run on.
2) All problems will run on the same number of nodes or be allocated the same number of nodes and idle the extra nodes
3) If more than one copy of an application is run all runs will occur simultaneously and all applications will run the same number of copies.

Examples of a couple valid optimized projection configurations are shown in Figures 4-2 and 4-3. Note that if the FOMs of each example were the same, the example in Figure 4-3 is preferred due to fewer nodes being used per job.

	½ of nodes	½ of nodes
Application run 1	Kripke	AMG
Application run 2	Pennant	Quicksilver

Figure 4-2: Shows an example $S_{throughput}$ optimized runs with two benchmarks running simultaneously on half the system.

	1/8 of nodes	1/8 of nodes	1/8 of nodes	1/8 of nodes	1/8 of nodes	1/8 of nodes	1/8 of nodes	1/8 of nodes
Application run	Kripke	Kripke	AMG	AMG	Pennant	Pennant	Quicksilver	Quicksilver

Figure 4-3: Shows an example $S_{throughput}$ optimized run with 2 copies of each Throughput benchmark running simultaneously on the same number of nodes.

If a vendor uses multiple application runs then the FOM is calculated as follows:

$$S_{throughput} = \frac{(\prod_{i=1}^{4} S_i)^{\frac{1}{4}}}{number\ of\ application\ runs}$$

Equation 4-4: Calculation of Aggregate Throughput Benchmark Metric for Optimized Runs

If multiple copies of the same application are run then each applications S_i is the sum of those runs and Equation 4-3 is used.

The CORAL target problem sizes were chosen to reach the 50x FOM increase as follows:

Problem sizes were selected that are **2-10x larger** than the baseline calculation and all target problems will fit into 1,250 TiB (1/4) of memory. The problems should be projected to complete at least **5-25x faster** – giving an **overall FOM increase of 50x.** Be advised that achieving a target $S_{throughput}$ via running fewer copies of each job (strong scaling) will be looked at less favorably than achieving the same $S_{throughput}$ via more copies of a job because it is easier to schedule a machine when the workload contains smaller jobs. In addition, the intent of the benchmark problem sizes is that the offeror will need 5 PiB of total memory to fit the largest benchmarks (AMG and Kripke) for the unoptimized runs. For the optimized runs it is expected that the maximum number of copies of each application that an offeror can run will be equal to y where the entire machine contains 5*y PiB of memory. If a machine has multiple levels of memory it would need 1.25 PiB of the type of memory of a given level to fit a single copy of the largest throughput benchmarks in that level of memory if strong scaled to run across the entire system.

4.4.3 Data Science and Deep Learning Benchmarks

The Data Science and Deep Learning Benchmarks are intended to test how well the CORAL machines will handle scientific data analytics, deep learning, and machine learning. Each of the two Tier 1 data science and deep learning benchmark suites consist of one or more motifs that cover some of the key algorithms and methods in this area and machine characteristics such as lower precision floating point, integer operations, instruction throughput, and indirect addressing performance. The CANDLE benchmarks require the use of Tensorflow or similar framework. Each Data Science and Deep Learning benchmark includes the calculation of its figure of merit S_i value that takes into consideration both the execution rate and the size of the problem.

A single, aggregated performance metric S_{data} is calculated by the formula:

$$S_{data} = \left(\prod_{i=1}^{2} S_i \right)^{\frac{1}{2}}$$

Equation 4-5: Calculation of Aggregate Data Science and Deep Learning Benchmark Metric

Benchmark suites, which include multiple benchmarks, use their own internal weighting schemes that calculate a specific S_i. The target S_{data} improvement is 50x. It is expected that this will be achieved through a combination of hardware improvements, software framework tuning and application work.

Each Data Science and Machine Learning benchmark is unique and the run rules for each vary as well. The specific details for each benchmark are included in their summary files on the benchmark website.

4.4.4 Skeleton Benchmarks

Skeleton benchmarks are designed to target specific aspects of the machine, and are used by the CORAL evaluation team to determine characteristics such as measured versus peak performance/bandwidth, overall system balance, and areas of potential bottlenecks. No normalization of the skeleton benchmarks is required. The Skeleton benchmark results should be reported as their estimated raw values on the proposed CORAL system.

4.4.5 Allowed Modifications for the Scalable Science, Throughput, and Data Science Benchmarks

The source code and compile scripts downloaded from the CORAL benchmark web site may be modified as necessary to compile and to run the benchmarks on the Offeror's system. Other allowable changes to the compile scripts include optimizations obtained from compiler flags that do not require modifications of the source code. The only source code changes allowed for the baseline are to add or to modify portable directives or to replace source code with standard library calls. The extent of any modifications will factor into the evaluation of the projections – with fewer changes preferred. If any source code modifications are made they must be documented and provided back to CORAL under the same license as the original source code. The baseline estimates can include expected improvements to compilers, threading runtimes, MPI implementations and similar system software.

Beyond this, the Scalable Science and Throughput Benchmarks can be optimized as desired by the Offeror. Performance improvements from pragma-style guidance in C, C++, and Fortran source files are preferred. Wholesale algorithm changes or manual rewriting of loops that become strongly architecture specific are of less value. Modifications must be documented and provided back to CORAL.

The Offeror is encouraged to optimize the Data Science and Deep Learning Benchmarks using any reimplementation that provides scalability toward the size of the proposed system and brings out any novel deep learning capabilities of the proposed system. As with the other benchmarks any modifications to the Data Science and Deep Learning Benchmarks must be documented and provided back to CORAL.

We encourage the Offeror to report both their "as is" numbers and their optimized numbers to highlight the "as is" potential of their machine and the performance that can be achieved with various amounts of effort.

In partnership with the Laboratories, the successful Offeror(s) will continue efforts to improve the efficiency and scalability of the benchmarks between initial contract award and delivery of the system(s). The Offeror's goal in these improvement efforts is to emphasize higher level optimizations as well as compiler optimization technology improvements while maintaining readable and maintainable code, and avoiding vendor-specific or proprietary methodologies.

4.5 Reporting Guidelines

The Offeror may use benchmark results from existing systems to extrapolate and/or to estimate the benchmark performance on future proposed systems. CORAL provides two items to assist the Offeror in this task. First, CORAL has already run the benchmarks at scale on Sequoia, Mira and/or Titan. The results of these runs are provided on the benchmark website for the Offeror to use in their estimates of performance on the proposed CORAL system. Second, a "CORAL_Benchmark_Results" Excel spreadsheet is available on the benchmark website that should be used to report the results for all runs reported as part of the Offeror's proposal response. Each reported run must explicitly identify:

1) The hardware and system software configuration used;
2) The build and execution environment configuration used;
3) The source change configuration used; and
4) Any extrapolation and/or estimation procedures used.

4.6 Alternative Programming Models (TR-2)

The Laboratories are interested in task-based and other emerging programming models and expect some applications that run on the final system will use these programming models. (Example models include Charm++, DARMA and Legion.)

The Offeror should describe what hardware and software features of their proposed system would allow the acceleration of task-based runtimes. In addition, the Offeror should describe how they could partner with the Laboratories to port task-based models to the CORAL system.

5.0 CORAL Compute Partition

This section describes additional hardware and software requirements for the CORAL compute partition described by the Offeror as part of Section 3.1. If the proposed compute partition contains multiple compute node (CN) types (such as different nodes optimized for learning, throughput, and serial efficiency), then the Offeror will describe the available node types and explain how the Offeror decided on the number and mix of node types in the proposed configuration. If the compute partition has a mix of node types, then the relative cost of these node types will be provided. For each CN type, the Offeror will separately describe how each CN type meets the requirements listed in this section or, in the case that a particular CN type fails to meet a particular requirement in this section, why that choice is appropriate.

If individual CNs contain multiple processor types, e.g., CPU and accelerator, Offeror will describe how ALL processor types will meet the following requirements where applicable.

5.1 Compute Partition Hardware Requirements

5.1.1 Arithmetic Precision (TR-1)

The CN processor cores will have the ability to operate on 32-bit and 64-bit IEEE 754 floating-point numbers. Integer performance and other precision options of benefit to Data Science and Deep Learning should also be described.

5.1.2 Inter Core Communication (TR-1)

CN cores will provide atomic capabilities required to construct the usual higher level synchronizations (e.g., critical section or barrier). These capabilities will allow the construction of memory and execution synchronization that is extremely low latency. As the number of user threads can be large in a CORAL node, special hardware mechanisms will be provided that allow groups of threads to coordinate collectively at a cost comparable to the cost of a memory access. Multiple groups should be able to synchronize concurrently. Hardware support will be provided to allow for DMA to be coherent with the local node memory. These synchronization capabilities or their higher-level equivalents will be directly accessible from user programs.

The Offeror will specify the overhead, assuming no contention, of all supplied atomic instructions.

5.1.3 Hardware Support for Low Overhead Threads (TR-1)

The CNs will provide documented hardware mechanisms to spawn, to control and to terminate low overhead computational threads, including a low overhead locking mechanism and a highly efficient fetch and increment operation for memory consistency among the threads. The Offeror will fully describe these mechanisms; their limitations and the potential benefit to CORAL applications for exploiting OpenMP and/or POSIX thread parallelism within MPI processes.

5.1.4 Hardware Interrupt (TR-2)

CN hardware will support interrupting given subsets of cores based on conditions detected by the operating system or other cores within the subset executing the same user application.

5.1.5 Hardware Performance Monitors (TR-1)

The CNs will have hardware support for monitoring system performance. The hardware performance monitor (HPM) interface will be capable of separately counting hardware events generated by every thread executing on every core and accelerator in the node. The HPM will not only include those events that are generated from core-specific resources (e.g., L1 cache miss) but also those from the shared resources outside the cores (e.g., memory controller events). In particular, the Offeror's HPM will provide a rich set of events pertaining to the motion of data as it flows across the full memory hierarchy between all proposed computation units (e.g., CPU and accelerators). Further, the HPM will include hardware support for monitoring message passing performance and congestion on all node interconnect interfaces of all proposed networks.

The HPM will have 64b counters and the ability to notify the node OS of counter wrapping. The HPM will support setting of counter values, saving them and restoring them, as well as reading them in order to support sampling based on counter wrapping. The HPM will also support instruction-based sampling (e.g., Intel PEBS or AMD IBS) that tracks latencies or other data of interest in relation to specific instructions. All HPM data will be made available directly to system monitoring tools (see Section 8.0), and to application programmers and to code development tools (see Section 9.2.2.8) executed by non-privileged users.

5.1.6 Hardware Power and Energy Monitors and Control (TR-2)

CN hardware will support user-level monitoring and control of system power and energy. The Offeror will document a Hardware Power Monitor and Control Interface (HPMCI) that will use this hardware support to measure and to log (see Section 10.3) the total power of a node and to control its power consumption using 64b counters. HPMCI will support monitoring and control during idle periods as well as during active execution of user applications. HPMCI will provide user-level mechanisms to start and to stop all measurements. Nonblocking versions of all HPMCI measurement and control mechanisms will be available.

HPMCI will provide several power domains to isolate subsystem power measurement and control including, but not limited to, individual cores and processors, memory, and I/O subsystems, e.g., network. If HPMCI does not provide separate power domains per individual processor core then HPMCI will group cores into small subsets for monitoring and control.

5.1.7 Hardware Debugging Support (TR-1)

CN cores will have hardware support for debugging of user applications, and in particular, hardware that enables setting regular data watchpoints and breakpoints. The Offeror will fully describe the hardware

debugging facility and limitations. These hardware features will be made available directly to applications programmers in a documented API and utilized by the code development tools including the debugger.

5.1.8 Clearing Local NVRAM (TR-1)

If the CNs are configured with local NVRAM, then there will be a scalable mechanism to clear selective regions of the NVRAM so that it can be used in a secure computing environment. The clearing mechanism will not incur excessive wear of the media if the media's usage is limited by the expected operational lifetime of the system.

5.1.9 Support for Innovative Node Programming Models (TR-2)

The CNs will provide hardware support for innovative node programming models such as Deep Learning. The Offeror will fully describe these hardware facilities, along with limitations and potential benefit to CORAL applications for exploiting innovative programming models. Hardware facilities for node parallelism and Low Overhead Threads (see Section 5.1.3) will be combinable to allow the programming models to be nested within an application call stack.

5.1.10 Single Address Space between CPUs and Accelerators (TR-2)

If the proposed CN contains both CPUs and accelerators, then CORAL prefers a single address space between the CPUs and accelerators, which is a programming convenience that enables faster and wider adoption of accelerators. Ideally, the programming environment will provide methods for using accelerator memory as a scratchpad as well as accepting dynamically allocated pointers.

5.1.11 High-Bandwidth Connectivity to Accelerators (TR-2)

If the proposed CN contains both CPUs and accelerators, then CORAL prefers that the connectivity bandwidth between CPUs and accelerators matches or exceeds the bandwidth of the system memory.

5.2 Compute Partition Software Requirements

The CN Operating System (CNOS) will provide a reliable environment with low runtime overhead and OS noise to enable highly scalable MPI applications running on a large number of CNs with multiple styles of concurrency within each MPI process.

5.2.1 CNOS Supported System Calls (TR-1)

The CNOS will be Linux or will provide Linux-like APIs and behaviors. The Offeror will list deviation from Linux supported calls. The Offeror will propose a CNOS that extends Linux with specific system calls such as an API to measure memory consumption at runtime, to fetch node specific personality data (coordinates, etc.), or to determine MPI node rank mappings.

All file IO will have user configurable buffer lengths. CNOS will automatically flush all user buffers associated with a job upon normal completion or explicit call to "abort()" termination of the job. The CNOS will also have an API for application invoked flushing of all user buffers.

5.2.2 CNOS Execution Model (TR-1)

The CNOS will support the following application runtime/job launching requirements:

- Processes may be threaded and can dynamically load libraries via dlopen() and related library functions, such as dlmopen.

- All tasks on a single CN will be able to allocate memory regions dynamically that are addressable by all processes on that node. Allocation and de-allocation may be a collective operation among a subset of the processes on a CN.
- The MPI API will be supported for process-to-process communication within a job. Additional native packet transport libraries will be exposed.
- The Pthread interface will be supported and will allow pinning of threads to hardware. MPI calls will be permitted from each Pthread.
- OpenMP threading will be supported. MPI calls will be permitted in the serial regions between parallel regions. MPI calls will be supported in OpenMP parallel loops and regions, with possible restrictions necessitated by the semantics of MPI and OpenMP.
- A job may consist of a set of applications launched as a single MPMD (Multiple Program, Multiple Data) job on a specified number of CNs. The CNOS will support running each application on distinct sets of nodes as well as running multiple binaries on the same set of nodes using distinct subsets of cores within those nodes.
- The Offeror is only required to supply and to support one OS per node type, but the architecture should not preclude the booting of different node OSs. So a job may specify the CNOS kernel, or kernel version, to boot and to run the job on the CN.

5.2.3 Runtime Variability

Reproducible performance from run to run is a highly desired property of an HPC system, both from the standpoint of predictable behavior of a production system, and as a necessity when doing performance debugging of an individual application. Different metrics are warranted for these two use cases.

5.2.3.1 Production Workload Runtime Variability (TR-1)

The runtime variability of applications on the CORAL system will be less than 30% for all successful runs executed during the stability testing (ST) phase of acceptance. The applications used during the ST phase will execute problems that closely match a realistic workload. Target performance values for each performance run will be measured on a quiet system and in isolation during the performance testing (PT) phase of acceptance. For non-performance runs, the average runtime will be used as the reference value to assess the runtime variability during the ST phase.

5.2.3.2 Individual Application Runtime Variability (TR-1)

The runtime of each individual Scalable Science and Throughput benchmark on a dedicated system will not vary by more than 5% across executions. Offeror can use Quality of Service (QOS) techniques to guarantee consistent performance at the expense of maximum performance.

5.2.4 Preload Shared Library Mechanism (TR-1)

The Offeror will propose CNOS functionality equivalent to Linux Standards Base (LSB) 4.1 (or then current) LD_PRELOAD mechanism to preload shared libraries.

5.2.5 CNOS Python Support (TR-1)

The CNOS will support the launching and running of applications based on multiple languages, including both Python 2.7 and Python 3.6 (or then current versions) as released by http://www.python.org. Python applications may use dynamically linked libraries and SWIG (www.swig.org) and f2py generated

wrappers for the Python defined API for the ability to call C, C++ and Fortran library routines. The Offeror will provide optimized versions of the Python modules numpy, scipy, and mpi4py, tensorflow, and theano, and will track future releases.

5.2.6 Page Table Mappings and TLB Misses (TR-1)

The CNOS will have support for multiple page sizes and for very large pages (256 MB and up). The Offeror will propose CNOS and/or hardware techniques for page table mapping that minimize translation look-aside buffer (TLB) misses.

5.2.7 Guard Pages and Copy-On-Write Support (TR-2)

The CNOS will provide fine-grained guard page and copy-on-write support. When a guard page is accessed via read or write, the CNOS will raise a signal. The address of the instruction that caused the violation, along with other execution context, will be passed to any installed signal handler via the `siginfo_t` structure.

5.2.8 Scalable Dynamic Loading Support (TR-1)

The CNOS will support the scalable loading of dynamic libraries (object code) and scripts (interpreted code). Library loading for a dynamically linked application at scale (by exploiting the parallel file system) will be as fast or preferably faster than loading the same libraries during the startup of a sequential job on a single node. Likewise, loading libraries via dlopen() or similar runtime calls will be as fast or faster than loading the same libraries for a sequential job. Loading of scripts (*.py files) or compiled byte code (*.pyc files) will have equivalently scalable performance.

5.2.9 Scalable Shared Library Support (TR-1)

If a shared library is used by more than one process on a node, there will be one and only one copy of the library's code section resident in the node's memory.

5.2.10 Persistent, Shared Memory Regions (TR-1)

The CNOS will provide a mechanism to allocate multiple, persistent, shared memory regions, similar to System V shared memory segments. If the node contains NVRAM, there should be an option to allocate shared memory in NVRAM. All user-level threads may access a given region. The regions are released at the end of a job, but may persist beyond the life of the processes that created them, such that processes of a subsequent task in a job's workflow can attach to and access data written in them. The intended functionality is to allow for on-node storage of checkpoints and data for post-processing.

5.2.11 CNOS RAMdisk Support (TR-1)

The CNOS will provide a file system interface to a portion of the CN memory (i.e., a RAMdisk in CN memory or a region of CN memory accessible via the MMAP interface). The RAMdisk may be read and written from user applications with standard POSIX file I/O or MMAP functions using the RAMdisk mount point. The RAMdisk file system, files and data will survive application abnormal termination and thereby will permit a restarted application to read previously written files and data from the RAMdisk. The RAMdisk will be freed when the job ends.

5.2.12 Memory Interface to NVRAM (TR-1)

If NVRAM is present on the CN, either physically attached or logically presented through the I/O subsystem, then the Offeror will make it accessible via load/store memory semantics, either directly or

through memory mapped I/O. This interface will achieve within 85% of direct I/O performance for a read-only workload. With this interface, a write-only workload (with a cold cache) will achieve 95% of the performance achieved by a direct I/O read-modify-write workload as measured by the Livermore Random I/O Testbench (LRIOT). These tests will be run using a data set size that is at least twice the capacity of DRAM (including HBM).

Node failures may leave the data structures, allocated via the memory interface to the NVRAM, in an inconsistent/invalid state, causing the application to crash the next time it starts. Therefore, the memory interface will provide semantics to ensure consistency of the allocated memory in the face of failures.

5.2.13 Thread Location and Placement (TR-2)

The CNOS will provide a mechanism for controlling the placement and relocation of threads similar to the Linux `sched_setaffinity()` and `sched_getaffinity()` functions. The CNOS will also provide a mechanism for querying thread location.

5.2.14 Memory Utilization (TR-2)

The CNOS will provide a mechanism for a process or thread to gather information on memory usage and availability. This information will include current heap size, current stack size, heap high water mark, available heap memory, available stack memory and mapping of allocated memory across the address space.

5.2.15 Signals (TR-2)

The CNOS will provide POSIX compliance for signals and threads including: nested signals; and proper saving and restoring of signal mask.

5.2.16 Base Core File Generation (TR-1)

Upon abnormal program termination or by user intervention, the CNOS will support the generation of both full binary core files and lightweight core files (see Section 9.2.2.5). Controls will be provided to select sets of nodes or MPI processes permitted to dump core and which type of core file. The core file will contain the execution states of all threads used by the application including, but not limited to, the stack back-traces and register states of the threads running on the main CN processor as well as any CN accelerator(s) (i.e., all CN processors regardless of processor type). Lightweight core file generation will be scalable to the size of the machine. The provided controls will be used either before the job is executed (e.g., through environment variables or options to the job-launch program) or at run-time.

5.2.17 Stack-Heap Collision Detection (TR-1)

The CNOS will detect and trap stack-heap collisions for stack and heap memory. The maximum size of the main program stack will be controllable via *setrlimit(),* and stack overflow will be reported as a SIGSEGV signal. The heap and main stack will not overlap in the 64-bit virtual address space.

5.2.18 Thread Stack Overflow (TR-1)

The CNOS will detect thread stack overflow on the main CN processor as well as any CN accelerator(s) (i.e., on all CN processors regardless of processor type).

5.2.19 Software Support for Innovative Hardware Features (TR-1)

The CNOS will include any software support required to exploit any innovative hardware features that are part of Offeror's proposed solution. To the extent possible, standard programming models and APIs will be used to exploit such features.

6.0 Input/Output Subsystem

The evolution of node local and system local storage has led to the replacement of an optional storage system requirement in the original CORAL RFP with a TR-1 requirement to include the entire I/O subsystem in the RFP response.

ORNL and LLNL use their file systems in different ways that require different I/O subsystem requirements:

- The I/O subsystem for the 2021 system at ORNL will be a center-wide storage system, providing a global POSIX namespace that is accessible by both the CN partition as well as external resources in the facility;
- The I/O subsystem for the 2022 system at LLNL will not be a center-wide system but will support two types of namespaces: a global namespace and transient namespaces and will support two logical storage tiers, each optimized based on their performance and data residency requirements.

The 2021 System's I/O subsystem and the 2022 System's I/O subsystem requirements are described in separate sections to make it clear to Offerors what to respond to in each of the two proposals

The Input/Output (I/O) subsystem provides high-performance read/write storage for input data, output data, and defensive checkpoints. The I/O subsystem will provide the functionality traditionally associated with a high-performance parallel file system (e.g., Lustre or GPFS), as well as meeting the extreme performance requirements necessary to support checkpointing of an exascale class system. Considerable flexibility in the composition and placement of the I/O subsystem within the overall system architecture is allowed and encouraged.

I/O subsystem storage components are independent of, and will be provided in addition to, any memory supplied to meet CN requirements (see Section 5.2.12). If the storage is considered part of the system memory, it cannot be counted towards the I/O subsystem capacity. If the storage can be configured to be part of either the system memory or the I/O subsystem capacity then the Offeror will describe a base configuration that assigns fixed portions (not exceeding 100% in total) to those functions.

Given that the site locations and the years delivered are different for the ORNL exascale system proposal and the LLNL/ANL exascale system proposal, the IO Subsystem requirements differ and are described in Sections 6.1 and 6.2 respectively.

6.1 Requirements for the 2021 System's IO Subsystem

The I/O subsystem for the 2021 system at ORNL will be a center-wide storage system, providing a global POSIX namespace that is accessible by both the CN partition as well as external resources in the facility. The global namespace, encompassing all storage tiers, may employ caching techniques to meet performance requirements as long as data is available for application restart in the face of an I/O subsystem failure within a reasonable time without user intervention. All data written to the global namespace should be durable within 15 minutes.

The storage system, a shared resource, will provide simultaneous access to various clusters within the facility. The center-wide storage system will allow users to compile applications concurrently, and will allow simulation data written from the CN partition to be accessed from other clusters.

The storage system will be periodically purged. Efficient purging requires the I/O subsystem to provide high IOP levels for walking the file system and unlinking files. Overall system performance should not degrade while purging is in progress. The Offeror is encouraged to provide intelligent policy-driven and system-automated purge mechanisms as part of the I/O subsystem.

ORNL has analyzed where the largest I/O loads are on their system. The primary use cases are described in Appendix A. For most ORNL applications, the periodic scientific outputs to the I/O subsystem also serve as checkpoint/restart files. Most applications create one file per MPI process, write the data, and close the file. Other applications open one or more shared files instead of one file per process. In both the file per process and shared file models, data will be written to the global namespace. The application outputs will later be read on one or more machines connected to the center-wide file system. The I/O subsystem design will efficiently support both writing application results and reading input data at job launch. When the CN partition is going offline, the I/O subsystem must ensure that any cached data is migrated to the portion of the subsystem that will remain up for the external resources.

6.1.1 Design and Composition

6.1.1.1 Global POSIX Namespace (TR-1)

The I/O subsystem will provide a single, global POSIX namespace, encompassing all storage tiers, that provides the required capacity and performance. The global namespace will present a single mount point. Offeror will describe how data and metadata will be managed within the global namespace at all levels within the subsystem. Any application utilizing the POSIX API will not require any code modification to access the global POSIX namespace.

The Laboratory will accept relaxed POSIX semantics (see 6.1.5.1). Specifically, the Laboratory does not require support for Read After Write (RAW) between different processes on different nodes on the same open file.

6.1.1.2 High-Level Center-Wide Design (TR-1)

The Offeror will provide a high-level overview of the proposed center-wide I/O subsystem design, description of key capabilities, system composition, and performance, capacity, and reliability characteristics for each tier and for the overall center-wide system.

The Offeror will provide a description of the data flow and management including:

- within the I/O subsystem (e.g., between tiers);
- center-wide data flow and management, i.e., between the CORAL I/O subsystem and external computational systems; and
- in connection to data repositories, such as HPSS, when applicable.

6.1.1.3 Component-Level Design (TR-1)

The Offeror will describe all components of the I/O subsystem, including but not limited to, I/O servers, metadata servers, storage media, storage enclosures, storage controllers, I/O network topology, switches and network interfaces connecting I/O subsystem components, and associated hardware including racks. The Offeror will provide architectural diagrams, labeling all component elements and providing

bandwidth and latency characteristics of and between elements. The Offeror will specify the quantity of components that will be provided to meet the CORAL I/O subsystem capacity and performance requirements.

6.1.2 Capacity

6.1.2.1 Minimum I/O Subsystem Aggregate Capacity (TR-1)

The Offeror's I/O subsystem will provide a minimum of 100 times of the aggregate CN system memory size of uncompressed and usable capacity. The capacity excludes any overhead of data redundancy and reliability in the global namespace and will not include any storage used for hot-sparing.

6.1.2.2 Closest Tier Capacity (TR-1)

The I/O subsystem tier closest to the CNs will provide sufficient capacity to hold at least 100% of the aggregate CN partition memory size.

6.1.2.3 Maximum Single File Size (TR-1)

The I/O subsystem will support single file sizes equal to 100% of the aggregate CN memory size.

6.1.2.4 Metadata Capacity (TR-1)

The I/O subsystem will support the storage of at least 500 Billion files, at least 500 Billion directories, and at least 10 million files in a single directory.

6.1.3 Performance

6.1.3.1 Metadata Performance

6.1.3.1.1 Single Directory File Create Performance (TR-1)

The I/O subsystem will provide a sustained file and directory create rate of 50,000 per second in a single directory. This performance will be observed from the CN partition using sufficient CNs. The Offeror will describe the assumptions and required number of CNs.

6.1.3.1.2 Small File Size Transactions Performance (TR-1)

The I/O subsystem will provide an aggregate of 15 million transactions per second for parallel file create operations where 32 KiB is written into each file from the CN partition in parallel using a sufficient number of CNs. The Offeror will describe the assumptions and required number of CNs used to achieve this performance.

6.1.3.1.3 Single FEN Aggregate Transactions Performance (TR-1)

The I/O subsystem will sustain 1 million transactions per second, for parallel file create operations where 32 KiB is written into each file for a duration of 30 minutes. The Offeror will report the performance of open, create, stat, write, and close operations separately.

6.1.3.1.4 Metadata Performance Scalability (TR-2)

The I/O subsystem will support 10 concurrent instantiations of the metadata performance requirements specified in Section 6.1.3.1.3 without any degradation in performance. The Offeror will describe any assumptions made, e.g., how far apart in the directory tree the processes must be to meet this requirement.

6.1.3.1.5 Complete POSIX Namespace Tree Walk Performance (TR-1)

The I/O subsystem will walk the entire global namespace with 500 Billion directories and 500 Billion files in not more than 18 hours. The Offeror will describe all assumptions required to achieve this performance.

6.1.3.1.6 Sustained unlink() Performance (TR-1)

The I/O subsystem will perform a parallel unlink() operation of 100 million files in not more than 2,000 seconds (approx. ½ hour). The Offeror will describe all assumptions required to achieve this performance.

6.1.3.2 CN File I/O Performance

For the following CN file I/O performance requirements, the amount of data transferred is 20% of the aggregate memory of the CN partition unless stated otherwise.

6.1.3.2.1 File Per Process Performance Scaling (TR-1)

The Offeror will describe the best achievable file per process sequential and random performance for writes and reads scaling under ideal conditions. The time excludes the metadata operations. The Offeror will provide performance for each quintile up to the full machine size using 20% of the used CN memory.

6.1.3.2.2 File Per Process Checkpoint Performance (TR-1)

The I/O subsystem will perform a complete parallel write I/O phase each hour in at most 3 minutes. The time includes both the required metadata operations (file create, file open, and file close) and writing the data. The files will be created in a single directory. The Offeror will use the maximum number of MPI processes, used to achieve the Scalable Science Benchmark FOMs, spanning the entire CN partition to drive the I/O subsystem. The Offeror will describe the time to create the files, to write the data, and to close the files separately, as well as the aggregate time to complete all three, including the number of MPI processes and number of CNs.

6.1.3.2.3 File Per Process Availability for Restart (TR-1)

Within 15 minutes of the data being written as in Section 6.1.3.2.2, the I/O subsystem will ensure that data is available for a restart with one of the job's CN offline.

6.1.3.2.4 File Per Process Performance for Restart (TR-1)

The I/O subsystem will perform a complete parallel read I/O phase of the same size and at the same rate of the recently written data as in Section 6.1.3.2.2 using the same requirements with one of the job's CN offline. The time includes the required metadata operations (file open and file close) and reading the data.

6.1.3.2.5 Job Launch Read Performance (TR-1)

The I/O subsystem will perform a complete parallel read I/O phase at the start of a job within 10 minutes from the farthest tier from the CN. The time includes both the required metadata operations (file open and file close) and reading the data. The Offeror will describe the time to open the files, to read the data, and to close the files separately, as well as the aggregate time to complete all three.

6.1.3.2.6 Pre-Staged Read Performance (TR-2)

The I/O subsystem will perform a complete parallel read I/O phase at the start of a job within 3 minutes from the closest tier to the CN. The time includes the required metadata operations (file open and file close) and reading the data. The I/O subsystem may require assistance from the SRM to stage in the required data, but without requiring user intervention other than providing a list of directories or files to the SRM. The Offeror will describe the time for the application to open the files, to read the data, and to

close the files separately, as well as the aggregate time to complete all three. The Offeror will also describe the process to enable this capability and any requirements on the SRM and/or user.

6.1.3.2.7 Shared File Non-Overlapping Write Performance (TR-1)

The Offeror will describe the time to create a shared file, to write the data, and to close the file as well as the aggregate time to complete all three when MPI processes perform non-overlapping writes. Using the number of MPI processes and data set size from Section 6.1.3.2, the I/O subsystem will perform the I/O phase (open/create, write, close) each hour in at most 5 minutes.

6.1.3.2.8 Shared File Non-Overlapping Read Performance (TR-1)

The Offeror will describe the time to open a shared file, to read the data, and to close the file as well as the aggregate time to complete all three when MPI processes perform non-overlapping reads. Using the number of MPI processes and data set size from Section 6.1.3.2, the I/O subsystem will perform the I/O phase (open, read, close) each hour in at most 5 minutes.

6.1.3.3 Center-wide I/O Performance (TR-1)

The Offeror will provide a minimum of 3 TB/s of I/O performance from the I/O subsystem to center-wide systems (i.e., non-CORAL systems within the Laboratory data center) using the global namespace. This performance will be observed using sequential and random 4 MB writes and reads. This performance will be able to be exercised without degrading the I/O performance of the CN partition. The Offeror will describe at which tier of the I/O subsystem hierarchy this performance will be met.

6.1.3.4 Maximum Time to Durability (TR-1)

After an application writes the files of size described in Section 6.1.3.2 to the global namespace are closed, the data will be durable in at most 15 minutes.

6.1.3.5 Deterministic I/O Performance (TR-1)

The I/O subsystem design will deliver deterministic performance. Individual storage units within a tier will not vary in performance by more than 10%. Any internal I/O subsystem functions including disk scrubbing or parity reconstruction, but excluding garbage collection or any scheduled data management activities, such as data purging or migration between tiers, will not degrade the overall observed I/O performance of the I/O subsystem for the duration of any time that internal I/O functions are active.

6.1.3.6 Aged I/O Performance (TR-1)

I/O subsystem performance at 85% utilization should be the same as an empty (freshly formatted) I/O subsystem. The Offeror will describe how the I/O subsystem will achieve this performance at 85% utilization.

6.1.4 Metadata Services

6.1.4.1 Metadata Services Architecture (TR-1)

The Offeror will fully describe the architecture of the metadata service including detailed descriptions of all hardware and software components to be provided as well as scalability and resiliency attributes.

6.1.4.2 Intelligent Purge (TR-2)

The Offeror will describe any provided intelligent policy-driven and system-automated purge mechanisms as part of the I/O subsystem.

6.1.4.3 Policy-Based Data Migration Across Center (TR-2)

The Offeror will describe how the storage system's metadata management service can be combined with user (or administrator) specified policies, towards automated data migration to other center storage (e.g., hooks to move data to an archival system based on file aging).

6.1.4.4 Innovative Metadata Features (TR-2)

The Offeror will describe novel techniques employed to achieve scalable metadata namespace management, supporting up to the number of files and directories stated in Section 6.1.2.4. The Offeror will detail techniques for efficient indexing and searching of traditional file/directory metadata, large-scale parallel tree walk, fast or on-the-fly metadata summary for collections of files/directories. The Offeror will describe methods to tightly couple (or to combine) additional user-defined metadata tags with the files for allowing users to specify metadata tags defining entire files or collections of files (e.g., tenth checkpoint of a job run). The Offeror will explain scalable methods to store, to update, to index and to search through the metadata including performance and storage overhead. The metadata search strategies will include fast and efficient techniques to locate items of interest.

6.1.4.5 Server-Side Data Extraction (TR-3)

Based on the user-defined metadata, the Offeror will describe how other advanced features like server-side data extraction/reduction can be realized to minimize overall data movement (e.g., identify files of interest and extract relevant portions or variables from them) within the I/O subsystem. The Offeror will describe secure ways to run user code on storage servers or will provide trusted code to accomplish such common tasks.

6.1.5 Functionality

6.1.5.1 Close-to-Open Consistency (TR-2)

At a minimum, the I/O subsystem will provide file close-to-file open consistency. When an application process calls close() on a per-process file or when the last process calls close() on a shared file, the contents will be consistent when the file is next opened. The Offeror will describe the consistency semantics provided by the I/O subsystem.

6.1.5.2 Non-POSIX Interfaces (TR-3)

The Offeror's solution may present non-POSIX interfaces (e.g., KV store, object/S3, RDD, and MPI-IO) to the I/O subsystem, in addition to the POSIX interface. Alternative/relaxed POSIX semantics (e.g., eventual consistency, relaxation of sequential consistency) may also be provided. The Offeror will describe in detail all aspects of these alternate I/O APIs including the consistency model for the data. Any deviations or relaxations from the defined industry standards in the provided non-POSIX APIs should be documented by the Offeror.

6.1.5.3 Compression (TR-2)

The Offeror will describe compression technologies used by the I/O subsystem, the resources used to implement the compression/decompression algorithms, the expected compression rates, and all compression/decompression-related performance impact.

6.1.5.4 Encryption (TR-2)

The Offeror will describe encryption technologies used by the I/O subsystem, the resources used to implement the encryption/decryption algorithms, and all encryption/decryption-related performance

impact and management. If provided, the Offeror will describe data-in-flight and data-at-rest encryption/decryption capabilities separately. If the offered solution includes self-encrypting drives (SED) the Offeror will provide pricing deltas for equivalent non-SEDs.

6.1.5.5 Data Life Cycle Management (TR-2)

The Offeror will describe all aspects of the data life cycle management within the I/O subsystem, including a mechanism that allows data to move across the hierarchy without requiring explicit user input. The proposed solution will be described in detail, including the policy engines, policy declarations, global metadata catalogs and management, performance, and data orchestration.

6.1.5.6 Security Features (TR-1)

The Offeror will describe the I/O security features including authentication and authorization, for the POSIX interface as well as for any non-POSIX interfaces that are offered.

6.1.6 Reliability, Resiliency, Availability, Serviceability

6.1.6.1 Durability (TR-1)

At a minimum, the farthest tier of the I/O subsystem will be durable. At the Offeror's discretion, other tiers may be made durable. The collection of durable tiers will be referred to as the "durable" portion of the I/O subsystem. The Offeror will describe the durability model of the I/O subsystem at each tier.

6.1.6.2 Serviceability (TR-1)

The Offeror will provide a serviceable I/O subsystem, where all components will be field-replaceable. In addition, Offeror will describe the individual field replaceable units (FRUs), in terms of their impact to operation and integrity of the I/O subsystem. Description of FRUs should include, at a minimum, whether that FRU is hot-swappable, warm-swappable, or cold-swappable. Offeror will describe situations where maintenance of specific serviceable items presents a change to the resiliency or data integrity of the I/O system during that maintenance period.

6.1.6.3 Reliability (TR-1)

The durable portion of the I/O subsystem will be highly reliable and available. It will be tolerant to electrical and mechanical distribution (power and cooling) events. Tolerance to electrical power events may be mitigated through various techniques such as, but not limited to, the use of diverse power supplies or mechanisms that ensure that data can be drained from caches. Tolerance to mechanical power events will include extended operating ranges for air-cooled equipment where no supplemental air cooling is available.

The durable portion of the I/O subsystem will be designed with hardware and/or software redundancy schemes that ensure the ability to recover data written to the I/O subsystem under a double failure of the same component type. The I/O subsystem will serve center-wide clients when the CN partition is down. If data exists within caches inside of the CN partition, the I/O subsystem will provide a mechanism to drain the data to the portion of the I/O subsystem that remains accessible for center-wide clients. If data exists within caches inside of the CN partition, the I/O subsystem will provide a mechanism to drain the data to the portion of the I/O subsystem that remains accessible for center-wide clients. In case of storage media failures, the rebuild processes will be fully automated and initiated by the I/O subsystem.

The Offeror will describe the failure modes of the I/O subsystem including the caches, specifically single points of failure inherent in the design, as well as hardware and software features that protect against failures. The description will include both electrical and cooling mechanisms, and how faults or degraded

electrical and mechanical aspects might affect the I/O subsystem. The Offeror will fully describe all I/O subsystem reliability, availability, and integrity aspects including a description of all software and hardware redundancy schemes. The Offeror will describe failure modes that would lead to inability to recover data written to the I/O subsystem. For all components within the I/O subsystem, the Offeror will describe whether the component can serve center-wide clients when the CN partition is down. The Offeror will describe the mechanism to drain tiers from inside the CN partition and the duration to do so. The Offeror will describe any additional redundancy schemes available and any other reliability features provided by the storage subsystem.

6.1.6.4 Hot Spare Storage Media (TR-2)

The Offeror will describe any hot spare capabilities of the proposed solution and will indicate how many hot spare drives or drive-equivalents are included in the proposed configuration.

6.1.6.5 Storage Media Rebuild Tuning Capabilities (TR-2)

The Offeror will provide an I/O subsystem with the ability to specify the allocation of resources (e.g., CPU) to a storage media rebuild operation that allow the Laboratories to balance ongoing I/O requests with background rebuild requests. The Offeror will describe the storage media rebuild tuning capabilities of the proposed storage media subsystems. Rebuild time estimates will be provided for the different rebuild priority levels and rebuild scenarios (e.g., rebuilding data to a replacement disk or rebuilding to spare capacity).

6.1.6.6 Parity Check on Read (TR-1)

Storage hardware and software will perform parity checks or T10 Data Integrity Feature (DIF), or comparable data integrity checks on all data read. The proposed system will ensure that a parity mismatch either returns an error or spawns a transparent retry of the read. The Offeror will describe how and where data integrity checks are performed.

6.1.6.7 Data on Storage Media Verification (TR-1)

An offline fsck run will not exceed 24 hours to complete on the entire global namespace that is populated with 500 Billion files and 500 Billion directories. The Offeror will also describe the online “fsck” for correcting errors, its performance overhead, as well as the expected time to completion given the scale of the system. The Offeror will provide all other tools necessary to verify the consistency of data on storage media and to repair inconsistencies.

6.1.6.8 Power Loss Data Save and Non-volatility (TR-1)

The proposed I/O subsystem will preserve data in the closest tier to the CN, in the event of power loss. The data in the I/O subsystem once persisted should be truly non-volatile in the face of power loss.

6.1.6.9 Fast Storage Media Rebuild Mechanism (TR-1)

Failed storage media should be rebuilt with correct data and the data integrity and storage redundancy should be restored in at most 6 hours from the time that the failure is detected for the durable portion of the I/O subsystem. The Offeror will describe mechanisms for storage media rebuilds and will provide projected rebuild times. Faster storage media rebuild mechanisms are highly encouraged. The I/O subsystem will maintain 70% of the required bandwidth during rebuild.

6.1.6.10 No Single Point of Failure (TR-1)

The I/O subsystem will not possess any single points of failure. If this requirement is not technically feasible, the Offeror will clearly identify and describe any components that are single points of failure.

6.1.6.11 Failover Mechanism (TR-1)

If the I/O subsystem architecture includes shared hardware components, those components will be capable of automatic and manual failover without data corruption. The Offeror will describe how the proposed I/O subsystem will support failover.

6.1.6.12 Uniform Power Distribution (TR-1)

The I/O subsystem power distribution mechanisms will be described. All storage systems will provide uniform power distribution based on no less than two independent inputs. The I/O subsystem will continue to run in the presence of a failure of a single input.

6.1.6.13 Hot Swapping Support (TR-1)

The I/O subsystem will support hot swapping of field replaceable components with minimal impact to the operating state of the system for all components including power supplies, fans, controllers, storage media, cabling, host adapters, and drive enclosure bays. If hot swapping is not technically possible, the Offeror will identify and describe all components that do not support the hot swapping capability.

6.1.6.14 Resiliency and RAS Features (TR-1)

The Offeror will provide component level RAS characteristics that are exploited to achieve a high level of system resilience and data integrity and rapid recovery from hardware and software failures. The Offeror will describe the endurance characteristics of each storage media type used in the proposed I/O subsystem. In the use of non-volatile storage, the RAS system will track and report wear-leveling.

6.1.6.15 Recovery Time (TR-2)

The Offeror will design the I/O subsystem such that recovery times from component failures are limited to the greatest extent possible. The Offeror will provide estimates of the times required to recover from various I/O subsystem faults due to hardware and/or software. The Offeror will identify and describe any cascading failures.

6.1.6.16 System Mean Time To Interrupt (TR-2)

The Offeror will provide the System Mean Time To Interrupt (SMTTI) for the proposed I/O subsystem configuration and will describe the basis for computing this value.

6.1.6.17 System Mean Time Between Failure and System Mean Time Between Repair (TR-2)

The Offeror will specify the Mean Time Between Failure (MTBF) and Mean Time Between Repair (MTBR) for all major components of the proposed solution including storage media, servers, controllers, power supplies, switches and fans.

6.1.6.18 Mean Time To Data Loss (TR-1)

The Offeror will calculate the Mean Time To Data Loss for the entire I/O subsystem configuration and will detail any cases that delay data access or make the data in the I/O subsystem unavailable. The Offeror will describe the process used for this calculation.

6.1.6.19 Scheduled Availability (TR-1)

The I/O subsystem will demonstrate a scheduled availability level of not less than 99.5%. Unscheduled downtimes are defined as any period in which any data stored within the I/O subsystem is inaccessible or any period in which new data cannot be stored within the I/O subsystem.

6.1.6.20 Data Integrity (TR-1)

The I/O subsystem will have built-in mechanisms to protect against silent data corruption.

6.1.6.21 Visible Fault Lights (TR-3)

All I/O subsystem hardware components will have externally visible fault lights or displays to indicate the failure and location of the failure for any commonly replaced FRU.

6.1.7 Administration, Management, and Monitoring

6.1.7.1 Statistics, Log Collection, Presentation, and API (TR-1)

In the telemetry database (see Section 10.3), the Offeror will provide detailed metadata and file I/O statistics on a per job basis, system-level I/O statistics, and all I/O subsystem related logs, including, but not limited to, logs related to reliability, security, health monitoring, functionality and performance. All collected statistics and logs will be accessible through a well-documented API query interface.

6.1.7.2 Remote Administration (TR-1)

The Offeror will provide an I/O subsystem capable of being managed remotely by the Laboratory operations team through a secured protocol such as the SSH and/or HTTPS protocols. At a minimum, SSH version 2 will be supported.

6.1.7.3 Command Line Interface (CLI) Support (TR-1)

Storage components and services will be configurable and will be capable of being monitored via a command line interface suitable for scripting in a Linux environment. Configurations using encrypted transport mechanisms such as TLS are preferred. The Offeror will describe the security characteristics of the command line interfaces.

6.1.7.4 Simple Network Management Protocol (TR-1)

SNMP version 2c (https://tools.ietf.org/html/rfc1901) will be supported for routine read-only queries of system status information. The SNMP v2c implementation will support, with no impact to device performance, a full “snmpwalk” of all MIB objects at a minimum recurring interval of 10 seconds. SNMP version 3 (https://tools.ietf.org/html/rfc3410) will be supported for read-write access to the system.

6.1.7.5 Complex Password Support (TR-1)

The I/O subsystem and its authentication-protected components will support complex passwords, defined as passwords with 8 or more characters and that require digits and special characters.

6.1.7.6 Password Update Support (TR-1)

The Laboratories will have the ability to change passwords periodically on all authentication-protected I/O subsystem components, without requiring the assistance of the Offeror.

6.1.7.7 FRU Inventory Interface (TR-1)

The Offeror will provide a scalable mechanism to collect device inventory information, including device serial numbers, for all FRUs within each SSU.

6.1.7.8 Software/Firmware Update Requirement (TR-1)

The Offeror will provide an I/O subsystem with methods to perform storage component software and firmware updates in a non-disruptive manner. The Offeror will describe the software/firmware update process. The Offeror will identify and describe the worst-case time estimates for each software/firmware update process separately.

6.1.7.9 Network Boot (TR-1)

The I/O subsystem will support network boot. The subsystem server nodes will obtain an OS image and associated OS and kernel configuration files from a network attached image and configuration server. The Offeror will describe the expected boot procedure and time to boot the entire I/O subsystem from a cold start position.

6.1.7.10 Quota Management (TR-2)

The I/O subsystem will provide the ability to manage the storage capacity of the system by using soft and hard quotas. The I/O subsystem will provide accurate reporting of the quota capacity usage. Quotas will be manageable throughout all storage layers. The quota management will allow for project quotas, which are a third class in addition to the standard POSIX uid or gid.

6.1.8 Additional Requirements

6.1.8.1 Official Release Tracking (TR-2)

All provided I/O subsystem software will track subsequent official releases by a maximum of four months for the lifetime of the I/O subsystem. The Offeror will describe mechanisms for notifying the Laboratories of official releases.

6.1.8.2 External I/O Subsystem Client Support (TR-1)

The Offeror will provide I/O subsystem client software for 2,500 non-CORAL nodes for use by external Linux systems in Laboratory data centers. These non-CORAL clients will have equivalent functionality to CORAL clients and will include client software licenses for x86, ARM, and Power nodes.

6.1.8.2.1 Additional Licenses (TO-1)

Additional licenses for 2,500 nodes of x86, ARM, and Power architectures will also be priced as an option by the Offeror.

6.1.8.3 Modification/Reconfiguration Authority (TR-1)

The Laboratories will have root-level access and authority to install, to modify, and to reconfigure any version of the I/O subsystem software. The Laboratories will also have the same ability to install, to configure, and to operate necessary software to comply with the Laboratory site security plan. All aforementioned activities will not require the assistance or the permission of the Offeror.

6.1.8.4 Full-scale Test Support (TR-1)

The Offeror will propose a method for doing full-scale tests of new client and server software features and versions without disturbing data on the production file system.

6.1.8.5 Documentation Requirements (TR-1)

The Offeror will provide full documentation and implementation details of all network and communication protocols used by the I/O subsystem.

6.1.8.6 I/O Aware Scheduling (TR-2)

If I/O capacity and bandwidth reservation mechanisms are provided by the SRM (see Section 8.3.6.6), the Offeror will describe how the I/O subsystem and SRM will provide guaranteed space and time constraints. The Offeror will describe any additional, advanced I/O awareness mechanisms, such as locality awareness, and explain how they work in conjunction with job scheduling.

6.2 Requirements for the 2022 System's IO Subsystem

The envisioned I/O subsystem will be logically hierarchical, consisting of two logical tiers of storage, each optimized based on their performance and data residency requirements. The Offeror may propose any organization that meets these requirements: the I/O subsystem may consist of a single monolithic physical storage tier or it may contain multiple intermediate physical tiers as long as all requirements of the two logical tiers are satisfied. The *performance tier* will be designed to deliver high I/O rates needed to maximize overall system throughput. Data written to this tier may be deleted at the end of a job, may be cached for near-term access (e.g., application restart, in-situ analysis or post-processing) or may be migrated to a lower storage tier for longer term retention. The performance tier may contain node-local storage, shared storage or some combination of the two. The I/O subsystem will also include a *capacity tier* that provides cost-effective, long-term persistent storage. Data in the capacity tier may be staged up through the storage hierarchy to facilitate faster job launch.

The I/O subsystem will support two types of namespaces: a *global namespace* and *transient namespaces*. The I/O subsystem will provide exactly one global namespace, which will be accessed through a single mount point and will provide seamless integration of all storage tiers. The global namespace may use the performance tier as a caching layer, with data being migrated to/from the capacity tier transparently. Transient namespaces will exist solely in the performance tier. They may be instantiated on a per-job, per-CN or possibly an independent basis and will serve to insulate the global namespace from the overhead of managing data that may not ultimately need to persist long term. Performance tier storage will be provisioned, potentially dynamically, between the global namespace and transient namespaces.

Checkpoint/Restart is the highest priority use case for the I/O subsystem. During job execution, the application's processes will periodically output datasets to the I/O subsystem. While the term *checkpoint* is often assumed to refer to defensive I/O to aid in restart after a system or application failure, these restart files often also contain scientific results that the user will analyze later. For that reason, checkpoints should not be assumed to be disposable. Checkpoint data sets may be written to the global namespace or to a transient namespace, depending on Laboratory site configuration preferences and/or user requirements. Any data written to a transient namespace, including checkpoints, must be explicitly persisted to the global namespace by the user or application middleware in order to be preserved beyond the lifespan of the transient namespace. Data written to the global namespace will be persisted without user intervention.

The I/O subsystem will support other use cases, including pre-staging of large data sets to accelerate job launch and read caching of large data sets such as training data for machine learning applications. Additionally, in the 2021 timeframe, we expect that analysis of simulation results will often be performed in-situ, either on the same nodes that are running the simulation or another set of nodes that are part of the same job. Some workflows may involve multiple iterations of simulation and analysis. I/O solutions that facilitate cross-node sharing of data residing in the performance tier are desirable.

6.2.1 Design and Composition

6.2.1.1 High-Level Design Overview (TR-1)

The Offeror will provide a high-level overview of the proposed I/O subsystem design, including description of key capabilities and differentiating attributes, as well as system composition, i.e., number and placement of storage tiers within the overall CORAL system architecture.

If Offeror's solution consists of multiple physical storage tiers, Offeror will describe the interface between physical tiers and will indicate to what extent the physical tiers can function and/or be accessed independently of one another.

The Offeror will provide a description of the data flow including:

- within the I/O subsystem (e.g., between tiers);
- "off cluster" data flow, i.e., between the CORAL I/O subsystem and external computational systems within the Laboratory data center; and
- in connection to data repositories such as an external file system (e.g., GPFS or Lustre) or external tape archive, such as HPSS.

If Offeror's solution requires I/O forwarding, the Offeror will describe this mechanism including its data protection and data availability characteristics.

The Offeror will complete Table 6-1, describing the high level characteristics of each physical storage tier within the I/O subsystem. The Offeror may add or delete rows as required.

Table 6-1: Storage Tier Characteristics

Tier Name	Storage Media	Capacity (PiB)	Bandwidth (TiB/s)

6.2.1.2 Metadata Services Architecture (TR-1)

The Offeror will fully and separately describe the architectures of the metadata services for the performance and capacity logical tiers including descriptions of all hardware and software components to be provided as well as scalability and resiliency attributes.

6.2.1.3 Component Level Description (TR-1)

The Offeror will provide a scalable solution that utilizes unit building blocks that allow for modular expansion and deployment of the I/O subsystem. The Offeror will describe all components of the I/O subsystem, including but not limited to, I/O servers, metadata servers, storage media, storage enclosures, storage controllers, network switches and interfaces connecting I/O subsystem components, and racks. Resilience and data protection characteristics of all storage media will be described.

The Offeror will provide architectural diagrams of the proposed solution, labeling all component elements and showing capacity, bandwidth and latency characteristics of and between elements. In a tiered solution, each tier should be clearly distinguished. The Offeror will specify the quantity of components at each tier that will be provided to meet the CORAL I/O subsystem capacity and performance requirements.

6.2.2 I/O Subsystem Access Modes (TR-1)

The I/O subsystem will be accessible through two different interfaces: a global namespace and transient namespaces, described below. Offeror will describe mechanisms to partition physical storage between these interfaces. Partitioning mechanisms may be restricted to system boot time or may be dynamically accessible, possibly limited to job launch time or otherwise.

6.2.2.1 Global Namespace (TR-1)

The I/O subsystem will provide a single, global POSIX namespace that presents a single mount point on the CN partition and the FENs, encompassing all physical storage tiers (excluding storage provisioned as transient namespaces (see Section 6.2.2.2)). The global namespace will also be accessible from external systems within the Laboratory data center. Offeror will describe how data and metadata will be managed within the global namespace and the data coherency guarantees that are provided within and between all clients (i.e., CNs, FENs, and external systems) that access the global namespace.

6.2.2.2 Transient Namespaces (TR-1)

The I/O subsystem will provide the ability to provision transient namespaces within the performance tier to meet the capacity requirements specified in Section 6.2.3.2. Transient namespaces will support both file-per-process and shared file I/O models. Metadata operations performed against a transient namespace will not impact performance of the global namespace. Transient namespaces will present a POSIX interface (see Section 6.2.5.1) and may also provide non-POSIX interfaces (see Section 6.2.5.2).

Transient namespaces may be allocated on a per-node or per-job basis or possibly as an entirely independent resource. The I/O subsystem will provide a mechanism for migrating data to and from transient namespaces (see Sections 6.2.4.2.5 and 6.2.4.2.6). The I/O subsystem will also provide a mechanism to automate deletion of data from transient namespaces based on user-specified policies.

The Offeror will describe their implementation of transient namespaces. This description will include: unit of allocation (per-job, per-node, independent); aggregate bandwidth and capacity; and per-allocation bandwidth and capacity. The Offeror will describe the process to allocate, to access and to destroy transient namespaces and their lifespan, i.e., whether they are ephemeral (exist only for the lifetime of a job) or whether they can persist longer term, and how data can be drained from, and staged into a transient namespace. The Offeror will discuss how file-per-process and shared file I/O models are supported. The Offeror will also describe characteristics of the storage media including data protection capabilities, e.g., erasure coding or RAID.

6.2.3 Capacity

6.2.3.1 Aggregate Capacity (TR-1)

The I/O subsystem will provide a minimum of 50 times the aggregate CN system memory size (see Section 3.2.3) of uncompressed and usable, long-term storage capacity. This capacity excludes any performance tier storage used for transient namespaces or inclusive caching of the global namespace. It also excludes overhead of data redundancy and reliability, e.g., RAID or hot-sparing.

6.2.3.2 Performance Tier Capacity (TR-1)

The I/O subsystem performance tier will provide sufficient capacity to hold at least 4 data sets (checkpoints or application output) where the size of a data set equals 50% of the CN partition memory size (see Section 3.2.3). This capacity must be accessible through the global namespace as well as through transient namespaces.

6.2.3.3 Maximum Single File Size (TR-1)

The I/O subsystem will support single file sizes equal to 100% of the aggregate CN partition memory size.

6.2.3.4 Metadata Capacity (TR-1)

The global namespace metadata service will support the storage of at least 500 Billion files, at least 500 Billion directories, and at least 10 million files in a single directory.

6.2.4 Performance

6.2.4.1 Metadata Performance

All performance levels defined in this section will be achievable in the global namespace and transient namespaces separately, unless otherwise specified.

6.2.4.1.1 Object Insertion/Deletion/Retrieval Rates (TR-1)

Given a single directory, the I/O subsystem will demonstrate the following metadata rates:

- File create (zero length): 50K/second;
- File stat: 1M/second;
- File unlink: 50K/second;
- Directory create: 25K/second.

All rates will be sustained for at least 20 seconds. The Offeror will specify the number of CNs required to achieve this performance.

6.2.4.1.2 CN Partition Aggregate Transactions Performance (TR-1)

The I/O subsystem will demonstrate at least 15 million 32KiB file create transactions per second in aggregate from multiple CNs, sustained for at least 20 seconds. A file create transaction consists of the following metadata operations: open (create), stat, write, close. Each CN will create files in its own directory. The Offeror will specify the number of CNs required to achieve this performance. The performance of open, stat, write and close operations will be reported separately.

6.2.4.1.3 FEN Aggregate Transactions Performance (TR-1)

The I/O subsystem will demonstrate at least 1 million 32KiB file create transactions per second in aggregate from all FENs, sustained for at least 20 seconds. A file create transaction consists of the following metadata operations: open (create), stat, write, close. Each FEN will create files in its own directory. The Offeror will specify the number of FENs required to achieve this performance. The performance of open, stat, write, and close operations will be reported separately. This requirement only applies to the global namespace.

6.2.4.1.4 Metadata Performance Scalability (TR-2)

The I/O subsystem will support 10 concurrent instantiations of the metadata performance tests specified in Section 6.2.4.1.3 without any degradation in performance. The Offeror will describe any assumptions, e.g., the distance in the directory tree between processes, that are required to meet this requirement.

6.2.4.1.5 Complete POSIX Namespace Tree Walk Performance (TR-1)

The I/O subsystem will walk the entire global namespace with 500 Billion directories and 500 Billion files in not more than 18 hours. The Offeror will describe all assumptions required to achieve this performance.

6.2.4.2 CN File I/O Performance

All performance levels defined in this section will be achieved using an aggregate data set that is 50% of the aggregate CN partition memory size (see Section 3.2.3), running on all nodes in the CN partition and with the maximum number of MPI processes per node used to achieve the Scalable Science Benchmark FOMs. All performance levels will be achievable for both the global namespace and transient namespaces. The Offeror will describe any assumptions or limitations related to achieving these results.

6.2.4.2.1 File Per Process Write Performance (TR-1)

The I/O subsystem will perform a parallel write to the performance tier within 5 minutes when each process writes to a separate file. The time includes the required metadata operations (file create, file open, and file close). One file per MPI process will be created in a common directory. The Offeror will specify the time to create the files, to write the data, and to close the files separately, as well as the aggregate time to complete all three operations, including the number of MPI processes, the number of CNs, and the number of MPI processes per CN.

6.2.4.2.2 File Per Process Read Performance (TR-1)

The I/O subsystem will perform a parallel read from the performance tier of the data set written in Section 6.2.4.2.1 within 5 minutes. The Offeror will specify the time to open the files, to read the data, and to close the files separately, as well as the aggregate time to complete all three operations, including the number of MPI processes, number of CNs, and number of MPI processes per CN.

6.2.4.2.3 Shared File Write Performance (TR-1)

The I/O subsystem will perform a parallel write to the performance tier within 5 minutes when the processes perform non-overlapping writes to a single shared file. The Offeror will specify the time to create the shared file, to write the data, and to close the file separately as well as the aggregate time to complete all three operations. The Offeror will additionally specify the performance level that can be achieved in the case of overlapping writes and will describe how overlapping writes are handled, i.e., if the I/O subsystem automatically serializes competing writes or if the application must use file locks.

6.2.4.2.4 Shared File Read Performance (TR-1)

The I/O subsystem will perform a parallel read from the performance tier of the data set written in Section 6.2.4.2.3 within 5 minutes. The Offeror will describe the time to open the shared file, to read the data, and to close the file separately as well as the aggregate time to complete all three operations. The Offeror will specify the performance level that can be achieved in the case of overlapping reads and will describe how overlapping reads are handled.

6.2.4.2.5 Long-Term Data Persistence (TR-1)

The I/O subsystem will be capable of persisting a data set to the capacity tier within 20 minutes. This performance will be achievable for both the file-per-process data set described in Section 6.2.4.2.1 and the shared file described in Section 6.2.4.2.3. The time includes the required metadata operations (file open and file close) as well as the time to transfer the data. Data residing in the global namespace will be persisted automatically. For data residing in a transient namespace, the Offeror will describe the mechanism for persisting the data and to what extent this process is automatable and asynchronous.

6.2.4.2.6 Job Launch Read Performance (TR-1)

The I/O subsystem will perform a parallel read of a data set from the capacity tier to the CN partition within 20 minutes. This performance will be achievable for both the file-per-process data set described in

Section 6.2.4.2.1 and the shared file described in Section 6.2.4.2.3. The time includes the required metadata operations (file open and file close) as well as the time to transfer the data.

6.2.4.2.7 Asynchronous Data Stage-In (TR-2)

The I/O subsystem, in conjunction with the SRM, will provide a mechanism to asynchronously transfer a data set from the capacity tier to the performance tier within 25 minutes. The Offeror will describe the mechanisms for stage-in for both the global namespace and transient namespaces. It is desirable that such staging mechanisms not require user intervention other than providing a list of directories or files to be staged. Staging mechanisms will avoid stalling a job if one of the reserved nodes fails prior to job launch and will not interfere with backfill scheduling other than possibly to require re-staging data if the CN allocation changes as a result of a backfill scheduling decision.

6.2.4.2.8 Uniform File I/O Performance (TR-1)

I/O performance will be within 5% across all nodes within the CN partition. Performance will be measured by running single node instances of the IOR benchmark across the entire CN partition. Both read and write performance will fall within the 5% variability window when accessing the I/O subsystem through both the global namespace and transient namespaces.

6.2.4.2.9 Compute Partition Scalable I/O Performance (TR-1)

The Offeror will describe how performance of the performance tier can scale as the size of the CN partition is increased or decreased.

6.2.4.3 External I/O Bandwidth (TR-1)

The I/O subsystem global namespace will be accessible from external systems (i.e., non-CORAL systems within the Laboratory data center) using Offeror-provided client licenses (see Section 6.2.8.2). Write bandwidth will be no less than 1/3 the bandwidth required to provide data persistence as described in Section 6.2.4.2.5. Read bandwidth will be no less than 1/3 the bandwidth required to meet data stage-in performance in Section 6.2.4.2.6. This performance will be observed using sequential and random 4 MB writes and reads. The Offeror will describe the network path between the I/O subsystem and external systems, e.g., gateway nodes or SAN extension. In a physically tiered solution, the Offeror will fully describe the data paths involved to access data in the I/O subsystem externally. The Offeror will describe data coherence guarantees between internal and external data accesses.

6.2.4.4 Deterministic I/O Subsystem Performance (TR-1)

The I/O subsystem will deliver deterministic performance. Assuming a design consisting of modular building blocks as described in Section 6.2.1.3, individual building blocks will not vary in performance by more than 5%. Any internal I/O subsystem functions, including disk scrubbing or parity reconstruction, will not degrade the observed performance of the I/O subsystem. I/O subsystem performance will not degrade by more than 5% over the warrantied life of the system.

6.2.4.5 Aged File I/O Performance (TR-1)

An I/O subsystem with 85% utilization of aggregate capacity (see Section 6.2.3.1) will provide the same performance as an empty (freshly formatted) system.

6.2.5 Functionality

6.2.5.1 POSIX Interface (TR-1)

The Offeror's solution will present a POSIX interface to the I/O subsystem. The Offeror will describe any deviations from POSIX semantics. Transient namespaces may implement relaxed POSIX semantics.

6.2.5.2 Non-POSIX Interfaces (TR-2)

The Offeror's solution may present non-POSIX interfaces (e.g., KV store, object/S3, RDD, and MPI-IO) to the I/O subsystem. These interfaces may be available in the global namespace and/or transient namespaces. The Offeror will describe in detail all aspects of these alternate I/O APIs including the consistency model for the data. Any deviations or relaxations from the defined industry standards in the provided non-POSIX APIs will be described.

6.2.5.3 Close-to-Open Consistency (TR-2)

The I/O subsystem's POSIX interface will provide file close-to-open consistency. When an application process calls close() on a per-process file or when the last process calls close() on a shared file, the contents will be consistent when the file is next opened. The Offeror will describe the consistency semantics that the I/O subsystem provides.

6.2.5.4 Security Features (TR-1)

The Offeror will describe the I/O security features including authentication and authorization, for the POSIX interface as well as for any non-POSIX interfaces that are offered.

6.2.5.5 Quota Support (TR-1)

The I/O subsystem will support the ability to define and to manage quotas in the global namespace. It will be possible to apply quotas to, at minimum, individual users and groups and on both total number of files (i.e., inode quotas) and the total amount of data space consumed. The quota system will provide the ability to generate per-user and full system quota reports in a performant manner. The Offeror will describe the quota system in detail including supported quota types (hard, soft), action taken by the I/O subsystem when quota is reached, and how data compression (see Section 6.2.5.6), if implemented, impacts quotas.

6.2.5.6 Compression (TR-2)

The Offeror will describe compression technologies used by the I/O subsystem, the resources used to implement the compression/decompression algorithms, the expected compression rates, and all compression/decompression-related performance impact.

6.2.5.7 Encryption (TR-2)

The Offeror will describe encryption technologies used by the I/O subsystem, the resources used to implement the encryption/decryption algorithms, and all encryption/decryption-related performance impact. If provided, the Offeror will describe data-in-flight and data-at-rest encryption/decryption capabilities separately.

6.2.5.8 Application Jitter (TR-1)

If the Offeror's I/O subsystem includes CN-local storage, the application jitter induced due to another application's I/O operations (e.g., job A accessing storage located on a CN belonging to job B) should remain below 5% . The Offeror will describe the expected application jitter.

6.2.5.9 Network Contention (TR-1)

If Offeror's I/O subsystem uses the same network fabric as is used for application communication on the CN partition, the Offeror will describe network QoS or similar mechanisms that are used to arbitrate between I/O and application communication.

6.2.5.10 Innovative Metadata Features (TR-2)

The Offeror will describe novel techniques employed to achieve scalable metadata namespace management, supporting up to the number of files and directories stated in Section 6.2.3.4. The Offeror will detail techniques for efficient indexing and searching of traditional file/directory metadata, large-scale parallel tree walk, and fast or on-the-fly metadata summary for collections of files/directories. The Offeror will describe methods to couple (or to combine) additional user-defined metadata tags with files to allow users to specify metadata tags to define entire files or collections of files (e.g., tenth checkpoint of a job run). The Offeror will explain scalable methods to store, to update, to index and to search the metadata including performance and storage overhead. The metadata search strategies will include fast and efficient techniques to locate items of interest.

6.2.5.11 Policy-based External Data Migration (TR-2)

The Offeror will describe how the I/O subsystem's metadata management service can be combined with user (or administrator) specified policies, to automate data migration and/or replication to other storage tiers within the Laboratory's data center (e.g., hooks to move data to an archival system based on file aging).

6.2.5.12 I/O Aware Scheduling (TR-2)

If I/O capacity and bandwidth reservation mechanisms are provided by the SRM (see Sections 8.3.6.5 and 8.3.6.6), the Offeror will describe how the I/O subsystem and SRM will provide guaranteed space and time constraints. The Offeror will describe any additional, advanced I/O awareness mechanisms, such as locality awareness, and explain how they work in conjunction with job scheduling.

6.2.5.13 Server-Side Data Extraction (TR-3)

The Offeror will describe any capabilities within the I/O subsystem to perform server-side data extraction/reduction in order to minimize overall data movement (e.g., identify files of interest and extract relevant portions or variables from them) including mechanisms to execute user code securely on storage servers.

6.2.6 Reliability, Resiliency, Availability, Serviceability

6.2.6.1 Reliability (TR-1)

The Offeror will fully describe all aspects of the I/O subsystem that contribute to its reliability, availability, serviceability and data integrity, at both a full system level and component level. The Offeror will describe the failure modes of the I/O subsystem, specifically: single points of failure inherent in the design; hardware and software features that protect against failures; and failure modes that would lead to an inability to recover data. Endurance characteristics of each storage media type used in the proposed solution will be specified. In the case of storage media failures, the rebuild process will be fully automated and initiated by the I/O subsystem. The Offeror will describe recovery procedures for other failure modes.

The global namespace will be accessible to FENs and external clients when the CN partition is offline. Offeror will describe the consistency model for accesses to data existing in multiple storage tiers (e.g., cached) when one of those tiers is non-operational.

6.2.6.2 Hot Spare Storage Media (TR-1)

The Offeror will describe any hot spare capabilities of the proposed I/O subsystem and will indicate how many hot spare drives or drive-equivalents are included in the proposed configuration.

6.2.6.3 Storage Media Rebuild Tuning Capabilities (TR-2)

The Offeror will provide an I/O subsystem with the ability to specify the allocation of resources (e.g., CPU) to a storage media rebuild operation that allow the Laboratories to balance ongoing I/O requests with background rebuild requests. The Offeror will describe the storage media rebuild tuning capabilities of the proposed storage media subsystems. Rebuild time estimates will be provided for the different rebuild priority levels and rebuild scenarios (e.g., rebuilding data to a replacement disk or rebuilding to spare capacity).

6.2.6.4 Parity Check on Read (TR-1)

Storage hardware and software will perform parity checks or T10 Data Integrity Feature (DIF), or comparable data integrity checks on all data read. The proposed system will ensure that a parity mismatch either returns an error or spawns a transparent retry of the read. The Offeror will describe how and where data integrity checks are performed.

6.2.6.5 Data on Storage Media Verification (TR-1)

The Offeror will describe all tools that the storage subsystem provides to verify the consistency of data on storage media and all tools that are available to repair inconsistencies. The Offeror will also describe the online and offline "fsck" for correcting errors, its performance overhead, as well as expected time to completion as a function of the scale of the system. An offline fsck run will not exceed 24 hours to complete on the entire global namespace that is populated with 500 Billion files and 500 Billion directories.

6.2.6.6 Data Acknowledgement Guarantee (TR-1)

The provided I/O subsystem will guarantee that data resides on non-volatile or protected storage prior to completion of the issuing I/O command and returning an acknowledgment to the caller.

6.2.6.7 Power Loss Data Save and Non-volatility (TR-1)

The Offeror will describe how cached data is preserved in the event of power loss. The data in the I/O subsystem including any cached data once it is persisted should be truly non-volatile in the face of power loss; that is, the data written to the I/O subsystem can still be retrieved if a subsystem component fails or external power to the I/O subsystem is lost. If batteries are required to protect the data then the length of time that the batteries will provide protection will be detailed.

6.2.6.8 Fast Storage Media Rebuild Mechanism (TR-1)

Failed storage media will be rebuilt with correct data and the data integrity and storage redundancy will be restored in at most 6 hours from the time that the failure is detected. The Offeror will describe mechanisms for storage media rebuilds and provide projected rebuild times. Faster storage media rebuild mechanisms are highly desired.

6.2.6.9 No Single Point of Failure (TR-1)

The I/O subsystem will not possess any single points of failure. If this requirement is not technically feasible, the Offeror will clearly identify and describe any components that are single points of failure as well as mitigation and recovery procedures for failure of these components.

6.2.6.10 Failover Mechanism (TR-1)

If the I/O subsystem includes shared hardware components, those components will be capable of automatic and manual failover without data corruption. The Offeror will describe how the proposed I/O subsystem will support failover.

6.2.6.11 Uniform Power Distribution (TR-1)

The I/O subsystem power distribution mechanisms will be described. All storage systems will provide uniform power distribution based on no less than two independent inputs. The I/O subsystem will continue to run in the presence of a failure of a single input.

6.2.6.12 Storage Media Rebuild Performance (TR-1)

The I/O subsystem will maintain 70% of the required bandwidth in the presence of concurrent rebuilds or recovery operations (such as rebalancing data after replacing a failed storage media) on up to 10% of the available redundancy groups. These concurrent rebuild or recovery operations will require no greater than 12 hours to complete.

6.2.6.13 Hot Swapping Support (TR-1)

The I/O subsystem will support hot swapping of field replaceable components with minimal impact to the operating state of the system for all components including power supplies, fans, controllers, storage media, cabling, host adapters, and drive enclosure bays. The Offeror will identify any components that do not support the hot swapping capability.

6.2.6.14 Recovery Time (TR-1)

The Offeror will provide estimates of the time required to recover from various I/O subsystem faults due to hardware and/or software. The Offeror will identify and describe any cascading failures.

6.2.6.15 System Mean Time Between Application Interrupt (TR-1)

The Offeror will provide the System Mean Time Between Application Interrupt (SMTBAI) for the proposed I/O subsystem configuration and will describe the basis for computing this value.

6.2.6.16 System Mean Time Between Failure and System Mean Time Between Repair (TR-1)

The Offeror will specify the Mean Time Between Failure (MTBF) and Mean Time Between Repair (MTBR) for all major components of the proposed solution including storage media, servers, controllers, power supplies, switches and fans.

6.2.6.17 Mean Time To Data Loss (TR-1)

The Offeror will calculate the Mean Time To Data Loss for the entire I/O subsystem configuration and will detail any cases that delay data access or make the data in the I/O subsystem unavailable. The Offeror will describe the process used for this calculation.

6.2.6.18 Scheduled Availability (TR-1)

The I/O subsystem will demonstrate a scheduled availability level of not less than 99.5%. The Offeror will describe any scheduled maintenance periods required to attain this availability level. Unscheduled downtimes are defined as any period in which any data stored within the I/O subsystem is inaccessible or any period in which new data cannot be stored within the I/O subsystem.

6.2.6.19 Data Integrity (TR-1)

The I/O subsystem will have built-in mechanisms to protect against silent data corruption.

6.2.6.20 Visible Fault Lights (TR-3)

All I/O subsystem hardware components will have externally visible fault lights or displays to indicate the failure and location of the failure for any commonly replaced FRU.

6.2.7 Administration, Management and Monitoring

6.2.7.1 Statistics, Log Collection, Presentation, and API (TR-1)

The I/O subsystem will collect detailed metadata and file I/O statistics and will present these statistics on a per job basis. Additionally, system-level I/O statistics will be collected and presented. All I/O subsystem related logs will be collected and presented, including but not limited, to logs related to reliability, security, health monitoring, functionality and performance. All collected statistics and logs will be accessible through a well-documented API query interface.

6.2.7.2 Remote Administration (TR-1)

The I/O subsystem will be capable of being managed remotely by the Laboratory operations team through a secured protocol such as the SSH and/or HTTPS protocols. At a minimum, SSH version 2 will be supported.

6.2.7.3 CLI Support (TR-1)

Storage components and services will be configurable and will be capable of being monitored via a command line interface suitable for scripting in a Linux environment. Configurations using encrypted transport mechanisms such as TLS are preferred. The Offeror will describe the security characteristics of the command line interfaces.

6.2.7.4 Simple Network Management Protocol (TR-1)

SNMP version 2c or later will be supported for routine read-only queries of system status information. The SNMP implementation will support, with no impact to device performance, a full “snmpwalk” of all MIB objects at a minimum recurring interval of 10 seconds. The SNMP implementation will support, but not require, authentication and encryption for read-write access.

6.2.7.5 Complex Password Support (TR-1)

The I/O subsystem and its authentication-protected components will support complex passwords, defined as passwords with 8 or more characters and that require digits and special characters.

6.2.7.6 Password Update Support (TR-1)

The Laboratories will have the ability to change passwords periodically on all authentication-protected I/O subsystem components, without requiring the assistance of the Offeror.

6.2.7.7 FRU Inventory Interface (TR-1)

The I/O subsystem will provide a scalable mechanism to collect device inventory information, including device serial numbers, for all FRUs within the I/O subsystem.

6.2.7.8 Software/Firmware Update Requirement (TR-1)

The I/O subsystem will provide a mechanism to perform storage component software and firmware updates in a non-disruptive manner. The Offeror will describe the software/firmware update process. The

Offeror will separately identify and describe the worst-case estimates of the time to perform each software/firmware update process.

6.2.7.9 Network Boot (TR-1)

The I/O subsystem will support network boot. The subsystem server nodes will obtain an OS image and associated OS and kernel configuration files from a network attached image and configuration server. The Offeror will describe the expected boot procedure and time to boot the entire I/O subsystem from a cold start position.

6.2.8 Additional Requirements

6.2.8.1 Official Release Tracking (TR-2)

All provided I/O subsystem software will track subsequent official releases by a maximum of four months for the lifetime of the I/O subsystem. The Offeror will describe mechanisms for notifying the Laboratories of official releases.

6.2.8.2 External I/O Subsystem Client Licenses (TR-1)

The Offeror will provide 1000 licenses for I/O subsystem client software to be used on external Linux systems in Laboratory data centers. These non-CORAL clients will have equivalent functionality to CORAL clients and will support x86, ARM, and Power architectures.

6.2.8.3 Additional External I/O Subsystem Client Licenses (TO-1)

The Offeror will provide pricing in increments of 1000 for additional I/O subsystem client software licenses that meet the requirements in Section 6.2.8.2.

6.2.8.4 Modification/Reconfiguration Authority (TR-1)

The Laboratories will have root-level access and authority to install, to modify, and to reconfigure any version of the I/O subsystem software. The Laboratories will also have the same ability to install, to configure, and to operate necessary software to comply with the Laboratory site security plan. All aforementioned activities will not require the assistance or the permission of the Offeror.

6.2.8.5 Full-scale Test Support (TR-1)

The Offeror will propose a method for doing full-scale tests of new client and server software features and versions without disturbing data on the production file system.

6.2.8.6 Documentation Requirements (TR-1)

The Offeror will provide full documentation and implementation details of all network and communication protocols used by the I/O subsystem.

7.0 CORAL High Performance Interconnect (TR-1)

The Offeror will provide a high-level description of the system interconnect's topology, latencies, bandwidths, bi-section bandwidth, routing algorithm, and congestion mitigation techniques. If the system provides low-power (e.g., link idling) or degraded-operating modes, Offeror will describe them.

7.1 High Performance Interconnect Hardware Requirements

7.1.1 Node Interconnect Interface (TR-1)

The Offeror will provide a physical network or networks for high-performance intra-application communication within the CORAL system. The Offeror will configure each node in the system with one or more high speed, high messaging rate interconnect interfaces. This (these) interface(s) will allow all compute elements in the system simultaneously to communicate synchronously or asynchronously with the high speed interconnect. The CORAL interconnect will enable low-latency communication for one- and two-sided paradigms.

7.1.2 Interconnect Hardware Bit Error Rate (TR-1)

The CORAL full system Bit Error Rate (BER) for non-recovered errors in the CN interconnect will be less than 1 bit per month for the entire system. This error rate applies to errors that are not automatically corrected through Link-Level Retry or Forward Error Correction (FEC), if used, or ECC or CRC checks with automatic resends. Any loss in bandwidth associated with the FEC or resends would reduce the sustained interconnect bandwidth and is accounted for in sustained bandwidth for the CORAL interconnect. The system MTBF calculation will account for network errors as well.

7.2 Communication/Computation Overlap (TR-2)

The Offeror will provide both hardware and software support for effective computation and communication overlap for both point-to-point operations and collective operations, i.e., the ability of the interconnect subsystem to progress outstanding communication requests in the background of the main computation thread.

7.2.1 Low-Power Overlap (TR-2)

The Offeror will describe overlap capabilities when in low power mode (e.g., hardware offloaded communications or communication-specific low power core). Describe any latency impacts when using low-power mode as well as power savings.

7.3 Programming Models Requirements

7.3.1 Low-level Network Communication API (TR-1)

The Offeror will provide and fully support the necessary system software to enable a rich set of programming models (not just MPI) as well as capability for tools that need to communicate within the compute partition and to other devices in the system (e.g., nodes connected to the storage network). This requirement can be met in a variety of ways, but the preferred one is for the Offeror to provide a lower-level communication API that supports a rich set of functionality, including Remote Memory Access (RMA) and a Scalable Messaging Service (SMS) such as libfabric, UCX, or Portals.

The lower-level communication API (LLCA) will provide the necessary functionality to support complete implementations of GA/ARMCI (http://hpc.pnl.gov/globalarrays/index.shtml), Charm++ (http://charm.cs.uiuc.edu/software), GASNet (http://gasnet.cs.berkeley.edu), OpenSHMEM (http://openshmem.org/), and Legion (http://legion.stanford.edu), which are collectively called “Other Programming Models” (OPMs). The LLCA will also support distributed tools (DTs) that communicate across one or more networks and may need to communicate and/or synchronize with the application processes but that may have different lifetimes (i.e., are initialized and terminated independently of the compute partition and applications running therein). One example of a DT is MRNet

(http://www.paradyn.org/mrnet/). MPI, OPMs and DTs are direct users of the LLCA that are collectively called Programming Models (PMs).

Any application using multiple PMs will be able to use the LLCA directly in a robust and performant way. In particular, the LLCA will support simultaneous use of multiple PMs without additional programming overhead relative to their independent usage. For example, an application that uses one PM will be able to call a library that uses another PM and will run correctly without any code changes to the application or the library with respect to PM initialization or use. Also, no PM will be able to monopolize network resources such that the other cannot function, although proportional performance degradation may occur when hardware resources are shared. Disproportionate performance degradation - meaning that the summed performance of N PMs is significantly less than the performance of one PM – will not occur. Application failure due to one PM monopolizing network resources (including registered/pinned/RMA-aware memory segments) will not occur.

The specific features required of the LLCA include the following.

7.3.1.1 Scalable Messaging Service (TR-1)

The LLCA's Scalable Messaging Service (SMS) will provide reliable, point-to-point, small, asynchronous message communication. The Offeror may limit the maximum message size, but that limit will not be smaller than 128 bytes and the application will be able to query the limit.

7.3.1.2 Remote Memory Access

7.3.1.2.1 Asynchronous Progress on RMA Operations (TR-1)

Remote writes (puts) will complete in a timely fashion without any application activity on the remote process (i.e., they will be one-sided).

7.3.1.2.2 Registration of Memory (TR-1)

If registered memory is required for RMA communication, then the LLCA will expose this via registration and deregistration calls. Such calls will be local (i.e., non-collective).

Registration and deregistration of memory will be fast. The time required to register and to deregister a segment of memory will be less than the time required to communicate the same segment of memory to a remote endpoint once it is registered.

If memory registration is not required, the Offeror will describe the process to access memory remotely.

7.3.1.2.3 Support for Contiguous and Noncontiguous One-Sided Put and Get Operations (TR-1)

Contiguous and noncontiguous one-sided put and get operations will be provided. The noncontiguous support will support the transfer of a vector of contiguous segments of arbitrary length.

7.3.1.2.4 Hardware-Based Scatter-Gather Engines for Noncontiguous Operations (TR-3)

Non-contiguous one-sided put and get operations will use hardware scatter-gather to provide the highest possible bandwidth for messages where the contiguous message size is smaller than the packet size.

7.3.1.2.5 RMA Operation Message Sizes (TR-1)

RMA operations will support messages as small as 1 byte; however, the best performance is only expected for 8-byte messages and larger.

7.3.1.2.6 Remote Atomic Operations (TR-1)

Remote atomic operations on 32-bit and 64-bit integers will be supported. Atomic operations required include {add,or,xor,and,max,min}, fetch-and-{add,or,xor,and,max,min} as well as swap and compare-and-swap. The Offeror will describe all supported atomic operations.

7.3.1.2.7 Atomic Operations on Floating-Point Operations (TR-3)

Atomic operations on 32-bit and 64-bit floating-point operations will be supported.

7.3.1.2.8 Unaligned RMA Operations (TR-3)

The LLCA will support unaligned RMA operations, including unaligned source and sink addresses as well as lengths.

7.3.1.2.9 Symmetric Memory Allocation (TR-1)

The LLCA will support – possibly in collaboration with the OS and/or other runtime libraries – symmetric memory allocation such that RMA can be performed to all network endpoints without the storage of O(number of endpoints) of metadata.

7.3.1.2.10 Remote Completion Notification for RMA Operations (TR-1)

The LLCA will support the ability to request an optional remote completion notification for RMA operations. When requested, the LLCA will guarantee that the remote notification is not triggered until the RMA operation is complete. The notification may be delivered using the SMS service, for example, or from a separate method.

7.3.1.2.11 Scalable State and Metadata (TR-1)

The LLCA will have scalable state and metadata. Internal state for the LLCA that scales with the number of nodes or cores must be kept to a minimum.

7.3.1.2.12 Point-wise Ordering of RMA Operations (TR-1)

The LLCA will provide a mechanism to cause point-wise ordering of RMA operations that is not the default mode of operation.

7.3.1.3 Reentrant Calls (TR-1)

Multithreaded use of the LLCA will be supported. Reentrant LLCA calls will be supported, at least as an option. It will be possible for multiple threads to issue communication operations via the LLCA at the same time without mutual exclusion, provided that they use disjoint resources. Similarly, in the event-driven messaging service, multiple threads will be permitted to process events at the same time.

7.3.1.4 Accelerator-Initiated/Targeted Operations (TR-2)

If the system has multiple processor types (i.e., accelerators or coprocessors), each processor type will be able to initiate LLCA operations without explicit host activity and the LLCA will support RMA communication to each processor type's memory without explicit activity by the remote node host.

7.3.1.5 Support for Inter-job Communication (TR-1)

In order to support pipelined workflows between distinct jobs such as provided by ADIOS (http://www.olcf.ornl.gov/center-projects/adios/), the LLCA will provide mechanisms to implement policies to restrict and/or to allow separate jobs running in the same compute partition to intercommunicate. These mechanisms will be adjustable by the CORAL site operators.

7.3.1.6 Fault Isolation and Fault Tolerance (TR-1)

The LLCA will support a mode that does not abort a job upon errors, even fatal errors in a process or in a link of the network. The LLCA will return useful error codes that enable an application or distributed tool to continue operating. If possible, the LLCA will permit the re-establishment of communication with processes that have failed and have been restarted. The LLCA will guarantee that any new instance of the process does not receive messages sent to the failed instance and is not the target of an RMA operation intended for the failed instance.

The LLCA will be able to route around failed links automatically, provided at least one path on the network between two communicating processes remains available. The LLCA will be able to reintegrate failed links once they again become available.

7.3.1.7 Support for Non-compute Nodes (TR-1)

In order to support services and tools such as user-space I/O forwarding layers, profilers, event trace generators and debuggers, the LLCA will support RMA and SMS to nodes outside of the compute partition (e.g., FENs).

7.3.1.8 Dynamic Connection Support (TR-2)

The LLCA will provide a rendezvous mechanism to establish communication using client/server semantics similar to connect/accept. The communication may be in-band (i.e., using native LLCA primitives) or out-of-band (e.g., using sockets).

7.3.1.9 Documentation (TR-1)

Documentation of the LLCA will be thorough and contain example code for all API calls. The documentation or example code will not be proprietary in order to permit third-party software developers to support the LLCA. Vendors are encouraged, but not required, to continue supporting existing LLCAs in order to enable a smooth transition to the CORAL systems.

7.3.1.10 MPI Hardware Acceleration (TR-1)

The Offeror will provide a detailed description of the interconnect's capabilities to accelerate MPI operations in hardware. This may include such features as collective offloads, hardware tag matching, and hardware endpoint features.

7.3.1.11 Network Partitioning and Variability Reduction (TR-1)

The Offeror will describe any capabilities to partition the network in order to reduce inter-job contention on the interconnect. If partitioning is provided, the Offeror will describe how the network can be partitioned, the number of concurrent partitions supported, the time to setup/configure a partition, and any latency or bandwidth impacts of partitioning. Offeror will further describe any interconnect features and configuration options that reduce interconnect performance variability.

7.4 Quality of Service/Message Classes (TR-2)

The Offeror’s interconnect will provide QoS capabilities (e.g., in the form of virtual channels) that can be used to prevent core communication traffic from interfering with other classes of communication such as debugging and performance tools or with I/O traffic. Additional virtual channels for efficient adaptive routing may also be specified as well as a capability to prevent different application traffic from interfering with each other (either through QoS capabilities or appropriate job partitioning).

7.4.1 Small Message Tail Latency (TR-3)

The Offeror's interconnect will provide capabilities to avoid long tail latencies. The Offeror will describe the small message (e.g., 8 byte) latency distribution for an application using 20% of the CN partition performing periodic nearest neighbor communication, in which the MPI processes are placed randomly throughout the allocated CNs (i.e., not tightly packed), in the presence of a competing application messaging and I/O patterns, including, for example, FFTs, large point-to-point, and incast.

7.5 Counters (TR-2)

The Offeror's interconnect will provide real-time telemetry, FRU, and RAS metrics. The Offeror will describe what metrics are accessible and describe any provided software tools for collating and analyzing the data. Telemetry data will be user-level accessible.

7.5.1 Profiling support for Interconnect (TR-3)

The Offeror will provide access to the telemetry data for a job from the application processes, to enable profiling support for the network. This support will include switch and/or router data, congestion state, throttling, and latency information for select packets traversing the network.

If the Offeror supports offloading of collectives to the network, the profiling capabilities will include the offloaded operations.

7.6 Network Models (TR-3)

The Offeror will provide coarse grained network models (preferably SST; Codes acceptable), with response to aid in proposal evaluation. Models will include topology, reasonable approximations of routing algorithms, and congestion avoidance. Models will be used to evaluate single and multi-application workloads to aid in the selection process. Models will be able to simulate traces formats from DUMPI, Darshan, and OTF2.

8.0 Base Operating System, Middleware and System Resource Management

8.1 Base Operating System Requirements (TR-1)

The Offeror will provide on CORAL Front End Environment (FEE) and System Management Nodes (SMN) a standard multiuser Linux Standards Base specification V4.1 or then current (http://www.linux-foundation.org/collaborate/workgroups/lsb) compliant interactive base operating system (BOS). The BOS will:

- Include the full feature set and packages available in a then-current standard Linux distribution;
- Utilize standard Linux packaging methods; all files in the distribution will be owned by a package and dependencies between packages will be enforced;
- Include source code for all base Linux packages from which the corresponding binary packages can be built in a reproducible fashion;
- Provide consistent APIs and ABIs within a major release of distribution (meaning a binary built on version X.1 will run on any other version X.n).

The BOS will trail the official distribution release by no more than eight months. Updates will continue throughout the life of the system, including both major and minor versions and be buildable from source by the Laboratories.

8.1.1 Kernel Debugging (TR-2)

The Linux kernel in the Offeror's BOS will function correctly when all common debugging options are enabled including those features that are enabled at compile time. Kdump (or equivalent) will work reliably and dumps will work over a network (preferred) or to local non-volatile storage. Crash (or other online and offline kernel debugger) will work reliably.

8.1.2 Networking Protocols (TR-1)

The Offeror's BOS will support the Open Group (C808) Networking Services (XNS) Issue 5.2 (http://www.opengroup.org/pubs/catalog/c808.htm) and include IETF standards-compliant versions of the following protocols: IPv4, IPv6, TCP/IP, UDP, NFSv3, NFSv4 and RIP.

8.1.3 Reliable System Logging (TR-1)

The Offeror's BOS will include standards-based system logging. The BOS will have the ability to log to local disk as well as to send log messages reliably to multiple remote systems. In case of network outages, the logging daemon should queue messages locally and deliver them remotely when network connectivity is restored.

8.1.4 Operating System Security

8.1.4.1 Authentication and Access Control (TR-1)

The Offeror's BOS will implement basic Linux authentication and authorization functions. All authentication-related actions will be logged including: logon and logoff; password changes; unsuccessful logon attempts; and blocking of a user along with the reason for blocking. User access will be denied after an administrator-configured number of unsuccessful logon attempts. All Offeror-supplied login utilities and authentication APIs will allow for replacement of the standard authentication mechanism with a site-specific pluggable authentication module (PAM).

8.1.4.2 Software Security Compliance (TR-2)

The BOS will be configurable to comply with industry standard best security configuration guidelines such as those from the Center for Internet Security (http://benchmarks.cisecurity.org).

8.2 Distributed Computing Middleware

The following requirements apply only to the CORAL system Front End Environment (FEE).

8.2.1 Kerberos (TR-1)

The Offeror will provide, but may not require, the Massachusetts Institute of Technology (MIT) Kerberos V5 reference implementation, Release 1.11 or then current, client software.

8.2.2 LDAP Client (TR-1)

The Offeror will provide LDAP version 3, or then current, client software, including support for SASL/GSSAPI, SSL and Kerberos V5. The supplied LDAP command-line utilities and client libraries will be fully interoperable with an OpenLDAP Release 2.4 or later LDAP server.

8.2.3 Cluster Wide Service Security (TR-1)

All system services including debugging, performance monitoring, event tracing, resource management and control will support interfacing with the BOS PAM (Section 8.1.4.1) function. This protocol will be efficient and scalable so that the authentication and authorization step for any size job launch is less than 5% of the total job launch time.

8.2.4 Grid Security Infrastructure (TR-2)

The Offeror will provide in place of or addition to Kerberos, GSI (Grid Security Infrastructure) compatible authentication and security mechanisms including the use of X.509 certificates.

8.3 System Resource Management (SRM) (TR-1)

System resource management (SRM) is integral to the efficient functioning of the CORAL system. The CORAL system poses new SRM challenges due to its extreme scale, the diversity of resources that must be managed, and evolving workload and tool requirements. The Offeror will provide SRM in an integrated system software design that meets these challenges and results in a highly productive system that seamlessly leverages the CORAL system's advanced architectural features.

8.3.1 Open Source Offeror-Provided SRM Software (TR-1)

The Offeror will provide an open source SRM or work with 3rd party vendor(s) to provide an open source implementation. The SRM software will be provided as an installable binary in the BOS native packaging format as well as buildable source. The offeror will provide ongoing support of the SRM software.

8.3.2 Batch Functionality

The SRM will provide the common features provided by most HPC batch systems (such as SLURM, TORQUE, and Cobalt) for queuing, scheduling, and managing CORAL workloads.

8.3.2.1 Job Submission (TR-1)

The SRM will accept batch submissions from non-privileged users and generate a unique batch identifier for each job. Submissions will be accepted from the FENs and the CNs. To support advanced workflows, submissions will also be accepted from remote, non-CORAL resources.

8.3.2.2 User Submission Filtering (TR-1)

The SRM will provide a script-based mechanism for the Laboratories to inspect, to augment, and to modify all user-provided submission directives. Users will not be able to bypass this mechanism.

8.3.2.3 Interactive Jobs (TR-1)

The SRM will support interactive jobs where a user's terminal is connected to the batch job's standard input and output. The SRM will also support interactive jobs that forward X11 connections to enable GUI tools such as debuggers.

8.3.2.4 Project ID Association (TR-1)

The SRM will provide a means for users to associate each job with a project ID (independent of their fair-share account). A user will have a default project ID, and the capability to override it at job submission.

8.3.2.5 Prologues and Epilogues (TR-1)

The SRM will provide the ability to execute site-specific scripts as a privileged user and as the end user both before and after a job on the head node (if applicable) as well as each individual compute node to perform functions such as statistic gathering and file/memory cleanup.

8.3.3 Application Launch

The SRM (or other Offeror-provided software) will provide a mechanism whereby a user can launch any distributed application (i.e., not just MPI jobs) including daemons, and/or threads that run on a set of system resources allocated to that user.

8.3.3.1 Multiple Application Management (TR-1)

The application launcher will support running multiple applications concurrently within a job allocation, commonly called "tasks," "job steps," or "ensemble runs." Users will not have to specify resources (such as compute nodes by hostname) manually to ensure that applications do not overlap, and application launch requests that cannot be immediately satisfied due to resource exhaustion will be queued and run once resources become free.

8.3.3.2 Multiple Program Multiple Data (TR-1)

The application launcher will support launching a collection of processes (in the case of MPI, all processes in a single *MPI_COMM_WORLD*) where each process may require different binaries, command line arguments, and/or amounts of compute resources.

8.3.3.3 Task Binding (TR-1)

The application launcher will provide optimized placement and binding of processes launched within a node, avoiding interference between processes and minimizing latency and unnecessary data transfer. Users will be able to influence/specify the placement and binding.

8.3.3.4 Container Support (TR-2)

The application launcher will support running processes inside containers to enable alternate userspace environments. Common container formats will be supported, such as the Open Container Initiative (https://www.opencontainers.org/) v1.0 (or then current) Image Specification used by Docker or Singularity's image format (http://singularity.lbl.gov).

8.3.4 Scheduling Policies and Limits

Offeror's SRM will support a broad set of customizable scheduling policies and limits to implement the complex requirements of the Laboratories.

8.3.4.1 Priority Factors (TR-1)

The SRM will prioritize batch jobs based on a variety of factors, including user, account, queue, job size, job queue time, and fair share.

8.3.4.2 Fair Share Scheduling (TR-1)

The SRM will implement hierarchical fair-share scheduling, and provide a mechanism for administrators to set usage targets by fair-share account and user.

8.3.4.3 Backfill Scheduling (TR-1)

The SRM will support the ability to run jobs outside of normal priority-order to improve resource utilization when running these jobs would not delay the start of the highest priority jobs.

8.3.4.4 Topology-Aware Scheduling (TR-2)

The SRM will be aware of the system interconnect topology and make scheduling decisions to optimize average/maximum hop count, latency, and bandwidth. Users will be able to specify the sensitivity of their job to interconnect placement and specify a maximum duration to delay job start waiting for optimal placement. Within a job, the SRM will expose information to allow tools and batch scripts to determine the interconnection topology and other detailed information concerning the resources allocated to it. Special privileges will not be required to access this information.

8.3.4.5 Job Limits (TR-1)

The SRM will support a limit on the number of jobs running and jobs being considered for scheduling (blocked) by user, account, and queue.

8.3.4.6 Policy Import (TR-1)

SRM priority factors and job limits will be importable from external systems on a regular basis. The SRM will be able to update its policies without a full restart.

8.3.4.7 Job Dependencies (TR-1)

The SRM will support user-directed job dependencies, including dependencies conditional on the exit code of its dependent job. Jobs with unmet dependencies will not be eligible for execution.

8.3.4.8 Preemption (TR-1)

The SRM will support job preemption. Policies (such as queue, job walltime, and job size) can be set to influence which jobs are preemptable and which jobs can preempt other jobs.

8.3.4.9 Resource Reservations (TR-1)

The SRM will support administrator-created advanced reservations for resources, including nodes, storage, licenses, and generic resources. Reservations will support Access Control Lists (ACLs) that limit which user, group, or account may access the reserved resources.

8.3.4.10 Scheduling Policy Plugin Interface (TR-2)

The SRM will provide a plugin interface to replace Offeror-supplied scheduling algorithms with site-specified scheduling behavior.

8.3.4.11 User Interface (TR-1)

The SRM will provide a summary status including jobs grouped in the following states: running, eligible, blocked, and data staging (if applicable). For eligible jobs, the SRM will show jobs in priority order, will be able to explain all calculated priority values, and will provide an estimated start time. For blocked jobs, the SRM will provide the reason(s) that the job is blocked. The SRM will provide an interface to display backfill windows.

8.3.4.12 Scheduler Simulation (TR-2)

The SRM will support an offline simulation mode that allows the Laboratories to test alternate scheduling policies without impacting the production system.

8.3.5 Reporting

8.3.5.1 Resource Utilization Reporting (TR-1)

For utilization reporting, the SRM will recognize four mutually exclusive resource states: allocated, reserved, idle, and down. The SRM will provide an interface to report the time spent in each of these states, for any given set of resources over any given period of time.

8.3.5.2 Job Reporting (TR-1)

The SRM will provide an interface to report the time used by all jobs, broken down by fair-share-account, user, project ID, assigned resources, or any combination thereof, over any given period of time.

8.3.5.3 Job History Data Dump (TR-1)

The SRM will provide a means to dump a record of all jobs that have executed on the system, including any state transitions of assigned resources, and RAS events that occurred during execution.

8.3.6 Advanced SRM Functionality

8.3.6.1 Power Management (TR-1)

The SRM will assist with power management, allowing jobs to be submitted with power budgets, scheduling jobs according to power availability, and managing power caps on assigned resources such that jobs remain under budget, while (optionally) maintaining performance uniformity. The SRM will utilize the node-level HPMCI (Section 5.1.6) capabilities for this function.

8.3.6.2 Job Energy Reporting (TR-1)

Job energy usage will be reported to users and recorded in the telemetry data base (Section 10.3) for system accounting purposes.

8.3.6.3 Power Usage Hysteresis (TR-2)

The SRM will provide the ability to constrain power usage ramp-up and ramp-down rates to meet facilities requirements.

8.3.6.4 Resource Allocation Elasticity (TR-2)

The SRM will provide a capability for resource allocation elasticity so that a running job can request additional resources, e.g., to replace a node that has failed, to increase the power cap of a node that is arriving late to barriers or to allocate and to release nodes as the workload moves through phases of execution.

8.3.6.5 Local Storage Management (TR-2)

The SRM will manage local/embedded storage as a resource, facilitating 1) creation of job-local name spaces, 2) staging of data on/off job-local storage, and 3) scheduling computation for data locality.

8.3.6.6 I/O System Bandwidth Management (TR-2)

The SRM will manage the I/O subsystem bandwidth as a resource, for example, allowing a job to request desired I/O bandwidth, then at runtime, manipulating I/O subsystem resources allocated to the job and/or utilizing quality-of-service hooks, if available, to provide the requested bandwidth. The solution will ensure that a single job will not cause I/O starvation for other jobs. Nonetheless, in the absence of other competing jobs, a job will achieve higher I/O bandwidth than requested. The SRM will also provide

the ability for administrators to place I/O bandwidth limits on jobs, resulting in bandwidth throttling if such limits are exceeded.

8.3.6.7 Fault Notification (TR-1)

The SRM will provide a mechanism such as CIFTS FTB-API (http://www.mcs.anl.gov/research/cifts/) to notify fault-tolerant runtimes when a system fault occurs that might require the runtime to take some recovery action.

8.3.7 SRM Application Programming Interface (API)

The Offeror will provide an SRM-API to support Laboratory-developed tools that interact with the SRM. The SRM-API will have access to all data described in the following.

8.3.7.1 Compute Status Query Interface (TR-1)

The SRM-API will provide an interface to query the current status of all hardware, software, and running jobs on compute resources.

8.3.7.2 Job Signaling Interface (TR-1)

The SRM-API will provide an interface to pass POSIX-style signals to running jobs. The SRM-API will require authentication of the user issuing the signal to the job.

8.3.7.3 Cross-Language Compatibility (TR-1)

The SRM-API will be written in a way that facilitates cross-language binding and will not preclude the generation of bindings to other programming languages.

8.3.7.4 Multiple Language APIs (TR-3)

The SRM-API will be provided for multiple languages including, but not limited to C, C++ and Python.

8.3.7.5 Real-time Notification Interface (TR-2)

The SRM-API will provide real-time notifications of status changes in both user jobs and the hardware required to run user applications. This interface will provide reliable message delivery of such events. The interface will provide events through a message queuing protocol like AMQP (http://www.amqp.org/).

8.3.8 Performance and Scalability

8.3.8.1 Support for Thousands of Jobs (TR-1)

The SRM system will support thousands of simultaneous running jobs and tens of thousands of simultaneous idle/blocked jobs.

8.3.8.2 Job Start Performance (TR-1)

The SRM will be able to start a full-system job within three minutes. The SRM will be able to clean up after a successful full-system job within three minutes. The SRM will be able to clean up after an unsuccessful full-system job within ten minutes.

8.3.8.3 System State Responsiveness (TR-1)

The SRM and SRM-API will provide the overall system status in a scalable fashion, providing information needed in less than five (5.0) seconds.

8.3.8.4 Concurrent Access and Control (TR-1)

The SRM and SRM-API will be safe for concurrent access to its data, as well as issuing concurrent commands. Multiple processes issuing commands concurrently will not leave the system in an inconsistent state.

8.3.8.5 Centralized Access (TR-1)

The SRM and SRM-API will provide all status information and execute all provisioning and application control commands from any front-end node. Multiple nodes will not have to be explicitly queried for information; the SRM will handle any aggregation.

8.3.9 Laboratory-Provided SRM (TR-2)

The Laboratories have investments in SRM software that they may wish to deploy on the CORAL platform, for user interface ubiquity, scheduling policy implementation, or integration with other advanced system software deployed at the site. Therefore, in addition to providing an integrated SRM solution, the Offeror will expose open, documented, abstract interfaces that enable the integrated SRM to be replaced with site-provided SRM software. These interfaces will be the same ones used by the integrated SRM to ensure appropriate attention as the system is designed and developed. The decision to use the integrated SRM versus site-provided SRM software will be made by each site.

9.0 Front-End Environment

The FEE includes Front-End Node (FEN) hardware as well as software necessary to support end-users of the system. The licenses for all software provided shall allow for use by any users of the system. The Laboratories manage access to their respective systems and can include users from outside the Laboratories.

FEE functionality can be broken down into four main use cases: data analysis, interactive, compilation and job execution. These use cases drive specific FEN requirements as follows.

Data analysis: Pre- and post-processing of data for and from the computation nodes may require visualization and workflow management tools, which often require large amounts of memory capacity and bandwidth and network bandwidth both to end-user networks and to the I/O subsystem.

Interactive use: FENs must support interactive: editing; job submission; tracking of job progress; and review of job output. Interactive use, being the most general use case, is the most prone to causing system instability. In other words, users will stress the nodes causing a crash with some regularity, which is the main motivation for isolating interactive use from the other use cases.

Code compilation: The compilation of some CORAL applications is resource and time intensive, and multiple compilations must be able to run simultaneously.

Job execution: Batch jobs are initiated from the FENs. Batch scripts often do more than just launch the job. Batch scripts can move or preprocess data or automatically generate the input deck. These nodes must be very stable, so ideally they should be distinct from the interactive nodes, although users will still need to log into these nodes, for example to attach debuggers or profiling tools.

9.1 Front-End Node (FEN) Hardware Requirements

Having stable, robust FENs directly affects user experience and code development efficiency. The following requirements are specific to the FEN hardware.

9.1.1 FEN Count (TR-1)

The Offeror will propose a pool of FENs that satisfies the use cases described in Section 9.0. If distinct FEN types are proposed, the Offeror will justify the distribution of the different types proposed. Alternatively, Offeror may propose an integrated "Front End Service" that seamlessly and possibly dynamically provides multiple FEN types that are optimized for these use cases.

9.1.2 FEN Disk Resources (TR-1)

The FEN will have sufficient disk resources in aggregate to store: 1) multiple system software images for each node type; and 2) 50 TB in aggregate, of local temporary storage space. These storage resources will be packaged with the node (i.e., physically local) or packaged remotely, but locally mounted. FEN storage may consist of hard disks or NVRAM. The FEN locally mounted storage will be configured with High Availability, High IOPS RAID 6 (or better) arrays of hard disks or NVRAM. The FEN will be able to boot over a network and mount a shared root file system.

9.1.3 FEN High-Availability (TR-1)

All FENs and disk arrays will have high availability features including but not limited to redundant and hot swappable power supplies, hot swappable disk drives (if packaged locally), hot swappable fans, spare drives, and ECC memory.

9.1.4 FEN IO Configuration (TR-2)

All FENs will have sufficient network interfaces to access a local site network, the management network and the file system network. All FENs will also have sufficient storage interfaces to access the FEN Disk Resources described in Section 9.1.2. The IO slots and adapters provided will be industry standard. The FEN will have at least two free IO slots such that the sites can configure additional network or storage interfaces as needed.

9.1.5 FEN Delivered Performance (TR-2)

The Offeror's proposed FEN configuration will have sufficient processing power, memory capacity and bandwidth, number of interfaces and delivered bandwidth and local storage capacity and bandwidth to provide FEN functions described in Section 9.0. The front-end environment will collectively support 50 interactive users, 5 data analysis front-end applications, 500 batch jobs, and 20 simultaneous compilations of software of equivalent complexity to the latest GNU Compiler Suite.

9.1.6 FEN Access Management (TR-2)

Access to the Data Analysis, Code Compilations and Job Execution FENs will be controlled by the System Resource Management (SRM) system.

9.1.7 Interactive FEN Login Load Balancing (TR-2)

The Offeror will provide a mechanism whereby user logins are dynamically load balanced across the available FENs based on the load average of each FEN or Laboratory defined policy. Alternatively, the Offeror will enable the dynamic reconfiguration of an FEN – such as in a logical partition or virtual machine – that may span or be a subset of multiple physical hosts.

9.2 Front-End Environment Software Requirements

9.2.1 Parallelizing Compilers/Translators

9.2.1.1 Baseline Languages (TR-1)

The Offeror will provide fully supported implementations of Fortran 2008 (ISO/IEC 1539-1:2010, ISO/IEC TR 19767:2005(E), ISO/IEC TR 29113 - https://www.iso.org/standard/50459.html), C (ANSI/ISO/IEC 9899:2011; ISO/IEC 9899:2011 Cor. 1:2012(E) - http://www.open-std.org/jtc1/sc22/wg14/www/standards), and C++ (ANSI/ISO/IEC 14882:2014 - http://www.open-std.org/jtc1/sc22/wg21/docs/standards) or then current versions. Fortran, C, and C++ are referred to as the baseline languages. An assembler will be provided. The Offeror will provide the fully supported capability to build programs from a mixture of the baseline languages (i.e., inter-language sub-procedure invocation will be supported).

9.2.1.2 Baseline Language Optimizations (TR-1)

The Offeror will provide baseline language compilers that perform high levels of optimization that allow the application programmer to use all CN supported hardware features. Baseline language compilers will support directives to provide information (e.g., aliasing information beyond the restrict keyword) required for or to direct additional optimizations.

9.2.1.3 Baseline Language 64b Pointer Default (TR-1)

The Offeror will provide compilers for the baseline languages that are configured with the default mode of producing 64b executables. A 64b executable is one with all virtual memory pointers having 64b. All operating system calls will be available to 64b executables. Offeror supplied libraries will provide 64b objects. The Offeror's software will be fully tested with 64b executables.

9.2.1.4 Baseline Language Standardization Tracking (TR-1)

The Offeror will provide a version of the baseline languages that is standard compliant within eighteen months after ANSI or ISO/IEC standardization, whichever occurs earlier. The Offeror is encouraged to adhere to the current proposed standard.

9.2.1.5 Common Preprocessor for Baseline Languages (TR-2)

The Offeror will provide preprocessing of ANSI C preprocessor directives in programs written in any of the baseline languages. The preprocessor will set macros that specify the target execution environment in order to facilitate configuration for cross-compilation environments.

9.2.1.6 Baseline Language Compiler Generated Listings (TR-2)

The Offeror will provide baseline language compiler options to produce code listings with pseudo-assembly listings, optimizations performed and/or inhibitors to optimizations on a line-by-line, code block-by-code block or loop-by-loop basis and variable types and memory layout.

9.2.1.7 Cray Pointer Functionality (TR-2)

The Offeror will support Cray style pointers in an ANSI X3.9-1977 Fortran compliant compiler.

9.2.1.8 Baseline Language Support for OpenMP Parallelism (TR-1)

The Fortran, C, and C++ compilers will support OpenMP Version 5.0 (or then current) (http://www.openmp.org). All baseline language compilers will include the ability to perform automatic

parallelization. The baseline language compilers will produce symbol tables and any other information required to enable debugging of OpenMP parallelized CORAL applications.

9.2.1.8.1 OpenMP Performance Optimizations (TR-2)

The baseline languages and runtime library support for the compute node will include optimizations that minimize the overhead of locks, critical regions, barriers, atomic operations, tasks and self-scheduling "do-loops" by using special compute node hardware features. The time to execute an OpenMP barrier with NCORE OpenMP threads will be less than 200 clock cycles. The overhead for OpenMP Parallel FOR with NCORE OpenMP threads will be less than 500 cycles in the case of static scheduling.

9.2.1.8.2 OpenMP Performance Interface (TR-1)

The baseline languages will implement any OpenMP performance interface specified in OpenMP Version 5.0 (or then current). The baseline languages will support mapping to source code including Fortran modules and C++ namespaces.

9.2.1.8.3 OpenMP Debug Interface (TR-1)

The baseline languages will implement any OpenMP debug interface specified in an OpenMP technical report or adopted in the OpenMP specification.

9.2.1.8.4 OpenMP Language Standardization Tracking (TR-1)

The Offeror will provide an OpenMP implementation that is standard compliant within nine months after OpenMP Architecture Review Board standardization. The Offeror is encouraged to adhere to the current proposed standard.

9.2.1.9 Baseline Language Support for OpenACC Parallelism (TR-2)

For the 2021 system proposal, the Offeror will provide Fortran, C, and C++ compilers or interpreters that support node parallelism through OpenACC Version 2.6 (or then current) (http://openacc.org/) in order to support CORAL applications that currently use OpenACC. The Offeror will support interoperability of OpenACC with OpenMP 5.0 (or then current directives).

9.2.1.9.1 OpenACC Language Standardization Tracking (TR-2)

For the 2021 system proposal, the Offeror will provide an OpenACC implementation that is standard compliant within nine months after OpenACC-Standard.org standardization. The Offeror is encouraged to adhere to the current proposed standard.

9.2.1.10 Baseline Language Support for Sanitizer Instrumentations (TR-2)

All baseline languages will support sanitizer instrumentations (https://github.com/google/sanitizers) for code-correctness checks, which include, but are not limited to address, thread, memory, leak, and undefined-behavior sanitizers. OpenMP support for ThreadSanitizer will be enabled by annotating the OpenMP runtime via the OpenMP Performance Interface as used in Archer OpenMP data race detector (https://github.com/PRUNERS/archer).

9.2.1.11 Baseline Language Support for POSIX Threads (TR-1)

All baseline languages will support node parallelism through POSIX threads Version 2.0 (or then current) (http://www.opengroup.org/onlinepubs/007908799/xsh/threads.html). The baseline language compilers will produce symbol tables and any other information required by the debugger to enable debugging of POSIX thread parallelized CORAL applications.

9.2.1.12 Baseline Language Support for Thread-Local Storage (TR-2)

All baseline languages will provide support for storing static variables in thread-local storage, with a distinct copy for each thread. This support will be provided both per variable, through type modifiers (__thread, for C, thread_local, for C++), and globally, through a compilation switch that applies to all static variables declared in the program.

9.2.1.13 Baseline Language and GNU Interoperability (TR-1)

The baseline language compilers will produce binaries that are compatible with the GNU compilers and loaders. In particular, the delivered baseline compiler OpenMP runtime libraries will be compatible with the GNU OpenMP libraries. That is, a single OpenMP based application can be built, run and debugged using modules generated from both Offeror supplied baseline language compilers and GNU compilers.

9.2.1.14 Support for GCC Compiler Extensions (TR-1)

The Offeror's C and C++ compilers will support GCC-style inline assembly syntax (see http://gcc.gnu.org/onlinedocs/gcc/Extended-Asm.html#Extended-Asm) in addition to support for atomic operations that are part of the C11 and C++14 languages. The Offeror's C and C++ compilers will support GCC atomic extensions (see http://gcc.gnu.org/onlinedocs/gcc/_005f_005fsync-Builtins.html#g_t_005f_005fsync-Builtins and http://gcc.gnu.org/onlinedocs/gcc/_005f_005fatomic-Builtins.html#g_t_005f_005fatomic-Builtins). The Offeror's C and C++ compilers and the linker will support thread-local storage, including the C++14 keyword 'thread_local' and the GCC '__thread' language extension keyword (https://gcc.gnu.org/onlinedocs/gcc-3.3/gcc/Thread-Local.html).

9.2.1.15 Runtime GNU Libc Backtrace (TR-2)

The baseline language compilers runtime support will provide the same backtrace functionality as GNU libc (see http://www.gnu.org/software/libc/manual/html_node/Backtraces.html).

9.2.1.16 Baseline Debug Information (TR-1)

The baseline languages and OpenMP support will produce DWARF 5 (or then current) compliant debugging information. The debug information will be qualified against a test suite to ensure proper operations of both debugging and performance analysis tools. The debugging information will be present and accurate on inlined functions (e.g., through DWARF's inlined-subroutine and parameter constructs). This support will allow tools to construct accurate stack traces that consist of both the inlined subroutines and their containing subroutines. The baseline information will be produced even when the code is compiled with "-O -g".

9.2.1.17 Accurate Debug Information for Optimized Applications (TR-2)

The baseline language compilers and OpenMP support will produce debug information for applications that are compiled with the "-O -g" code optimization. The code generated with these options will be optimized while still allowing for accurate debug information, which is required by both debugging and performance analysis tools when working with optimized applications. The debugging information will support source context including program variables and stack traces. When the code is optimized such that the location of a variable changes during its lifetime at runtime, the debugging information will provide a list of its locations (e.g., through DWARF's location-lists construct). The debugging information will also include accurate information about inlined functions (e.g., through DWARF's inlined-subroutine and parameter constructs). The runtime libraries of the baseline languages will retain key debugging information such as stack frame information. Python will provide hooks for the debuggers to use in reconstructing Python-level stack traces from raw stack traces. The Python-level traces will include not

only Python function names but also other information relevant to these functions including, but not limited to, the script files and line numbers that contain these functions and Python variables. The Offeror will enable these hooks by ensuring that key variables within the Python interpreter will not be optimized out.

9.2.1.18 Debugging Information Compression (TR-2)

The baseline languages will support compression and duplicate-elimination of the debugging information found in object files, dynamic shared objects, and executables. The baseline languages will support split DWARF to split object file and package representations, which allows DWARF information to be kept separate from executable files.

9.2.1.19 Floating Point Exception Handling (TR-2)

The baseline languages will provide compiler flags that allow an application to detect Floating Point Exception (FPE) conditions occurring at runtime within a module compiled with those flags. This support will provide the compiled modules with an option to select any combination of the floating point exceptions defined for IEEE-754. Further, the baseline languages will provide a compiler flag to inject 64b signaling NaNs into the heap and stack memory such that an application can easily detect the use of uninitialized memory.

9.2.1.20 Pointer Disambiguation Directives in C++ and C (TR-2)

The Offeror will supply directives and attributes to disambiguate aliasing of pointer arrays, and will provide robust SIMD optimizations with respect to them. Examples of directives include support for the __restrict__ pseudo-keyword in the C++ compiler for any declared pointer and specification through the __declspec or __attribute__ mechanisms found in all C/C++ compilers, avoiding Offeror-specific directives where possible, with a preference for the __attribute__ mechanism. These features will be available within the scope of a typedef declaration, and will be propagated through all optimization machinery to where the typedef is used.

9.2.1.21 Weak Symbols (TR-2)

The baseline language compilers will support weak symbols with pragmas or attributes similar to the GCC "__attribute__((weak))" syntax.

9.2.1.22 Compiler Based Instrumentation (TR-2)

The baseline language compilers will provide compiler based instrumentation similar to the functionality provided through the GCC "-finstrument-functions" flag.

9.2.1.23 Linker Wrapping (TR-2)

The linker for the baseline language compilers will provide for link time wrapping of function symbols similar to the GNU ld "-wrap" flag functionality.

9.2.1.24 Multiple Language Implementations (TR-3)

CORAL finds it useful to have multiple implementations of the supported programming languages (C, C++, Fortran, OpenMP, OpenACC, per Offeror response) available and supported on the system.

9.2.2 Debugging and Tuning Tools (TR-1)

All debugging and tuning tools will be 64b executables and operate on 64b user applications. All debugging tools, profiling tools, and tool features will be functional on any accelerators or co-processors in the system.

9.2.2.1 Code Development Tools Infrastructure (CDTI) (TR-1)

The Offeror will propose a hierarchal mechanism for code development tools (CDT) to interact with CORAL applications on the system in a secure, reliable, and scalable manner. CDTI will include, but not be limited to, remote process control interface; launching and bootstrapping interface; the MPIR process acquisition interface (i.e., http://www.mpi-forum.org/docs/mpir-specification-10-11-2010.pdf); programmable core file generation interface (Section 5.2.16); CDT communication interface such as MRNet (http://www.paradyn.org/mrnet); and node-level dynamic instrumentation interface such as DynInst (http://www.dyninst.org).

9.2.2.2 Parallel Debugger for CORAL Applications

9.2.2.2.1 Baseline Debugger (TO-1)

The Offeror will separately propose Allinea DDT (http://www.allinea.com/products/ddt), Allinea Forge Professional (https://www.allinea.com/products/develop-allinea-forge), and Rogue Wave Software's TotalView (http://www.roguewave.com/products/totalview.aspx).

9.2.2.2.2 Functionality and Performance (TR-1)

The Offeror-proposed debuggers will be capable of debugging CORAL applications with multiple parallel programming paradigms (e.g., message passing and OpenMP thread parallelism) using debug libraries (e.g., Section 9.2.1.8.3 and Section 9.2.5.1.1) and multiple baseline languages (Section 9.2.1.1). The debugging capabilities will include, but not be limited to, scalable debugging of dynamically shared objects and debugging support for threads (e.g., an ability to show the state of thread objects such as acquired mutex locks and asynchronous thread control), optimized codes (Section 9.2.1.17), memory (including instant detection of access violation using guard page described in Section 5.2.7), and core files.

The Offeror will describe how each debugger will scale to use CDTI (Section 9.2.2.1) to establish a debug session for any subset of CN processes of a job (i.e., one process to all processes) either at job launch under the control of the debugger or via attaching to a running job, and to expand, to shrink or to shift the subset by attaching to more processes in the job and/or detaching from some of the already attached processes. The performance of any debugger operation will scale as a function of the process/thread count of the attached subset up to 20% of the size of the machine. The tools will perform and scale well on CORAL application executables that contain more than 800 MB aggregate debug symbols and shared library dependencies and that use more than 85% of total CN memory.

9.2.2.2.3 Increased Debugger Scalability (TR-2)

The Offeror-proposed debugger will scale well for large jobs. The Offeror will detail projected scalability, with appropriate documentation and justifications for claims and assumptions.

9.2.2.2.4 Efficient Handling for C++ Templates (TR-2)

The Offeror-proposed debuggers will work when it inserts breakpoints into source lines in C++ templates that correspond to up to 50,000 different code addresses.

9.2.2.2.5 Fast Conditional Breakpoints and Data Watchpoints (TR-2)

The Offeror-proposed debuggers will use the hardware support (see Section 5.1.7) to provide fast conditional breakpoints and data watchpoints in all baseline languages. An implementation for source code conditional breakpoints or watchpoints should add an overhead of less than 14 microseconds ($14x10^{-6}$ seconds) per execution of the non-satisfied condition when the condition is a simple compare of two variables local to the process or thread.

9.2.2.2.6 Reverse Debugging (TR-3)

The Offeror will propose a reverse-debugging mechanism. Using the mechanism, the debugger will step a parallel CORAL application backward as well as forward.

9.2.2.2.7 Interoperability with Record and Deterministic Replay (TR-3)

The Offeror-provided debuggers will interoperate seamlessly with record and replay tools such as (https://github.com/PRUNERS/ReMPI) that record MPI matching order in one run and deterministically replay the same order in subsequent runs.

9.2.2.3 Stack Traceback (TR-2)

The Offeror will propose runtime support for stack traceback error reporting. Critical information will be generated to STDERR upon interruption of a process or thread involving any trap for which the user program has not defined a handler. The information will include a source-level stack traceback (indicating the approximate location of the process or thread in terms of source routine and line number) and an indication of the interrupt type.

Default behavior when an application encounters an exception for which the user has not defined a handler is that the application dumps a core file. By linking in an Offeror-provided system library the application may instead dump a stack traceback. The stack traceback indicates the stack contents and call chain as well as the type of interrupt that occurred.

9.2.2.4 User Access to a Scalable Stack Trace Analysis Tool (TR-2)

The Offeror will supply a scalable stack trace analysis and display GUI-based tool that will allow non-privileged users to obtain a merged stack traceback securely and interactively from a running job.

9.2.2.5 Lightweight Corefile API (TR-2)

The Offeror will provide the standard lightweight corefile API, defined by the Parallel Tools Consortium or a mutually agreed-upon equivalent format, to trigger generation of aggregate traceback data for all running threads. The Parallel Tools Consortium defines a specific format for lightweight core files (see http://web.engr.oregonstate.edu/~pancake/ptools/lcb/). The Offeror will provide an environment variable (or an associated command-line flag), with which users can specify that the provided runtime will generate lightweight corefiles instead of standard Linux/Unix corefiles. User control will be provided to generate either lightweight or standard Linux/Unix corefiles from a selected subset of the MPI rank processes as well as to select the file-system locations into which to store them.

9.2.2.6 Profiling Tools for Applications (TR-1)

The Offeror will provide a range of application profiling tools including Open|SpeedShop (https://openspeedshop.org/), TAU (http://www.cs.uoregon.edu/research/tau/home.php), HPCToolkit (http://hpctoolkit.org/), VAMPIR (http://www.vampir.eu) and gprof (https://www.gnu.org/software/binutils).

9.2.2.6.1 Statistical Sampling Profiling (TR-1)

The Offeror will provide a mechanism for tools to register signal-handler callbacks that trigger on signals generated by the Timer API and hardware events. These callbacks will allow tools to record the application's stack trace, MPI rank, thread identifier, and information about the signaling event.

9.2.2.6.2 Lightweight Message-Passing Profiling (TR-1)

The Offeror will provide the mpiP library (http://mpip.sourceforge.net/), a lightweight, scalable profiling library for MPI that captures only timing statistics about each MPI process.

9.2.2.7 Event Tracing Tools for Applications (TR-1)

The Offeror will provide the Score-P measurement infrastructure (http://www.vi-hps.org/projects/score-p) for tracing programming model events as well as collecting hardware performance counters. The Open|SpeedShop POSIX I/O tracer will also be provided through the Open|SpeedShop toolset (https://openspeedshop.org). Trace records will utilize the OTF2 trace file format for all baseline languages. Distributed mechanisms for generating event records from all process and threads in the parallel program will include timestamp and event type. Event tracing tool APIs will provide functions to activate and to deactivate event monitoring during execution from within a process. By default, event tracing tools will not require dynamic activation to enable tracing.

9.2.2.8 Performance Monitor APIs and Tools for Applications (TR-1)

The Offeror will provide performance monitor APIs and tools, whereby performance measures from hardware performance monitors are obtained for individual threads or processes are reported and summarized for CORAL applications. The Offeror will provide a native API that allows full access to the performance monitor hardware. The Offeror will deliver PAPI, Version 5 (or then current), that gives user applications access to the 64b hardware performance monitors (Section 5) and exposes all HPM functionality to user applications. The native HPM API and PAPI will include functions that allow user applications to initialize the HPM, to initiate and to reset HPM counters, to read HPM counters and to generate interrupts on HPM counter overflow and to register interrupt handlers from each process and thread independently without affecting the counts on other process and threads. The APIs will make it possible to associate HPM counter values with code blocks executed by individual processes or threads. When there is no hardware support for accurate association (e.g., events that are generated from a shared resource outside the cores), the APIs will make it possible to make a best-effort association for key use cases (e.g., a single application is exclusively running on the node). Offeror will provide (non-privileged) user-level access to these counters including those shared resource counters in a secure manner without having to lower the OS's security configuration level. For performance metrics that need to be derived from multiple counters, documentation about these metrics and their counters will be provided.

9.2.2.9 Timer API (TR-2)

The Offeror will provide an API for interval wall clock and for interval timers local to a thread/process. The interval wall clock timer mean overhead will be less than 250 nanoseconds to invoke and will have a resolution of 1 processor clock period. The system and user timers mean overhead will be less than 250 nanoseconds to invoke and will have a global resolution of 3 microseconds (i.e., this wall clock is a system wide clock and is accurate across the system to 3 microseconds).

9.2.2.10 Valgrind Infrastructure and Tools (TR-1)

The Offeror will provide the open source Valgrind infrastructure and tools (http://valgrind.org) for the CN, as well as for the FEN environment. For the CN, the solution may require the application to link with

Valgrind prior to execution. The provided Valgrind tool ports will be offered for upstream publication through the Valgrind.org maintained repository. At a minimum, CORAL will be provided the source code and the ability to build the Valgrind tools. At a minimum, the Valgrind release 3.13.0 (or then current) tools Memcheck and Helgrind will be provided. The Offeror will make available the documentation required to port Valgrind through agreements that protect the intellectual property of the Offeror.

9.2.3 Facilitating Open Source Tool Development (TR-1)

The Offeror will provide sufficient documentation of the hardware and software to enable CORAL, open source projects, or subcontractors to implement compilers, assemblers, and libraries as open source software that make full use of the architecture including any accelerators, interconnects, or other performance-related features used by Offeror-supplied compilers, assemblers, and libraries. Sufficient documentation will be publicly available, unencumbered by licensing or disclosure agreements, as to allow open source projects not directly affiliated with CORAL to ascertain the functionality, correctness, and suitability of code contributed to their projects.

9.2.4 Application Building

9.2.4.1 FEN Cross-Compilation Environment for CN (TR-1)

The Offeror will provide a complete cross-compilation environment that allows the sites to compile and to link applications and daemons on the FEN for execution on the CN. The FEN environment will support the use of standard GNU Autotools (Autoconf 2.69, Automake 1.15, Libtool 2.4.6, or then current versions) for automatic configuration and building of libraries and applications, including detection of the correct ISA (Instruction Set Architecture), OS, and runtime libraries for the CN, rather than the FEN. GNU Autoconf requires an appropriate version of GNU M4.

9.2.4.2 GNU Make Utility (TR-1)

The Offeror will provide the GNU make utility version 4.2 (or then current) with the ability to utilize parallelism in performing the tasks in a makefile.

9.2.4.3 CMake (TR-1)

The Offeror will provide the CMake build system, version 3.8.2 (or then current). Offeror will also provide CMake platform files that enable cross-compiling. Platform files will correspond to tool chains available on the FENs and CNs, and all compiler tool chains available on these nodes.

9.2.4.4 Linker and Library Building Utility (TR-1)

The Offeror will provide an application linker with the capability to link object and library modules into dynamic and static executable binaries. A static execution binary has all user object modules and libraries statically linked when the binary is created. A dynamic executable binary has all user object modules and static libraries linked at binary creation, but user and system dynamic libraries are loaded at runtime on a demand basis.

9.2.5 Application Programming Interfaces (TR-1)

All Offeror supplied APIs will support 64b executables and be fully tested in 64b mode. In particular, benchmarks will be 64b executables that utilize MPI with multiple styles of SMP parallelism in a single 64b executable and run successfully with at least 1 GiB of user memory per user process over the entire machine.

9.2.5.1 Optimized Message-Passing Interface (MPI) Library (TR-1)

The Offeror will provide a fully supported, highly optimized implementation of the MPI-3.1 standard as defined by http://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf (or then current). The delivered MPI library will be thread safe and allow applications to use MPI from individual threads. MPI_THREAD_MULTIPLE and MPI_THREAD_FUNNELED threaded application modes will be supported. The MPI implementation will deliver asynchronous progress in all types of nonblocking communication, including nonblocking send-receive, nonblocking collectives and one-sided (also known as RMA). The Offeror will map the MPI implementation to the architecture so as to expose the full capabilities of the CORAL interconnect for a wide variety of usages. The Offeror will minimize scalability limitations, including memory usage in the implementation. The Offeror will provide (electronic) written documentation that describes the performance features of the MPI implementation for each software release on the proposed CORAL hardware. All environmental settings that impact MPI operation, buffering and performance and their impact to 64b user applications performance will be tested and their effectiveness and reliability documented. The environment information will be returned through matching control variables in the MPI tool information interface, MPI_T.

9.2.5.1.1 Support for MPI Message Queue Debugging (TR-2)

The Offeror-provided MPI library will enable MPI message queue debugging to work with the supplied debugger. The Offeror will provide a library that allows the debugger to access message queue information in MPI. The library will export a set of entry points as documented in the MPI message queue debug support API specification.

9.2.5.1.2 Performance Variables in the MPI Tool Information Interface (TR2)

The Offeror will expose useful MPI internal performance information through appropriate performance variables in the MPI tool information interface (MPI_T). Useful information could include performance variables that show blocking vs. communication time, memory consumption within MPI, or the length of any queue used in the MPI implementation.

9.2.5.1.3 MPI Standardization Tracking (TR-1)

The Offeror will provide an MPI implementation that is standard compliant within nine months after MPI Forum standardization. The Offeror is encouraged to adhere to the current proposed standard.

9.2.5.2 Graphical User Interface API (TR-1)

The Offeror will provide the typical Linux graphical user environment, including X11R7.7 (http://www.x.org/wiki/), Motif 2.3.4 (http://www.opengroup.org/motif/) and Qt 5.9 (http://en.wikipedia.org/wiki/Qt_(toolkit), or then current versions, applications, servers and API libraries. Secure viewing and usage of X-Windows to users remote workstations will be accomplished by Laboratory-provided SSH encrypted tunneling. All provided GUI API will be compatible with this approach. Offeror will provide NX, Neatx, or similar technology to facilitate the use of X11-based programs across a wide-area network. In addition to standard X11 support, the Offeror will provide remote desktop access to users via VNC.

9.2.5.3 Visualization API (TR-3)

The Offeror will provide OpenGL 4.3, or then current version, (http://www.opengl.org).

9.2.5.4 Math Libraries (TR-2)

The Offeror will provide optimized single-node floating point (e.g., SIMD, Vectorization) mathematics libraries including: standard Offeror math libraries, Level 1 BLAS, Level 2 BLAS, Level 3 BLAS, LAPACK version 3.4.2 (or then current), and FFTW 3.3.3 (or then current), for dense single and double precision real and complex data. The Offeror will provide source code for the optimized DGEMM library that is used in their best performing LINPACK calculation.

The Offeror will provide optimized parallel math libraries including: standard Offeror parallel math libraries, ScaLAPACK 1.8 (or then current), Trilinos 11.0 (or then current), and PETSc 3.3 (or then current). The PETSc library will be built to include Hypre 2.12 (or then current), SuperLU 5.2.1 (or then current), and ParMETIS 4.0.3 (or then current).

9.2.5.5 I/O Libraries (TR-2)

The Offeror will provide optimized I/O libraries netCDF 4.4.1.1 (or then current) and HDF5 1.10.1 (or then current). MDtest and MACSIO will be used to test the performance and correctness of these I/O libraries.

9.2.5.6 Machine Learning Libraries (TR-2)

The Offeror will provide optimized machine learning libraries. Where these libraries work in parallel they will use the full capabilities of the CORAL interconnect; the use of the Offeror's MPI library is encouraged. Wherever the Offeror uses optimized low-level libraries to build optimized machine learning libraries, those libraries and their APIs will also be provided and supported. They will be used and exploited by HPC-specific machine-learning toolkits (e.g., LBANN). In addition to the Offeror's own machine learning libraries, provided libraries will include at a minimum: Tensorflow (with Bazel toolchain files), Caffe2 (or then current), Keras (https://keras.io/), scikit-learn (http://scikit-learn.org/), and OpenCV (http://opencv.org/). The Offeror will also provide at a minimum three other common machine learning libraries, which may include any of the following: Neon (http://neon.nervanasys.com/), PyTorch (http://pytorch.org/), MxNet (http://mxnet.io/), Theano (http://www.deeplearning.net/software/theano), Dlib (http://dlib.net/), and CNTK https://github.com/Microsoft/CNTK).

The Offeror will provide a Spark solution and versions of Spark MLlib and GraphX optimized for the CORAL platform. Wherever I/O occurs, optimized machine learning libraries will make effective use of the CORAL I/O subsystem (e.g., during training workloads, and reading and writing checkpoint files).

9.2.5.7 Python (TR-1)

The proposed FEE will support Python consistent with the support provided in the CNOS (see Section 5.2.5).

9.2.5.8 Spack (TR-2)

The Offeror will provide the Spack package manager (https://github.com/LLNL/spack) and will provide Spack build recipes to build all provided math, machine learning, and I/O libraries (see Sections 9.2.5.4 and 9.2.5.5) from source using each of the supported compiler tool chains.

9.2.5.9 Library Support for Multiple Compilers (TR-3)

CORAL finds it useful to be able to install compiler tool chains in addition to those supplied and supported by the Offeror. In order for these compilers to be useful to users, a compatible set of key libraries must also be available, including libraries supplied by the Offeror. The Offeror will propose a

mechanism to support this need. Examples of mechanisms that might satisfy this requirement include, but are not limited to:

- On-site vendor support staff who will build and install Offeror-supplied libraries using site-installed compiler;
- The Offeror will provide complete source and Spack recipes to build Offeror-supplied libraries, and site staff will adapt recipes to the compiler and will build and install the libraries.

10.0 System Management and RAS Infrastructure

This section covers system management and Reliability, Availability and Serviceability (RAS) features, which are crucial to achieving a stable, reliable system.

10.1 Robust System Management Facility (TR-1)

The Offeror will provide a full-functioned, robust, scalable facility that enables efficient management of the CORAL system. This system management capability will run on one or more System Management Nodes (SMNs) and will control all aspects of system administration in aggregate, including modifying configuration files, software upgrades, file system manipulation, reboots, user account management and system monitoring.

10.1.1 System Management Architecture (TR-1)

The Offeror will describe the hardware and software architecture and major components of the system management facility, highlighting features that provide ease of management, operational efficiency, scalability, state consistency (software and hardware) and effective fault detection/isolation/recovery.

10.1.2 Fast, Reliable System Initialization (TR-1)

The Offeror will describe the cold boot process for the full CORAL system, including timing estimates for each phase. The major components of the CORAL system will boot in less than one (1) hour. The boot process should include all infrastructure, hardware, software and any file systems required for the CORAL system to operate as designed, including the CORAL I/O subsystem. Mounting of the I/O subsystem global namespace on all applicable CORAL nodes will add no more than an additional five (5) minutes to the boot process. The system boot will progress without human intervention with the exception of a final release to start batch jobs when all hardware and software is ready.

10.1.3 System Software Packaging (TR-1)

The Offeror will provide all software components of the CORAL system via a single Software Package Management System (SPMS). The SPMS will provide tools to install, to uninstall, to update, to remove, and to query all software components. The SPMS will allow multiple versions of packaged software to be installed in Laboratory-specified locations and used on the system at the same time, and will provide the ability to roll back to a previous software version. The contents of all SPMS packages will be catalogued in the telemetry database (see Section 10.3) on a per-file basis.

10.1.4 Remote Manageability (TR-1)

All nodes of the CORAL system will be 100% remotely manageable, and all routine administration tasks automatable in a manner that scales up to the full system size.

10.1.4.1 Out of Band Management Interface (TR-1)

The CORAL system nodes will provide an Out of Band (OOB) management interface. This interface will be accessible over the system management network. This interface will allow system RAS and system administration functions to be performed without impact to or dependence on the high performance interconnect.

10.1.4.2 Remote Console and Power Management (TR-2)

The Offeror will provide a polled console input/output device for each instance of the operating system kernel that is available via a system-wide console network that scales to permit simultaneous and continuous access to all consoles. The capability to log all console output to logfiles will be provided. Rack PDUs that provide remote on/off switching control of individual outlets via a well-known API are desired.

10.1.5 System Performance Analysis and Tuning (removed)

This requirement removed.

10.1.6 System-wide Authentication/Authorization Framework (TR-1)

The Offeror's proposed CORAL system will provide a common authentication/authorization framework including some means of integrating with external directory services. A user's credentials, once validated, will be honored by all CORAL system components (e.g., FEE, batch system, I/O subsystem, and CNOS). Similarly, a user's privileges, once established, will be enforced by all CORAL subsystems. This framework will integrate seamlessly with the PAM provided in Section 8.1.4.1.

10.2 Reliability, Availability and Serviceability (TR-1)

The Offeror's proposed CORAL system will be designed with Reliability, Availability and Serviceability (RAS) in mind. The Offeror will provide a scalable infrastructure that monitors and logs the system health and facilitates fault detection and isolation (see Section 10.3).

10.2.1 Mean Time Between Failure Calculation (TR-1)

The Offeror will provide the Mean Time Between Failure (MTBF) calculation for each FRU and node type. The Offeror will calculate overall CORAL system MTBF from these statistics.

10.2.2 System Effectiveness Level (TR-2)

Over any four week period, the system will have an effectiveness level of at least 95%. The effectiveness level is computed as the weighted average of period effectiveness levels. The weights are the period wall clock divided by the total period of measurement (four weeks). A new period of effectiveness starts whenever the operational configuration changes (e.g., a component fails or a component is returned to service). Period effectiveness level is computed as operational use time multiplied by max[0, (N-2D)/N] divided by the period wall clock time, where N is the number of CNs in the system and D is the number of CNs unable to run user jobs. Scheduled preventive maintenance is not included in operational use time.

10.2.3 Hardware RAS characteristics (TR-1)

The Offeror will describe component level RAS characteristics that are exploited to achieve a high level of system resilience and data integrity. This description should include methods of error detection, correction and containment across all major components and communication pathways. The Offeror will

describe RAS features of the memory subsystem, including advanced error correction capabilities of DRAM and endurance characteristics of NVRAM, if any, in the proposed solution.

10.2.4 Failure Detection, Reporting and Analysis (TR-1)

The Offeror will provide a mechanism for detecting and reporting failures of critical resources, including processors, network paths, and disks. The diagnostic routines will be capable of isolating hardware problems down to the Field Replaceable Unit (FRU) level.

10.2.5 Power Cycling (TR-3)

Each CORAL system component will be able to tolerate power cycling at least once per week over its life cycle. The CORAL system components should also remain reliable through long idle periods without being powered off.

10.2.6 FRU Labeling (TR-2)

All FRUs will have individual and unique serial numbers tracked by the control system. Each FRU will be labeled with its serial number in human readable text and machine readable barcode.

10.2.7 Scalable System Diagnostics (TR-2)

The Offeror will provide a scalable diagnostic code suite that checks processor, cache, memory, network and I/O interface functionality for the full system in under thirty (30) minutes. The supplied diagnostics will accurately isolate failures down to the FRU level.

10.2.8 Modular Serviceability (TR-1)

The service of system components, including nodes, network, power, cooling, and storage, will be possible with minimal impact and avoiding full-system outage. Hot swapping of failed FRUs will not require power cycling the cabinet in which the FRU is located.

10.2.9 RAS Reporting (TR-1)

All CORAL node types will report all RAS events that the hardware detects. Along with the type of event that occurred, the node will also gather relevant information to help isolate or understand the error condition.

10.2.9.1 RAS Analysis Tool (TR-2)

As the RAS data produced by the system is expected to be large, a RAS analysis tool will be provided. This tool will process and store all RAS data for both real time and future analysis in a telemetry database (Section 10.3). The RAS analysis tool will be capable of generating events based on logs, environmental data, or other RAS events.

10.2.10 Highly Reliable RAS Facility (TR-1)

The RAS facility will have no single points of failure. RAS infrastructure failures will not result in loss of visibility or manageability of the full system or degrade system availability.

10.2.11 Graceful Service Degradation (TR-2)

The Offeror's RAS facility will detect, isolate and mediate hardware and software faults in a way that minimizes the impact on overall system availability. Failure of hardware or software components will result in no worse than proportional degradation of system availability.

10.2.12 Comprehensive Error Reporting (TR-1)

All bit errors in the system (e.g., memory errors, data transmission errors, local disk read/write errors, and SAN interface data corruption), over-temperature conditions, voltage irregularities, fan speed fluctuations, and disk speed variations will be logged by the RAS facility. Recoverable and non-recoverable errors will be differentiated. The RAS facility will also identify irregularities in the functionality of software subsystems.

10.2.13 System Environmental Monitoring (TR-1)

The Offeror will provide the appropriate hardware sensors and software interface for the collection of system environmental data. This data will include power (voltage and current), temperature, humidity, fan speeds, and coolant flow rates collected at the component, node and rack level as appropriate. System environmental data will be collected in a scalable fashion, either on demand or on a continuous basis as configured by the system administrator.

10.2.14 Hardware Configuration Database (TR-2)

The RAS system will include a hardware database or equivalent that provides an authoritative representation of the configuration of CORAL system hardware. At minimum this will contain:

- Machine topology (compute nodes and I/O nodes);
- Network IP address of each hardware component's management interface;
- Status (measured and/or assumed) of each hardware component;
- Hardware history including FRU serial numbers and dates of installation and removal;
- Method for securely querying and updating the hardware database from CORAL system hosts other than the SMNs.

10.3 Integrated Telemetry Database and Analytics Infrastructure (TR-1)

To help understand utilization of all system resources, failure modes and trends, and to correlate the various data sources, the Offeror will provide an integrated telemetry database (ITDB) that facilitates collection and analysis of system management and monitoring data. The database will be scalable and accessible through a web interface and command-line interface. The Offeror will provide authentication and authorization mechanisms that enable subsets of data to be made available to different groups of people (e.g., system administrators, operators, end-users, and researchers). The database will provide rich search capabilities and advanced analytics that enable authorized users to perform complex event correlation and problem determination as well as to avoid outages proactively and to optimize system operation and performance.

10.3.1 Data Collection (TR-1)

The ITDB will provide efficient, scalable collection of important data about the health and performance of the CORAL system. This database will include information about all system components including the CN partition, FENs, management/service/utility nodes, network switches and all components of the I/O subsystem. Administrators will be able to tune the periodicity of log collection where appropriate, and to turn off data streams based on site requirements. The ITDB will be extensible: it will allow and support ingest of additional streams of data. The ITDB will support ingestion of data through multiple interfaces,

e.g., text files, syslog(), database APIs, and restful APIs. All data sources will include a common, high-precision timestamp to allow correlation between data sources. The ITDB will collect, at a minimum:

- System logs (Section 8.1.3)
- Console output (Section 10.1.4.2)
- Diagnostics results (Section 10.2.4)
- Hardware RAS information (Section 10.2.12)
- System environmental data (Section 10.2.13)
- SRM logs (Sections 8.3.5.3 and 8.3.6.2)
- Interconnect counters (Section 7.5)
- I/O subsystem logs and statistics (Sections 6.1.7.1 and 6.2.7.1)
- Power measurements of the compute partition and interconnect (Sections 5.1.6 and 8.3.6.2).

10.3.2 ITDB Collection and Storage Infrastructure (TR-1)

The ITDB will provide a scalable data repository to ingest, to store, to index and to process log data in real-time. Recent data, minimally one month's worth, will be indexed and available for fast, online querying. Historical data will be stored in the data repository and will be available for offline analysis. The solution will include the necessary database and storage infrastructure, including hardware, for at least five years of operation.

10.3.3 Data Analysis and Visualization (TR-1)

The ITDB will provide a well-defined interface to query, to analyze and to visualize collected data. It will include a suite of analytics tools to process both the real-time data streams and historical data. Analysis methods will include standard statistical analysis (e.g., min, max, median, average, sum, percentile, standard deviation, PDF, CDF, histograms, top-N), binning, autocorrelation, cross correlation, sliding-window analysis, visual analysis by way of line, bar, scatter and pie charts as well as basic data cleansing and data mining operations. Frameworks to perform more intensive processing of historical data will also be provided (e.g., Hadoop and Spark). The ITDB will provide an interface to create reports as well as custom dashboards for visualization.

10.3.4 Alerts (TR-1)

The ITDB will provide a mechanism for administrators to create alerts based on single events (e.g., I/O rate or node failure rate), correlation of multiple events, and on the results of custom analysis.

10.3.5 Authentication and Access Control (TR-1)

The ITDB will support, at a minimum, basic Linux authentication and authorization functions, as well as the ability to replace the standard authentication mechanism with a site-specific pluggable authentication module (PAM) as described in Section 8.1.4.1. The ITDB will provide the ability for an administrator to implement role-based access to the data.

11.0 CORAL Maintenance and Support

The Offeror will propose several support models, described below, to meet the needs of CORAL. Regardless of which model is selected, Laboratory hardware and software development personnel will work collaboratively with the Offeror as required to solve particularly difficult problems. A problem escalation procedure (Section 11.3) will be invoked when necessary. Should any significant hardware or

software issue arise, the Offeror is expected to provide additional on-site support resources as necessary to achieve timely resolution.

The requirements described in this section apply to the main CORAL system including the I/O subsystem.

11.1 Hardware Maintenance (TR-1)

11.1.1 Hardware Maintenance Offerings (TO-1)

The Offeror will supply hardware maintenance for the CORAL system for a five-year period starting with system acceptance. The Offeror will propose, as separately priced options on a per year basis, at least the following two hardware maintenance options. If desired, the Offeror may propose additional hardware maintenance options that might be of interest to CORAL from a cost efficiency standpoint.

9x5: Support of hardware will include 9x5xNBD (Next Business Day) response times during the warranty and post-warranty period. The 9x5xNBD includes service personnel onsite from 8:00am to 5:00pm local time, Monday through Friday who will respond to a service call within one hour after the call information is received. The service personnel will meet the clearance requirements for CORAL support personnel described in Section 11.6.

12x7: Support for hardware will be 12 hours a day, seven days a week, (0800-2000 Laboratory local time zone) with one hour response time. In the 12/7 maintenance model, Laboratory personnel may provide on-site, on-call 24x7 hardware failure response inside or outside the defined 12x7 maintenance window. Laboratory personnel will be trained and allowed to perform first-level hardware fault diagnosis and repair actions. During the defined maintenance window, the Offeror will provide hardware fault diagnosis and fault determination as well as second-level hardware fault diagnosis and fault determination as needed for any Laboratory-performed responses outside the defined maintenance window. Laboratory personnel will utilize Offeror-provided on-site parts cache (Section 11.1.2) so that FRUs can be quickly repaired or replaced and brought back on-line. If Laboratory personnel cannot repair failing components from the on-site parts cache, then Offeror personnel will be required to make on-site repairs. Redundant or non-critical components should carry 9x5 Next Business Day (NBD) support contracts.

11.1.2 On-site Parts Cache (TR-1)

The Offeror will provide an on-site parts cache of FRUs and hot spare nodes of each type proposed for the CORAL system. The size of the parts cache, based on Offeror's MTBF estimates for each component, will be sufficient to sustain necessary repair actions on all proposed hardware and keep them in fully operational status for at least one month without parts cache refresh. The required size of the parts cache will be recalculated at least every six months. The Offeror will resupply/refresh the parts cache as it is depleted for the five year hardware maintenance period. System components will be fully tested and burned in prior to delivery in order to minimize the number of "dead-on-arrival" components and infant mortality problems.

11.1.3 Engineering Defect Resolution (TR-1)

In the case of system engineering defects, the Offeror will address such issues as soon as possible via an interim hardware release as well as in subsequent normal releases of the hardware.

11.1.4 Secure FRU Components (TR-1)

The Offeror will identify any FRU in the proposed system that can persistently hold data in non-volatile memory or storage. The Offeror will deliver a Statement of Volatility for every unique FRU that contains

only volatile memory or storage and thus cannot hold user data after being powered off. The final disposal of FRU with non-volatile memory or storage that potentially contains user data will be decided by individual sites.

11.1.4.1 FRU with Non-Volatile Memory Destroyed (MO)

FRU with non-volatile memory or storage that potentially contains user data shall not be returned to the Offeror. Instead, the Laboratory will certify to Offeror that the FRU with non-volatile memory or storage that could potentially contain user data has been destroyed as part of the Offeror's RMA replacement procedure.

11.2 Software Support (TR-1)

The Offeror will supply software maintenance for each Offeror-supplied software component starting with the CORAL system acceptance and ending five years after the CORAL system acceptance. Offeror support for supplied software that is critical to the operation of the machine, including, but not limited to, system boot, job launch, I/O and RAS systems will be provided 24 hours a day, seven days a week with one hour response time during Laboratory business hours. Other supplied software will be 9x5 Next Business Day (NBD) support. Offeror provided software maintenance will include an electronic trouble reporting and tracking mechanism and periodic software updates. Any bug fixes developed by Laboratory personnel will be provided back to the selected Offeror.

11.2.1 Software Feature Evolution (TR-1)

The Offeror will support new software features on the delivered hardware for the full term of the warranty or maintenance period covered by an exercised option, with the exception of when new software features are specific to a different hardware platform. For software produced by the Offeror, new features will appear on the delivered hardware at the same time as provided on the Offeror's other commercial platforms. For software not produced by the Offeror, new features will appear on the delivered hardware within 6 months of general availability.

11.2.2 Compliance with DOE Security Mandates (TR-1)

DOE Security Orders may require the Laboratories and/or their Subcontractors to fix bugs or to implement security features in vendor operating systems and utilities. In this situation, the Offeror will be provided written notification of the changes to DOE Security Orders or their interpretation that would force changes in system functionality. If the request for change would result in a modification consistent with standard commercial offerings and product plans, the Offeror will perform the change. If the change is outside the range of standard offerings, the Offeror will make the operating system source code available to the Laboratories (at no additional cost, assuming the Laboratories holds the proper USL and other prerequisite licenses) under the terms and conditions of the Offeror's standard source code offering.

11.3 Problem Escalation (TR-1)

The Offeror will describe their technical problem escalation mechanism in the event that hardware or software issues are not being addressed to the Laboratories' satisfaction.

11.4 On-Line Documentation (TR-2)

The Offeror will supply local copies of documentation, preferably HTML or PDF-based, for all major hardware and software subsystems. This documentation will be viewable on site-local computing systems with no requirement for access to the Internet.

11.5 On-site Analyst Support (TO-1)

The Offeror will supply two on-site analysts to each CORAL site that procures one of its systems. One on-site systems programmer will be highly skilled in Linux systems programming and Offeror-supplied I/O subsystem software and will support Laboratory personnel in providing solutions to the current top issues. One on-site application analyst will be highly skilled in parallel application development, debugging, porting, performance analysis and optimization. The Laboratories may request additional on-site analysts, which will be priced separately.

11.6 Clearance Requirements for CORAL Support Personnel at LLNL (TR-1)

The proposed CORAL system will be installed in a Limited Access Area Vault Type Room (VTR) at LLNL. Offeror's support personnel will need to obtain DOE P approval (i.e. pass a preliminary background investigation) for repair actions at LLNL and be escorted during repair actions. USA Citizenship for Offeror support personnel is required. LLNL may pursue Q clearances for qualified Offeror personnel.

12.0 CORAL Facilities Requirements

The requirements described in this section apply to the complete CORAL system.

12.1 ANL Facilities Overview

The potential 2022-2023 system at Argonne will be sited in the data center in the Theory and Computing Sciences Building (Building 240) on the Argonne Campus in Lemont, IL, hereinafter referred to as TCS-DC. The floor area for siting this system is shown in Figure 12-1. It has an approximate area of 15,000 ft^2, which is approximately 122' by 122'. The TCS-DC floor is a 48" raised floor over a concrete slab. The raised floor is an FS Series raised floor system that can support a concentrated load of 3000 lbs. and a uniform load rating of 700 lbs./ft^2 when the under-floor steel panel is unaltered.

The planned power budget for the system and associated performance data store is 30MW.

The planned cooling budget for the system is 8,600 tons. Argonne prefers warmer water, and is interested in solutions that meet the ASHRAE Technical Committee 9.9 Liquid Cooling Guidelines, W3 level at a minimum.

Air cooling will be available at TCS-DC through the plenum beneath the raised floor. TCS-DC will have around 250,000 CFM available in the 2022-2023 time frame.

All unpacking/uncrating of equipment may need to be done at the loading dock or in an adjacent corridor. The current loading dock for the TCS-DC is 600 ft^2 area (30' x 20') with 2x overhead roll-up doors (8' by 10'), that will accommodate 1x tractor semi-trailers loading/unloading at a time. A new loading dock will be available for this 2022-2023 system and is expected to have similar or larger access.

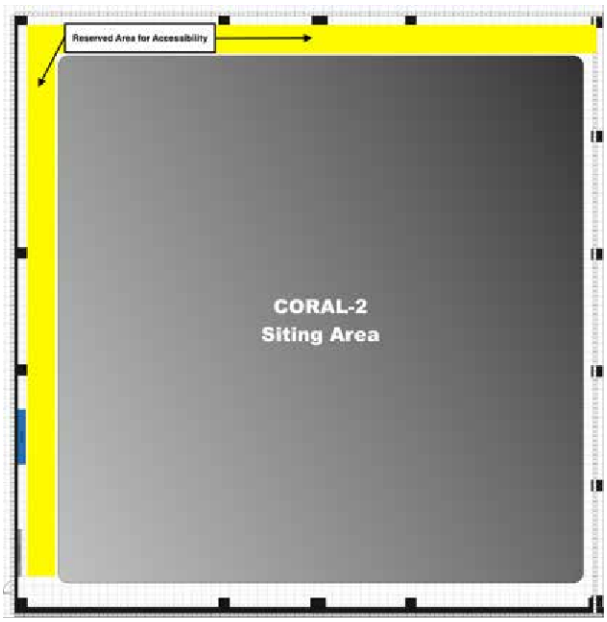

Figure 12-1: Argonne CORAL 2022=2023 Siting Area

12.2 LLNL Facilities Overview

B-453 is an existing facility at LLNL that will be used for CORAL. The new machine will be sited in the west room of B-453. B-453 has approximately 47,500ft^2 of 48" raised floor space and 45MW of power that will be scaled to 85MW by 2022. Up to 40MW is budgeted for the CORAL system. B-654 is another facility that may house a smaller CORAL system option as described in Section 3. This facility has approximately 6000 ft^2 and 6MW of power.

The heat exchange for air cooling exceeds 1.0M CFM in B-453's west room and exceeds 350,000CFM in B-654 for ancillary components of the CORAL system. B-453 currently has 10,000 tons of warm water cooling available and will double the capacity by 2022 At least 13,000 will be available for CORAL. B-654 does not have any warm water cooling available from the campus, it only has 800 gpm of water from an evaporative cooling process water loop, of which 600 gpm are available for this type of system.

The B-453 raised floor has a 250 lbs./ft^2 loading with the ability to accommodate up to 500 lbs/ft^2 through additional floor bracing. Rolling weights cannot exceed 2000 lbs./ft^2. B-654 has a floor rating twice that of B-453. Both B-453 and B-654 are unique facilities in that their construction consists of single story two level computer rooms. This design affords the capability of siting a machine with higher weight capacities on the slab on grade of the first level of the computer room.

B-453 has the most restrictive path of travel to the second level of the machine room. The smallest door opening is 7' 10" (W) x 7' 10" (H). Both facilities have a freight elevator capacity of 10,000 lbs.

The CORAL system in B-453 will be installed and located inside a Limited Access Area in a VTR. Access to the room will only be provided to authorized personnel under escort. On-site personnel will be required to submit DOE P-clearable applications for access; applications must be approved prior to entry into this facility. Proposals should indicate if the on-site team has members that are other than U.S. citizens. Physical access to this computer facility by foreign nationals from sensitive countries (www.llnl.gov/expcon/sensitive.html) is not allowed. These restrictions do not apply to B-654.

Head disk assemblies (HDAs) from disks or other non-volatile memory components used for classified processing cannot be taken off-site or returned to the factory.

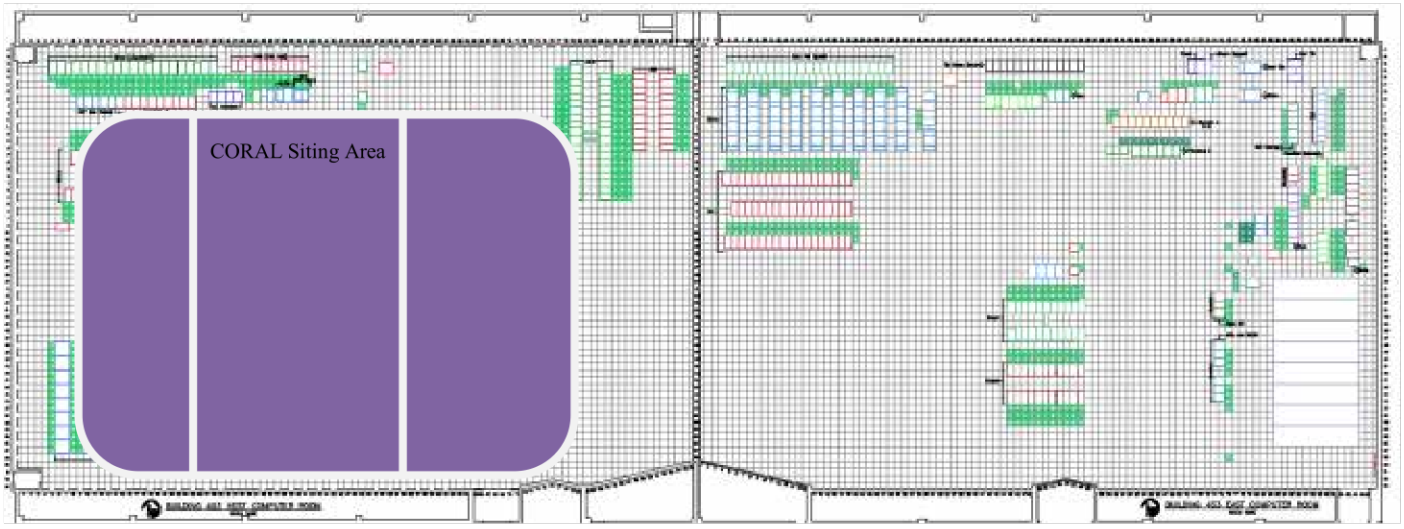

Figure 12-2: CORAL siting locations within LLNL B453 computer floors

12.3 ORNL Facilities Overview

12.3.1 ORNL Facility - Space

The CORAL system at ORNL will be installed in Building 5600 on the ORNL campus in Oak Ridge, TN. Within a single computer room in the facility, approximately 20,000 ft^2 of gross square footage is available for the CORAL system. The contiguous space that allows freight aisles and meets minimum clearance requirements for electrical distribution systems and ingress/egress is approximately 120'x120'. It is not clear span. There are a total of eight load-bearing columns on approximately 40' centers. The floor space is on a 36" raised floor over concrete slab. A drop-down ceiling that is 10' above the raised floor conceals a 4' plenum above. Total deck to deck height is approximately 17'.

The OLCF Data Center raised floor is a Tate Access Floor Bolted Stringer Understructure System that can support a design point load of 1500 lbs, a safety point load of 3000 lbs, and a rolling point load of 600 lbs. Finished equipment installation allows a uniform load rating of 500 lbs/ft^2.

The Offeror must plan for all technical load to be included in this footprint, including storage/file systems, infrastructure or management systems, network equipment associated with the system interconnect, and any supporting mechanical or electrical distribution items that are part of the solution.

The room air will be continuously filtered with MERV 8 filters as recommended by ANSI/ASHRAE Standard 127-2007 (ASHRAE 2007) or later.

Air entering a data center will be filtered with no less than MERV 11 filters as recommended by ASHRAE (2009b).

Gaseous corrosivity levels are actively monitored using copper and silver coupons, and the data center will be maintained within G1 (Mild) guidelines for the presence of corrosive gases, per ISA-71.04.

Access to the facility is all on-grade, via a loading dock.

Delivery and laydown space at ORNL is very limited especially where large systems are involved. ORNL has minimal conditioned space to store hardware prior to installation activities. The Offeror should

strongly consider obtaining the use of conditioned space in the local area to support the planned delivery and installation effort.

12.3.2 ORNL Facility - Electrical Distribution

The planned power budget for the CORAL system and associated I/O subsystem is constrained to no more than 40MW. The electrical distribution method for the Offeror's compute solution should be based on 3-phase wye (Y) 277/480VAC. The maximum size of any circuit supplying a compute rack will be 200A.

Delivery of electrical services should be assumed to be from overhead. Offeror may assume that ORNL will provide overhead cable tray, overhead busway, or similar. Offeror is responsible for cable management from the overhead electrical distribution system to their equipment. Offeror is responsible for cable management associated with any cabinet to cabinet network connections. Direct-wire connections to the Offeror's equipment will be provided by ORNL.

12.3.3 ORNL Facility – Mechanical Distribution

ORNL will maintain the operating environment within the data center in accordance with Class A1, as defined in the ASHRAE Thermal Guidelines for Data Processing Environments.

It is assumed that the Offeror will use a facility water supply temperature described by ASHRAE Technical Committee 9.9 Liquid Cooling Guidelines, as Liquid Cooling Class W3, or higher for the compute racks. The CORAL system's compute racks must operate without the direct use of chilled water for all hours of the year given the sites' climatic zones.

ORNL will provide chilled water as part of fresh air pressurization, air conditioning, and relative humidity control of the room. For support cabinets, the Offeror's solution is to utilize the W3 water when possible, but provide means to return rack discharge air to acceptable conditions using Facility chilled water when the W3 water is too warm. During normal operation, no air exchange will be allowed between the interior and exterior of the compute rack unless the specific site can accommodate this arrangement. During short-term maintenance, less than 2% of the system load may escape as parasitic heat to the data center ambient.

Delivery of mechanical services (W3 water) to the Offeror's solution for ORNL should be assumed to be from under the floor. Cooling distribution units (CDUs) for rack-level flow control of the cooling water is the responsibility of the Offeror, who can provide operating envelopes for supply temperatures, pressure, and flow for the primary Facilities cooling system as well as makeup water quality requirements. ORNL will provide suitable delivery mechanisms, with final design for those based on the Offeror's solution. The Offeror's CDU is to provide variable secondary water flow based on changing primary temperature and flow as well as changes in cooling demand by the connected IT equipment. The CDU's primary flow control valve is to be a fail closed, two way modulating valve with pressure independent flow control and balancing.

Should the Offeror's solution not include CDUs, rack-level flow control is to be included with the same valve characteristics as described above for the CDU's control valve. If Facility W3 cooling water is to be used at the rack level with no heat exchanger, the water quality requirements are to be no tighter than that of a typical closed loop hydronic system.

The Offeror's cooling equipment is to have a communications interface between it and the Facility. Table 12-1 reflects the types of data expected of this communications link. The Offeror will describe the actual data available on this communications link based on the proposed system.

Table 12-1: Cooling Equipment Monitoring Interface

Name	Where	Typical Provider	Frequency of measurement	Accuracy/Units
Water flow	System	Facility	Once every 30 sec	+/- 5% Liter/min (gal/min)
	CDU	Negotiated	Once every 30 sec	+/- 10% Liter/min (gal/min)
Thermal data	System or branch	Facility	Once every 60 sec	+/-1° C (1.8 °F)
	CDU	Negotiated	Once every 60 sec	+/-1° C (1.8 °F)
	Rack	HPC system	Once every 60 sec	+/-1° C (1.8 °F)
	Node	HPC system	Once per sec	+/-2° C (3.6 °F)
	Component	HPC system	Once per sec	+/-2° C (3.6 °F)
Power	System	Facility	Once per sec	+/- 5% Watts
	CDU	Negotiated	Once per sec	+/- 5% Watts
	Rack	HPC system	Once per sec	+/- 5% Watts
Dew Point Temperature	System	Facility	Once per 60 sec	+/-2° C (3.6 °F)
	Branch, rack or cabinet	Negotiated	Once per 60 sec	+/-2° C (3.6 °F)

Table 12-1: Cooling Equipment Monitoring Interface (continued)

Name	Where	Typical Provider	Frequency of measurement	Accuracy/Units
Pump Speed	System	Facility	Once per 60 sec	+/-3 % of full speed
	CDU	Negotiated	Once per sec	+/-3 % of full speed
Pressure Differential	System	Facility	Once per 5 sec	+/-10 kpa (1.5 PSI)
	Branch, rack or cabinet	Negotiated	Once per 5 sec	+/-10 kpa (1.5 PSI)
Valve Position	System	Facility	Once per 60 sec	+/- 5% (% Open)
	Branch	Negotiated	Once per 60 sec	+/- 5% (% Open)
	CDU	Negotiated	Once per 60 sec	+/- 5% (% Open)
	Rack	HPC system	Once per 3 sec	+/- 2% (% Open)

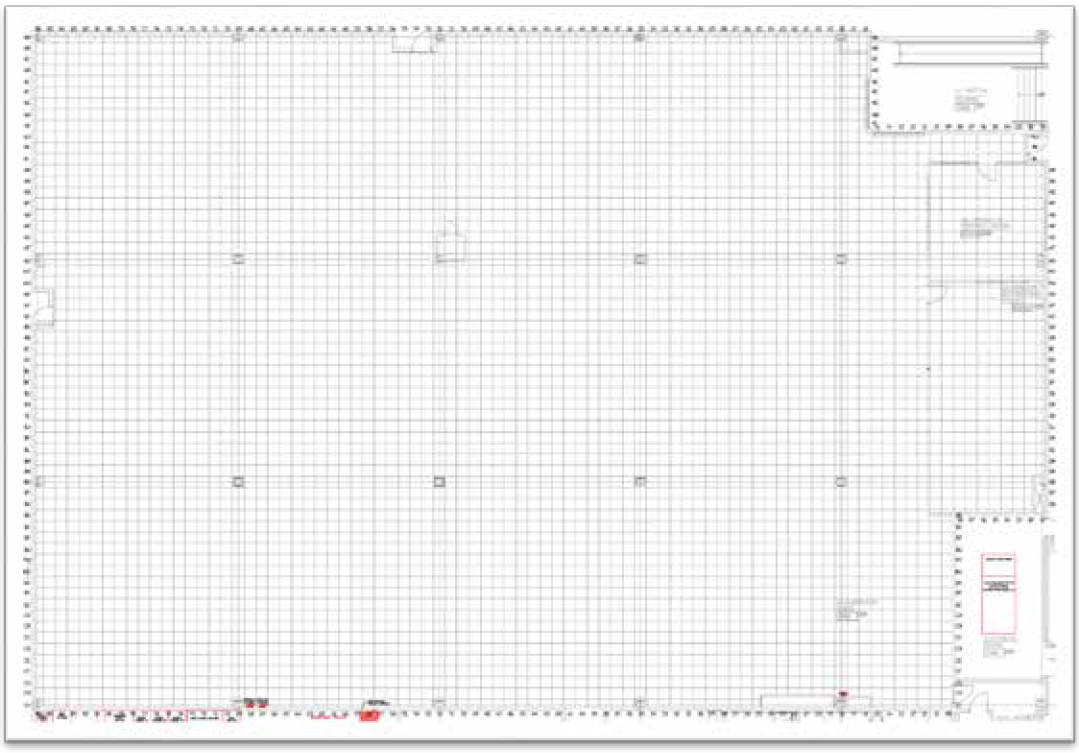

Figure 12-3: CORAL Siting Location within ORNL Building 5600

12.4 Laboratories Facilities Overview Summary

Table 12-2 summarizes the space, power and cooling constraints of the three facilities.

Table 12-2: Overview of Facilities Space, Power, and Cooling

	ANL	LLNL		ORNL
	TCS-DC	**B-453**	**B-654**	**5600**
Max Floor Load (static) (PSF)	700	500	500	500
Max Floor Load (rolling) (PSF)	1800 lb per panel	2000	2000	1500
Maximum Power (MW)	30	40	6	40
Available Liquid Cooling (TONS)	8,600	13,000	TBD	TBD
Available Air Cooling (CFM)	250,000	1,000,000	350,000	TBD
Floor Space (SQ FT)	15,000	15,000	6,000	20,000 (gross)

12.5 Power & Cooling Requirements (TR-1)

Without compromising performance objectives, the Offeror will minimize the power and cooling required by the proposed systems. The Offeror will describe how its proposal fits within the power budget. The Offeror will provide a detailed estimate of the total amount of power in kW (kilowatts) and kVA, and cooling in either refrigeration tons or BTU (British Thermal Units), required by the complete CORAL system. The estimate will describe the power and cooling loads for the individual racks (by rack type) and for each substantive component of the system. At a minimum, the Offeror will describe power and cooling estimates anticipated during special-purpose / diagnostic / benchmarking efforts that may best approach the design TDP, and maximum, typical / notional, and idle (minimum) operation.

The Offeror will describe the power factor, phase balancing, and harmonics management for their AC power systems.

The Offeror will provide rack mounted power supplies, PDUs, or similar items. that are rated to withstand 100kA available fault current. This requirement may be accomplished through the use of fuses that are series rated with any downstream electrical devices in the racks. Fuse protection external to the racks will be provided by Offeror if the rack mounted equipment is not rated to withstand 100kA of fault current. The Offeror will provide fusing equipment and an interconnecting circuit between fuses and each computer rack if fuses are mounted externally to the racks. Pin and sleeve connectors for 480VAC connections will not be used.

The facility will make direct connection of electrical service to the Offeror's equipment. To avoid requiring multiple lug connections for each phase at the upstream overcurrent device, paralleling of conductors in raceway that supply compute racks will not be allowed.

The connections from the Offeror's equipment to the facility's power distribution system will use liquid tight flexible metal conduit, flexible metal conduit, MC cable, or other permanent wiring method.

All equipment must be NRTL-certified.

12.5.1 Minimal electrical and mechanical connections (TR-2)

The Offeror's solution will minimize total number of electrical and mechanical connections that are required, both inside and outside the racks. Cooling water connections will be designed for minimal pressure drop and will not include threaded connections.

12.5.2 Power and cooling utilization data collection (TR-2)

At every electrical or mechanical connection from the facility infrastructure to the Offeror's solution, the Offeror will provide a mechanism for providing power and cooling utilization or consumption data for that connection. Each facility has a site-specific collection system that is part of its industrial control system (ICS). The Offeror will describe the interface from the solution to the ICS in terms of protocol(s), and the use of wired or wireless connections.

12.5.3 Rack-Integrated In-line PDU (TR-2)

The Offeror may propose a power distribution solution that supports more than one compute cabinet, using an in-line power distribution unit. If the Offeror does not provide an in-line PDU, the Offeror will provide a solution that minimizes the number of connections to each rack. The design of any PDU will provide protection of each branch circuit in that PDU and remote management of each branch circuit in that PDU. Regardless of the PDU design (in-line serving more than one rack, or in-rack), the Offeror will

provide a connection point for each device that the facility can permanently terminate to the appropriate building circuit.

12.5.4 Fault-Tolerant Power Distribution (TR-2)

For ANL, the Offeror will also describe an option, if available, that can provide load balanced and/or fault tolerant connections that can reduce the risk of a single point of failure within the power supply/distribution system.

12.5.5 Tolerance of Power Quality Variation (TR-1)

The design of the power system for the CORAL system will be tolerant of power quality events. The Offeror will describe the tolerance of their power system to power quality events, in terms of both voltage surge and sag, and in duration. All power supplies must be tolerant to voltage sag in accordance with the latest version of SEMI F47. Computer power at all three sites is reliable and clean, but not conditioned. There is no uninterruptible power available for the CORAL compute system. 208-240VAC components of the Offeror's solution, which might include the I/O subsystem, network components, and other infrastructure, may be supported by UPS systems in dual-fed configurations that can provide tolerance (short-term ride-through) to power quality events that exceed the SEMI F47 specification.

12.5.6 Power Factor and Harmonic Current Requirements (TR-1)

The power factor of the computer racks when operating at benchmark levels shall be $\geq$ 0.98. The power factor of 208-240VAC components of the Offeror's solution, which includes the I/O subsystem, network components, and other infrastructure, will be $\geq$ 0.95.

At benchmark power levels, the maximum total harmonic current and the maximum individual harmonic current levels will meet recommendations provided in Table 2 of IEEE STD 519-2014 for $I_{sc}/I_l \leq 20$.

12.5.7 Cooling Requirements (TR-1)

All air and liquid cooling required for each system will be listed separately, in relation to the heat load created by the operation of each system. The Offeror should understand that both air and liquid cooled systems will reside in the same room. For the portion of the racks that requires air cooling, the Offeror will specify the environmental conditions required in CFM, temperature, and humidity. The Offeror will specify the cooling envelopes required by each rack type, the allowable flow and temperature excursion magnitudes and durations, thresholds for flow and temperature at any warning, alarm, and critical/shutdown/throttling points in the envelope, and how any of the previously mentioned items may change at the idle, typical, and maximum operating points.

12.5.8 Liquid Cooling Solution Description (TR-1)

The Offeror will fully describe the liquid cooling apparatus and all implications for siting and facilities modifications (e.g., water connections, flow rates, temperature, humidity, pressure, quality, chemistry, and particulates). The solution will not preclude use of existing piping systems, which may include plastic, polypropylene, ductile steel, stainless steel, copper, and epoxy coated materials.

12.5.9 Liquid Cooling Temperature (TR-1)

The liquid temperature will be supplied at temperatures typical to industrial facilities. For example, chilled water temperature from chillers typically ranges from 42°F to 60°F while condenser water and LCW temperatures from cooling towers typically ranges from 75°F to 90°F. Where available, the solution may

use warm water cooling and reject the heat into the facility water loop. The Offeror will fully describe the range of operating conditions for the proposed solution.

12.5.10 Refrigerant-based Liquid Cooling Solution (TR-2)

The Offeror may propose an optional refrigerant-based liquid cooling solution. For solutions that use a refrigerant as part of their cooling solution, the Offeror will describe any toxicity, safety, handling, or environmental concerns that are pertinent to the operation of the system.

12.5.11 Hot Water Rejection Solution (TR-2)

The Offeror may propose an option for hot water heat rejection for the use of campus heating or power generation.

12.5.12 Alternative Integrated Cooling Solution (TO-1)

The Offeror will propose a heat exchange solution for the liquid cooling that is integrated into the design of the platform such that the facility is not required to install chillers.

12.6 Floor Space Requirements (TR-1)

The Offeror will provide a proposed floor plan for a CORAL system that fits into each Laboratory's site for their CORAL system as described in Sections 12.1, 12.2, and 12.3. The floor plan will show the placement of all system components as provided by the Offeror.

12.7 Rack Weight Requirements (TR-1)

The Offeror will propose a that it will not exceed the floor load rating of the facility.

12.8 Cable Management Requirements (TR-1)

The cable management system external to the cabinets may be accommodated above or beneath the raised floor. Cable tray sizes will be sized in such a way to accommodate shared cable trays. Cable management configuration will assume no more than 40% fill, non-conductive materials, comprise solid-bottom tray, and will ensure that all minimum bend radius requirements are met. When the cable management solution is overhead, the cable management system will be capable of having modesty panels so that the cable is not exposed. All cables will be contained in cable trays supplied by the Offeror.

12.9 Physical Access Requirements (TR-1)

The CORAL systems will be installed and physically located inside controlled access areas. The Laboratories will only provide access to these areas for authorized personnel. All on-site personnel will be required to submit applications for access and to be approved by standard Laboratory procedures prior to entry into the facilities. Offeror personnel that are not U.S. citizens may be further restricted, from both physical and system access, in accordance with the specific requirements of each facility.

Remote access is subject to the terms of the specific facility. The Offeror will understand and accommodate any further restrictions as specified in the individual laboratory sections above.

On-site space will be provided for personnel and equipment storage. The Offeror will describe the anticipated volume of equipment and supplies that must be accommodated as part of their maintenance schedule and plan.

12.10 Safety Requirements (TR-1)

Offeror personnel will practice safe work habits, and comply with all associated Laboratory Environment, Safety and Health (ES&H) requirements.

CORAL will allow any component of the machine to be serviced, repaired, or replaced in a de-energized state without disabling the operation of more than 5% of the machine. Any de-energized component will completely isolate all subsidiary components, through hardware and not software (i.e., on-off switches or switch-rated circuit breakers), without any potential for re-energization.

12.11 Safety and Power Standards (TR-1)

All equipment proposed by the Offeror will meet industry safety, and appropriate power quality and power supply standards. Equipment that is supplied by 480VAC systems shall be rated for +10% nominal voltage.

12.12 Rack Seismic Protection (TR-2)

At LLNL, system racks will be seismically qualified (in accordance with IEEE 344, ICC AC 156, or similar) and will be appropriately anchored.

12.13 Site Preparation Plan (TR-1)

Each site anticipates the need to complete substantial site preparation activities to accommodate a CORAL system. The Offeror will provide, in a timely fashion, site preparation instructions to the Laboratories delineating all site preparation work necessary to install and to operate the systems, as configured in the subcontract. The site preparation plan will detail schedule as well as technical requirements. The milestones and delivery schedule for the Site Preparation Plan are given below.

Table 12-3: Site Preparation Plan Milestones

Date	Milestone
Within 30 days of subcontract award.	Preliminary site preparation plan, includes at a minimum: high-level power and cooling requirements
No later than one year prior to the delivery of the first rack	Power and cooling infrastructure requirements finalized; updated site preparation plan delivered
No later than 6 months prior to the delivery of the first rack	Floor loading requirements finalized; updated site preparation plan delivered
No later than 3 months prior to the delivery of the first rack	Cable management requirements finalized; updated site preparation plan delivered
No later than one month prior to the delivery of the first rack	Complete, final site preparation plan site preparation plan delivered

13.0 Project Management (TR-1)

Documents described in this section are not required in the RFP response; however, the Offeror will confirm its commitment: 1) to include the following project management approaches and elements in its execution of any CORAL subcontract awarded; and 2) to provide the associated documentation by the required times and through a reliable and easily accessible mechanism that supports change control. The Offeror will provide in its RFP response a set of milestones for the deliverables in this section as described in the **Key Build Phase Milestone Dates.**

If the project is to succeed, there must truly be a "partnership" among all involved that goes beyond an ordinary vendor-customer relationship. The separate multi-year collaborative R&D effort by the selected Offeror and CORAL will help mitigate some risks. As the selected Offeror-CORAL partnership continues in the build and deployment phase, ultimately, the selected Offeror is responsible for the successful integration of all elements to satisfy the requirements of this procurement. Both CORAL and the selected Offeror must also recognize this acquisition as a primary institutional commitment. This project management approach is designed to help the Offeror successfully meet its commitment, to help the CORAL laboratories to track the project, and to help CORAL and the selected Offeror to understand and to mitigate risks successfully.

The specific detailed planning, effort tracking, and documentation requirements for the development, manufacturing, installation and support efforts that will be delivered as part of the subcontracts are delineated in the following sections.

Key Planning Deliverables

The Offeror will develop, deliver, submit for approval and maintain the following Planning Deliverables. Some of these plans are described in more detail below. Initial versions and updates of these plans shall be provided in specific time frames as discussed throughout this section. Each of the plans and any revisions will be submitted for comment and approval to the Laboratories project managers.

- Software License Agreement completion;
- Project Liaison Assignments: Offeror Project Manager, System Architect, Executive Liaison, Account Representative, and Software Liaison;
- Plan of Record, including Hardware and Software Schedule, and Project Milestones;
- Risk Management Plan;
- Collaboration Plan;
- Change Management Plan;
- Communication Plan;
- Quality Assurance and Factory Test Plan;
- Full-Term Hardware Development Plan
- Full-Term Software Development Plan
- Site Preparation Guide;
- Installation Process Plan;
- System Administration Guide;
- Maintenance and Support Plan.

Project Meetings and Performance Reviews

Upon subcontract award, the project meetings and performance reviews described below shall commence. The selected Offeror will submit a Quarterly Project Status Report at least five working days before each quarterly review. The report will provide the status of all work breakdown structure tasks and milestones in the critical path. It will also contain narrative descriptions of anticipated and actual problems, solutions,

and the impact on the project schedule. Numbered action items will be taken, assigned, logged, and tracked by the Offeror. The minutes of all project reviews will be recorded in detail by the selected Offeror and provided to the Laboratories for approval within 5 working days after the review.

Table 13-1: Project Meetings and Performance Reviews

Purpose	Subcontractor Deliverables	End Date
Monthly Project Teleconference	• Project status and issues updates • Updated project action item list and assignments • Updated schedule and critical path	Final acceptance
Quarterly face-to-face Review	• Quarterly status report • Plan of record status • Risk management status • Collaboration status • Minutes of progress review • Performance information (% complete) for all tracked tasks	Final acceptance
Executive Review • Held at every quarterly review	• Executive progress report • Issues updates	Final acceptance
Site Preparation and Operations Planning • Monthly at a minimum throughout the full term of the contract and more frequently closer to delivery and installation • Pre-installation at Laboratory • As needed during installation and testing	• Site preparation, status, issues, action items, and assignments • Updated Installation Plan and/or Installation Guide (as indicated)	Final acceptance

Project Working Groups

As described above, the CORAL contracts must represent a partnership that is committed to delivering the most useful system possible. Upon subcontract award, the selected Offeror and CORAL will assess the project for areas in which deep collaboration are necessary to ensure meeting that goal. The partnership will form working groups (WGs) for these topics. Each WG will interact in regard to all details of the technical topic; the selected Offeror will not attempt to limit the scope of these interactions. Specific details include NRE deliverables related to the technical topic and deployed software and hardware in the systems being built. The WGs will serve as a key conduit to identify, to refine and to understand CORAL requirements in detail and to ensure that the delivered system meets those requirements to the greatest extent possible. WGs will establish a regular schedule for electronic meetings (e.g., telecons). Each Quarterly face-to-face meeting will include WG breakout sessions that will last from half of a day to two full days. WG breakout sessions will be determined by project management, including CORAL and selected Offeror representatives. Project management will regularly assess WG progress and identify topics for which WGs are no longer required or additional topics for which new WGs are needed.

Key Build Phase Milestone Dates (TR-1)

Offeror will provide the Laboratories, in its proposal response, a proposed set of milestones for this section and, for each milestone, proposed associated payment that is applicable to Offeror's proposed development and deployment timeline and methodology. Offeror is encouraged to identify milestones for each year of the project that merit revenue that Offeror can legally recognize in that year.

Prior to award, CORAL and the Offeror will finalize the list of Key Build Phase Milestone Dates, including dates for necessary Go/No-Go decisions. Following is a list of the kinds of key dates of importance to CORAL. Other key dates may be needed for phased installations or deployments featuring major upgrades during the subcontract. Early completion is highly desired.

- Project Liaisons assigned;
- Plan of Record complete;
- CORAL early system access begins;
- If applicable, late-binding technology decisions;
- Build Go/No-Go, and decision to exercise any proposed system options;
- IO Subsystem Go/No-Go decision to exercise any proposed tier options;
- On Site Support Personnel arrive on site, e.g., hardware, storage and software specialists;
- Begin delivery and installation of system and exercised storage and network options;
- CORAL System Installation and Integration complete including IO Subsystem;
- CORAL System Accepted including IO Subsystem.

Key Elements of the Plan of Record

Within 60 days of subcontract award, the Offeror will provide a detailed Plan of Record (POR), which will include the following, in the minimum. This POR will be updated annually in the minimum but also anytime that circumstances dictate throughout the life of the project. Changes to any of the following key elements of the POR will be reported to the Laboratory's project managers within a week and documented in writing within a month

- **Project management plan** with management teams and organizational breakdown structure (OBS) identified.
- **Points of contact:** table of key personnel for contributing organizations within the company and its major subcontractors, and a description of their responsibilities and how these areas will be coordinated by the management team.
- **Work Breakdown Structure** (product oriented) including each major subsystems (e.g., CNs, I/O subsystem, SMNs, FENs), each software product (e.g., CNOS, RAS System, and Control System), and each major equipment delivery to the Laboratories.
- **Full term project schedule** and Gantt chart for the duration of the contract will be kept under configuration control with an audit trail of changes. The schedule will be developed using the Critical Path Method (CPM) scheduling technique and will utilize the same numbering scheme as the WBS. The Laboratories must concur with changes to capabilities, delivery/installation dates, and acceptance processes/schedules.
- **Project Plan Detail.** Using the same structure and sequence as this document, the POR will describe the planned tasks and their milestones in sufficient detail that CORAL and the subcontractor can assess and track progress. The plan should cover the duration of this contract and reflect a level of detail that covers the major subsections of this document. The Project Plan Detail will be kept under configuration control with an audit trail of changes. The Laboratories must concur with changes to capabilities, configurations, delivery/installation dates, and testing processes/schedules.

Key Elements of Risk Management Plan

Within 60 days of subcontract award, the Offeror will provide a detailed analysis of project risks and proposed risk management strategies. Overall the risk management plan will include the following, in the minimum. This plan will be updated quarterly throughout the life of the project. Changes that affect the management approach, status, mitigation strategy, or fallback strategy for any key risk will be reported to the Laboratory's project managers at least quarterly.

- Risk management process that includes the management approach, process, and schedule for updating risks;
- List and analysis of risks to contract schedule, scope/technical, and cost (where applicable);
- Responsible party/entity for each risk
- Risk mitigation and fallback strategies for key risks, with decision dates;
- Risk assessment related to secondary subcontractors, including a clear statement that the prime subcontractor accepts full financial responsibility for the relationship.

Key Elements of Collaboration Plan

Within 60 days of subcontract award, the Offeror will provide a detailed Collaboration Plan, which will include the following, in the minimum:

- WG topics and leadership teams (must include both Offeror and CORAL personnel);
- Arrangements for CORAL access to pre-production system(s) for porting, testing and integration at Offeror site;
- Opportunities for joint development of system capabilities, e.g., programming model, I/O, messaging, systems software;
- Open Source software components, development processes, availability, management plans, and responsibilities (CORAL and/or Offeror);
- Joint science, performance analysis, application analysis, and benchmarking activities;
- Provision of selected Offeror hardware, I/O and software staff to the Laboratory site(s);
- CORAL testing opportunities during installation.

Key Elements of Quality Assurance and Factory Test Plan

Within 60 days of subcontract award, the Offeror will provide a detailed Quality Assurance and Factory Test Plan, which will include the following, in the minimum. The Quality Assurance and Factory Test Plan will be updated, as necessary, to reflect the Offeror's latest processes and plan throughout the life of the project. Updates to this plan will be provided to the Laboratory's project managers at least quarterly.

- Process for qualifying vendors;
- Factory burn in and validation test plan;
- ASIC and system level margin testing;
- Pre-ship test plan for CORAL equipment.

Key Elements of Full-Term Hardware Development Plan

Within 60 days of subcontract award, the Offeror will provide a detailed Full-Term Hardware Development Plan, which will include the following, in the minimum. This plan will be updated as necessary throughout the life of the project. This plan will be kept under configuration control with an audit trail of changes. The Laboratories must concur with changes to capabilities and delivery/installation dates. A formal update to this plan will be provided to the Laboratory's project managers at every quarterly face-to-face meeting.

Processor Technology. Identify the planned milestones for processor development that lead to those to be deployed in the CORAL system. In particular, provide milestones for silicon process development, sampling, engineering quantities, and production quantities for each processor generation leading to the CORAL system.

Node Development. Provide the planned tasks and milestones for product development for all node types covered by this contract. Include tasks and milestones for: memory architecture; cache coherency protocols; ASIC development; performance modeling efforts; applications analysis; functional verification test; and system test. Indicate how and when this technology will be inserted to meet subcontract milestones.

CORAL High Performance Interconnect Development. Provide the planned tasks and milestones for interconnect research and development leading to the CORAL system. Include tasks and milestones for: switch ASIC development; interface components; cabling components; NIC and/or router design; overall bit error rate reduction; microcode, driver and MPI software development including support for multiple network adapters per node; functional verification test; and system test. Indicate how and when this technology will be inserted to meet subcontract milestones.

I/O Subsystem Development. Provide the planned tasks and milestones for I/O subsystem development leading to the CORAL system including functional verification and system test. The I/O subsystem test plan must delineate component and end-to-end testing. End-to-end testing is defined as starting (or ending) at a parallel application running on the CORAL system through the parallel I/O libraries down through the transport layers, through the device drivers and to the storage devices. If the proposed I/O subsystem solution includes multiple tiers then the plan must provide testing for each tier and tasks and milestones should cover each tier. Include tasks and milestones for: I/O subsystem network connectivity and associated hardware, storage devices and media, storage enclosures, storage controllers and associated hardware including racks, and electrical distribution. Indicate how and when this technology will be inserted to meet subcontract milestones.

System Scalability and Performance Testing. Provide the planned tasks and milestones for the scalability testing of system components. Include development of hardware for reliability, availability and serviceability (RAS).

Key Elements of Full-Term Software Development Plan

Within 60 days of subcontract award, the Offeror will provide a detailed Full-Term Software Development Plan, which will include the following. In each of these areas, the specific Open Source community model (if applicable) and development, testing and support plans should be discussed. This plan will be updated as necessary throughout the life of the project. This plan will be kept under configuration control with an audit trail of changes. The Laboratories must concur with changes to capabilities and delivery/installation dates. A formal update to this plan will be provided to the Laboratory's project managers at every quarterly face-to-face meeting.

CN Operating System (CNOS) Development. Provide the planned tasks and milestones for CN Operating System (CNOS) development. Include tasks and milestones for: execution model; support for dynamically linked libraries and Python based applications; shared memory region; thread mechanisms; memory management and utilization.

BOS Development. Provide the planned tasks and milestones for BOS development.

Integrated System Management Development. Provide the planned tasks and milestones for development of infrastructure and tools to manage the CORAL system as a single system via integrated system management. Include tasks and milestones for: system administration tools for installing and managing the cluster as a single system; user management, system scalable authentication mechanisms; and security services.

Reliability Availability and Serviceability. Provide the planned tasks and milestones for the development of scalable end-to-end RAS infrastructure and tools across all node types. Include tasks and milestones for: RAS reporting; RAS tools and infrastructure; system component discovery and monitoring; scalable FRU failure diagnostics and predictive failure approaches; error detection vs. retry; scalable system and interconnect diagnostics.

Resource Management Support. Provide the planned tasks and milestones for resource management development. Include tasks and milestones for: required interfaces; system monitoring tools; system scheduling; and scalable and reliable job launch, termination and control.

CORAL I/O Subsystem Development. Provide the planned tasks and milestones for CORAL IO Subsystem development described in Section 6. Include tasks and milestones for: file system and metadata performance; data integrity; resiliency; reliability; object insertion/deletion/retrieval performance; and manageability.

Input/Output Subsystem Support. Provide the planned tasks and milestones for supporting high-performance IO for parallel applications.

Compiler and Runtime Development. Provide the planned tasks and milestones for baseline language development. Include tasks and milestones for: mixed language support; compatibility with GNU compiler runtime; exploitation of novel hardware features for automatic and directed parallelization of applications; latency reduction techniques; compiler optimization for specialized hardware (e.g., vectorization or SIMD); language standards tracking.

Message Passing Environment. Provide the planned tasks and milestones for message passing development. Include tasks and milestones for: bandwidth and latency targets for MPI; MPI standard tracking; integration with debuggers, profilers and performance analysis tools; interoperability to cluster external resources.

Code Development Tools. Provide the planned tasks and milestones for code development tools development. Include tasks and milestones for: scalable code development tools infrastructure; remote process control tools interface; parallel make, profilers, debuggers, application performance monitoring tools, GUI development for code development tools.

Key Elements of Site Preparation Plan

The Offeror will provide a detailed Site Preparation Plan at least 1 year prior (two years preferred) to the first equipment delivery that will include at minimum the following items. At least 9 months prior to any part of the first system delivery the Final Site Preparation plan will be delivered to the Laboratory's project managers. The site preparation plan will be updated as necessary until the milestone for the Final

Site Preparation Plan (i.e., 9 months prior to delivery of part of the first system delivery). Changes to the Final Site Preparation Plan will be strongly discouraged and require the Laboratory's concurrence. The site preparation plan will be kept under configuration control with an audit trail of changes. Any changes to this plan will be provided to the Laboratory's project managers at least monthly and at the quarterly face-to-face meeting.

- Cabinet dimensions, packaging diagrams, and weights (in all configurations – in packaging, dry, with any liquid coolant);
- Electrical requirements for everything provided by selected Offeror;
- System layout and cabling requirements, including expansion options;
- Raised floor requirements and cutouts;
- Cable tray requirements;
- Environmental requirements;
- Expected power and cooling requirements;
- Cooling water quality requirements;
- Safety requirements.

Key Elements of Installation Process Plan

At least 1 year before the first equipment delivery, the Offeror will provide a detailed Installation Process Plan. This plan will be updated as necessary until the milestone for the delivery of the Final Installation Process Plan is reached. The Final Installation Process Plan will be provided at least 3 months prior to the first equipment delivery. The Installation Process Plan will be kept under configuration control with an audit trail of changes. Changes to the Installation Process Plan will be provided to the Laboratory's project managers at least monthly and at the quarterly face to face meeting. The Installation Process plan will include the following, in the minimum:

- Core installation team and staffing plan;
- Subcontractor-CORAL communications plan;
- Equipment delivery and testing schedule;
- Staging and temporary storage area needs;
- Pre-delivery access and work needs;
- Factory staging milestones;
- Shipping plans;
- Equipment movement process, from truck to computer room floor, to final locations;
- Equipment layout and installation sequence (for multi-stage deliveries);
- Bring-up plan;
- Testing and QA plan;
- Safety plan that includes a hazards analysis of the Offerer's installation activities.

Key Elements of System Administration Guide

At least 1 year before the first equipment delivery, the Offeror will provide a detailed System Administration Guide. This guide will be updated as necessary throughout the life of the project. This guide will be kept under configuration control with an audit trail of changes. Any changes to this guide

will be provided to the Laboratory's project managers at least monthly and at the quarterly face to face. The System Administration Guide will include the following, in the minimum:

- Cycling power;
- Configuring the system and running jobs;
- Management control system;
- Hardware monitor;
- Configuring the IO Subsystem;
- System operations;
- Running diagnostics;
- Problem determination;
- Safety considerations.

Key Elements of Maintenance and Support Plan

At least 1 year before the first equipment delivery, the Offeror will provide a detailed Maintenance and Support Plan. This plan will be kept under configuration control with an audit trail of changes. Any changes to this plan will be provided to the Laboratory's project managers at the quarterly face to face. The Maintenance and Support Plan will include the following:

- Obtaining hardware and software support from Offeror;
- Reporting and tracking system problems;
- Trouble report escalation process;
- CORAL and Offeror responsibilities in shared maintenance plan;
- Preventative maintenance requirements;
- Safety considerations;
- Cycling power;
- Parts replacement;
- Post-installation parts availability timeline.

13.1 Build System Prototype Review (TR-1)

Selected Offeror will deliver a final report on the system prototype results for the Laboratories' review and approval. As part of this review, the Laboratories will review the progress of the design and development of the system in meeting the requirements of the build SOW. The exact results to be reviewed will be specified in the individual Laboratory build SOW. The review will also finalize any strategies and requirements. This milestone will be complete when the project is reviewed at a face-to-face meeting and the updated plan is approved by the Laboratory Technical Representative in writing.

13.2 Acceptance Requirements (TR-1)

Upon delivery and installation, a series of performance, functionality, and availability tests will be performed prior to acceptance. Acceptance testing will comprise multiple components where the overall goal is to ensure that the system as a whole is high-performance, scalable, resilient and reliable. Acceptance testing will exercise the system infrastructure with a combination of benchmarks, forced failures, and stability tests. Any requirement described in the Technical Specification may generate a corresponding acceptance test. CORAL may identify other system aspects that merit testing; Offeror shall not attempt to limit the capabilities that CORAL may test. The specifics of the acceptance test plan will be determined during contract award negotiation. These acceptance requirements apply to the main CORAL

system and to the IO Subsystem as well as any additional systems procured as the result of exercising any options described in Section 3.7.

14.0 Appendix A Glossary

14.1 Hardware

CN	System compute nodes. Compute Nodes (CN) are nodes in the system on which user jobs execute.
Core	Portion of processor that contains execution units (e.g., instruction dispatch, integer, branch, load/store, and floating-point), registers and typically at least L1 data and instruction caches. Typical cores implement multiple hardware threads of execution and interface with other cores in a processor through the memory hierarchy and possibly other specialized synchronization and interrupt hardware.
DDR DIMM	Double data rate dual in-line memory module
FLOP	Floating Point Operation.
FLOPS	Floating Point Operation per second.
FMA	Fused Multiply Add (FMA) is a single 64b or 32b floating-point instruction that operates on three inputs by multiplying one pair of the inputs together and adding the third input to the multiply result
FPE	Floating Point Exception.
GB	gigaByte. gigaByte is a billion base 10 bytes. This is typically used in every context except for Random Access Memory size and is 10^9 (or 1,000,000,000) bytes.
GiB	gibiByte. gibiByte is a 1,073,741,824 base 10 bytes (i.e., 1024^3 bytes).
GFLOPS or GOP/s	gigaFLOPS. Billion (10^9 = 1,000,000,000) 64-bit floating point operations per second.
HBM	High Bandwidth Memory
IBA	InfiniBand™ Architecture (IBA) http://www.infinibandta.org/specs
ISA	Instruction Set Architecture.
FEN	Front End Nodes. Front End Nodes are nodes where users and administrators can login in and interact with the system.
MB	megaByte. megaByte is a million base 10 bytes. This is typically used in every context except for Random Access Memory size and is 10^6 (or 1,000,000) bytes.
MIB	Management Information Base is a database used for managing the entities in a communication network, typically used with SMNP.
MiB	mebiByte. mebiByte is a 1,048,576 base 10 bytes (i.e., 1024^2 bytes).
MFLOPS or MOP/s	megaFLOPS. Million (10^6 = 1,000,000) 64-bit floating point operations per second.

MTBAF	Mean Time Between (Hardware) Application Failure. A measurement of the expected hardware reliability of the system or component as seen from an application perspective. The MTBAF figure can be developed as the result of intensive testing, based on actual product experience, or predicted by analyzing known factors. Hardware failures of or transient errors in redundant components such as correctable single bit memory errors or the failure of an N+1 redundant power supply and do not cause an application to abnormally terminate do not count against this statistic. Thus, MTBAF ≥ MTBF.
MTBF	Mean Time Between (Hardware) Failure. A measurement of the expected hardware reliability of the system or component. The MTBF figure can be developed as the result of intensive testing, based on actual product experience, or predicted by analyzing known factors. See URL: http://www.t-cubed.com/faq_mtbf.htm
NCORE	The number of cores in the CN allocatable to and directly programmable by user MPI processes. If the peak petaFLOPS system characteristic requires multiple threads per core to be issuing floating-point instructions, then NCORE is the number of allocatable cores times that number of threads.
Node	A set of CPUs and associated accelerators (if present) sharing random access memory within the same coherent memory address space. From the SRM perspective, is the indivisible resource that can be allocated to a job.
Non-Volatile	Non-volatile memory, nonvolatile memory, NVM or non-volatile storage, is computer memory that can retain the stored information even when not powered.
NUMA	Non-Uniform Memory Access architecture.
NVRAM	Non-Volatile Random Access Memory
PB	petaByte. petaByte is a quadrillion base 10 bytes. This is typically used in every context except for Random Access Memory size and is 10^{15} (or 1,000,000,000,000) bytes.
PiB	pebiByte. pebiByte is a 1,125,899,906,842,624 base 10 bytes (i.e., 1024^5 bytes).
Peak FLOPS Rate	The maximum number of 64-bit floating point instructions (add, subtract, multiply or divide) or operations (instructions) per second that could conceivably be retired by the system.
Peta-Scale	The environment required to fully support production-level, realized petaFLOPS performance.
Processor	The computer ASIC die and package.
Scalable	A system attribute that increases in performance or size as some function of the peak rating of the system.
SECDED	Single Error Correction Double Error Detection. Storage and data transfer protection mechanism that can detect parity errors (single bit errors) and detect storage or data transfer errors with multiple bits in them.

SIMD	Single Instruction, Multiple Data (SIMD) instructions are processor instructions that operate on more than one set of input 64b or 32b floating-point values and produce more than one 64b or 32b floating-point value. Fused Multiply-Add (FMA) instructions are not SIMD. Examples of this are x86-64 SSE2 and Power VMX instructions.
SNMP	Simple Network Management Protocol is a popular protocol for network management. It is used for collecting information from, and configuring, network devices, such as servers, printers, hubs, switches, and routers on an Internet Protocol (IP) network.
Thread	Hardware threads are typically exposed through the operating system as independently schedulable sequences of instructions. A hardware thread executes a software thread within a Linux (or other) OS process.
TB	TeraByte. TeraByte is a trillion base 10 bytes. This is typically used in every context except for Random Access Memory size and is 10^{12} (or 1,000,000,000,000) bytes.
TiB	tibiByte. tibiByte is a 1,099,511,627,776 base 10 bytes (i.e., 1024^4 bytes).
TLB	Translation Look-aside Buffer (TLB) is a set of content addressable hardware registers on the processor that allows fast translation of virtual memory addresses into real memory addresses for virtual addresses that have an active TLB enetry.
TFLOPS	teraFLOPS. Trillion (10^{12} = 1,000,000,000,000) 64-bit floating point operations per second.
UMA	Uniform Memory Access architecture. The distance in core clocks between core registers and every element of node memory is the same. That is, load/store operations that are serviced by the node memory have the same latency to/from every core, no matter where the target physical location is in the node memory assuming no contention.

14.2 Software

32b executable	Executable binaries (user applications) with 32b (4B) virtual memory addressing.
64b executable	Executable binaries (user applications) with 64b (8B) virtual memory addressing.
API	Application Programming Interface: Syntax and semantics for invoking services from within an executing application.
Baseline Languages	The Baseline Languages are Fortran08, C, C++ and Python.
BIOS	Basic Input-Output System (BIOS) is low level (typically assembly language) code usually held in flash memory on the node that tests and functions the hardware upon power-up or reset or reboot and loads the operating system.

BOS	Base Operating System (BOS). Linux (LSB 3.1) compliant Operating System run on the FEN.
CDTI	The hierarchal Code Development Tools Infrastructure (CDTI) components are distributed throughout the CORAL system. Individual code development tool "front-end" components that interact with the user execute on the FEN (although the display may be remoted via an X-Window). Code development tool communications mechanisms interface the tool "front-ends" running on the FEN with the user application running on the CN through a single level fan-out hierarchy.
Current standard	Term applied when an API is not "frozen" on a particular version of a standard, but will be upgraded automatically by Offeror as new specifications are released
Fully supported	A software product-quality implementation, documented and maintained by the HPC machine supplier or an affiliated software supplier.
Job	An allocation of resources to a user for a specified period of time. The user should be given control over which resources can be allocated to a job.
CNOS	Light-Weight Kernel providing operating system functions to user applications running on CN.
OS	Operating System
Published (as applied to APIs):	Where an API is not required to be consistent across platforms, the capability lists it as "published," referring to the fact that it will be documented and supported, although it will be Offeror- or even platform-specific.
RPCTI	Remote process control code development tools interface that allows code development tools to interface from the FEN to the CNOS on the CN and operate on user processes and threads on the CN.
Single-point control	Refers to the ability to control or acquire information on all processes/PEs using a single command or operation.
Standard (as applied to APIs)	Where an API is required to be consistent across platforms, the reference standard is named as part of the capability.
Task	A process launched as a job step component, typically an MPI process.

Appendix A: 2021 I/O Subsystem Use Cases

A. **Writing Output Data/Restart Performance** (6.1.3.2.2, 6.1.3.2.3, 6.1.3.2.4, 6.1.3.2.7, 6.1.3.2.8, 6.1.3.4)
Writing Output Data is the highest priority use case for the I/O subsystem. During a job execution, the application's processes will periodically output datasets to the I/O subsystem. Most ORNL applications write periodic data that will be analyzed later as well as used for restart. Very few write defensive-only checkpoints. The typical application will create one file per MPI process, write the data, and close the file. Some applications may open one or more shared files instead of one file per process. The I/O subsystem will ensure that the written data is made durable within a reasonable time without user intervention or when requested explicitly by the user. The application process will be able to read recently written data back at the same rate as it was written. The writes to the global namespace should be durable within 15 minutes.

B. **Reading Input Data at Job Launch** (6.1.3.2.5, 6.1.3.2.6)
To minimize job start times, application processes should be able to read input datasets quickly when using the entire CN partition. The duration of the read I/O phase (file open, file read, file close) should be limited to 10 minutes. The I/O subsystem may require assistance from the scheduling system to stage in the required data, but without requiring user intervention other than providing a list of directories or files to the scheduling system. When preparing for new jobs, cache-warming mechanisms should avoid stalling a job if one of the reserved nodes fails prior to job launch.

C. **Workflows and In-situ Data Analysis** (6.1.3.3)
After the simulation data is generated on the CN partition and written and persisted to the I/O subsystem, the user may want to read back the same dataset to analyze. The user may run the analysis job on the same nodes, on a different set of nodes in the CN partition, or on another cluster within the Laboratory datacenter that has access to the I/O subsystem. When the analysis application reads the data, it should read the correct and up-to-date data. The I/O subsystem should enforce consistency after a file has been written and closed. The data may reside anywhere within the I/O subsystem and any process, inside or outside the CN partition, should be able to access and read the correct data. The I/O subsystem may move the data, as needed, when a process modifies a file.

D. **User-Interactive Directory Walks** (6.1.3.1.4)
From the FENs, users will interactively search for files using `ls -l` and `find`. It is expected that at least 10 users will be executing these commands to search directories concurrently.

E. **Multi-user Concurrent Compilations** (6.1.3.1.3)
From the FENs, multiple users will simultaneously compile applications and libraries (see Section 9.1.5). The I/O subsystem will support high rate metadata and data operations (both reads and writes) on small files ~32KiB from a single FEN.

F. **System-Automated Purge** (6.1.3.1.5, 6.1.3.1.6)
Even with a large usable capacity, the user-generated data may quickly fill the I/O subsystem. Labs employ periodic purging of user data to efficiently manage the I/O subsystem capacity. Bottlenecks in purging are centered around efficiently walking the namespace and unlinking files.

G. **System-Automated Metadata and Log Collection** (6.1.7.1)
The Laboratory's operations teams will periodically collect metadata information and logs from the I/O subsystem at both device and file system level. This data includes, but not limited to, file and directory counts, sizes, directory tree depth, per job read and write bandwidths, read and write IOPs,

I/O request sizes and I/O request latencies, and also, I/O subsystem device characteristics, read and write failures and number of read and write retries. It will be necessary to collect this data in a performant manner using well-documented APIs.

H. Advanced Metadata Capabilities (6.1.4.3, 6.1.4.4)
With ever growing I/O subsystem capacities, users have a hard time searching and finding data. An I/O subsystem that provides efficient, alternate methods of searching is expected to help users. Richer metadata annotation of files, some generated automatically and some appended by the user or through library APIs so that the user can efficiently search metadata for files of interest without negatively impacting global performance of the I/O subsystem metadata service is expected to assist in this process.

I. Memory Extension (5.2.12)
Some applications may require more memory than what is provided on the CN and it is expected that users may want to use the storage target of the nearest tier as an extended secondary memory partition.

J. Read Caching for Machine Learning/Deep Learning
Machine learning and deep learning applications are becoming increasingly relevant on large HPC systems. These applications need to read the same files multiple times. The size of the datasets is growing to PBs and these applications randomly read through the files thousands of times. Caching of common libraries and other system files to reduce the load on the I/O subsystem is desirable.