A Management Framework for Consolidated Big Data and HP

Project number: 45924

Deliverable 1 .1 - Platform Requirements

Date	30 September 2020
Activity 1	Platform Requirements, Design and APIs
Version	Final

Authors: João Paulo (INESC TEC); Ricardo Vilaça (MACC); Rui Ribeiro (MACC); Richard Todd Evans (UT/TACC); Amit Ruhela (UT/TACC); Stephen Harrel (UT/TACC); Bruno Antunes (Wavecom)

Reviewers: João Paulo (INESC TEC); Richard Todd Evans (UT/TACC); Mario David (LIP)

Partners

Funding

Table of Content

Executive Summary

High-Performance Computing (HPC) infrastructures are increasingly sought to support Big Data applications, whose workloads significantly differ from those of traditional parallel computing tasks. This is expected given the large pool of available computational resources, which can be leveraged to conduct a richer set of studies and analysis for areas such as healthcare, smart cities, natural sciences, among others. However, coping with the heterogeneous hardware of these large-scale infrastructures and the different HPC and Big Data application requirements raises new research and technological challenges. Namely, it becomes increasingly difficult to efficiently manage available computational and storage resources, to provide transparent application access to such resources, and to ensure performance isolation and fairness across the different workloads.

The BigHPC project aims at addressing these challenges with a novel management framework, for Big Data and parallel computing workloads, that can be seamlessly integrated with existing HPC infrastructures and software stacks. Namely, the project will develop novel monitoring, virtualization, and storage management components that can cope with the infrastructural scale and heterogeneity, as well as, the different workload requirements, while ensuring the best performance and resource usage for both applications and infrastructures.

These components will be integrated into a single software bundle that will be validated through real use-cases and a pilot deployed on both TACC and MACC data centers. Also, the proposed framework will be provided as a service for companies and institutions that wish to leverage their infrastructures for deploying Big Data and HPC applications.

This deliverable details the user and platform requirements for the BigHPC framework while taking into account the project's use-cases and the experience of TACC and MACC on providing HPC services to different stakeholders. This information is crucial to guide the design and development of the different platform components and to ensure their proper integration.

Glossary

API	Application Programming Interface
CLI	Command Line Interface
CLI	Command Line Interface
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
GPU	Graphics Processing Unit
HPC	High-Performance Computing
I/O	Input/Output
INESC TEC	Institute for Systems and Computer Engineering, Technology and Science
LIP	Laboratory of Instrumentation and Experimental Particle Physics
MACC	Minho Advanced Computing Center
MPI	Message Passing Interface
RAM	Random-Access Memory
RDMA	Remote Direct Memory Access
SDS	Software-Defined Storage
SVE	Scalable Vector Extension
TACC	Texas Advanced Computing Center

1. Introduction

HPC infrastructures and services are no longer solely targeted at highly parallel modeling and simulation tasks. Indeed, the computational power offered by these systems is now being used to support advanced Big Data analytics for fields such as healthcare, agriculture, environmental sciences, smart cities, fraud detection, among others [OG+15, NCR+18]. By combining both types of computational paradigms, HPC infrastructures will be key for improving the lives of citizens, speeding up scientific breakthroughs in different fields (e.g., health, IoT, biology, chemistry, physics), and increasing the competitiveness of companies.

As the utility and usage of HPC infrastructures increases, more computational and storage power is required to efficiently handle the amount of targeted data-driven applications. In fact, many HPC centers are now aiming at exascale supercomputers supporting at least one exaFLOPs (10^{18} operations per second), which represents a thousandfold increase in processing power over the first petascale computer deployed in 2008 [RD+15, ECS+17]. Although this is a necessary requirement for handling the increasing complexity and scale of HPC applications, there are several outstanding challenges that still need to be tackled so that this extra computational power can be fully leveraged.

Management of heterogeneous infrastructures and workloads: By adding more compute and storage nodes one is also increasing the complexity of the overall HPC distributed infrastructure and making it harder to monitor and manage. This complexity is increased due to the need of supporting highly heterogeneous applications that translate into different workloads with specific data storage and processing needs [ECS+17].

Support for general-purpose analytics: The increased heterogeneity also demands that HPC infrastructures are now able to support general-purpose applications that were not designed explicitly to run on specialized HPC hardware, which was typically the case for traditional modeling and simulation applications [KWG+13].

Avoiding the storage bottleneck: As a complementary challenge, by only increasing the computational power and improving the management of HPC infrastructures it may still not be possible to fully harness the capabilities of these infrastructures. In fact, many applications are now data-driven and will require efficient data storage and retrieval (e.g., low latency or/and high throughput) from HPC clusters. With an increasing number of applications and heterogeneous workloads, the storage systems supporting HPC may easily become a bottleneck [YDI+16, ECS+17]. As pointed out by several studies, the storage access time is one of the major bottlenecks limiting the efficiency of current and next-generation HPC infrastructures.

To sum up, the BigHPC project aims at addressing three main challenges: 1) improving the management of heterogeneous HPC infrastructures and workloads; 2) enabling the support for general-purpose analytical applications; and 3) solving the current storage access bottleneck of HPC services. Addressing these challenges is crucial for taking full advantage of the next generation of HPC infrastructures.

The goal of this deliverable is to further explore these three challenges in terms of infrastructural and user requirements. This analysis will be based on the current infrastructures supported at MACC and TACC while focusing on the efficient support of both parallel and Big Data workloads.

The functional and non-functional requirements described in this document, along with an analysis of current solutions available for managing HPC infrastructures, will be key to drive the design, architecture and implementation of the Big HPC platform.

The document is structured as follows: Section 2 provides a preliminary overview of the envisioned BigHPC framework. Section 3 details the framework's general requirements, as well as, MACC and TACC's infrastructures, software and supported workloads. Then, by building on the previous information, Section 4 further discusses the functional and nonfunctional requirements for the components of BigHPC's framework. Section 5 concludes this deliverable.

2. The BigHPC Platform

BigHPC will design and implement a novel solution for monitoring and managing the infrastructure, data and applications of current and next-generation HPC data centers. The proposed solution aims at enabling both traditional HPC, as well as, novel Big Data analytics applications to be deployed on top of heterogeneous HPC hardware. Also, it will ensure that resources (e.g., CPU, RAM, storage, network) are monitored and managed efficiently, thus leveraging the full capabilities of the infrastructure while ensuring that the performance and availability requirements of the different applications are met.

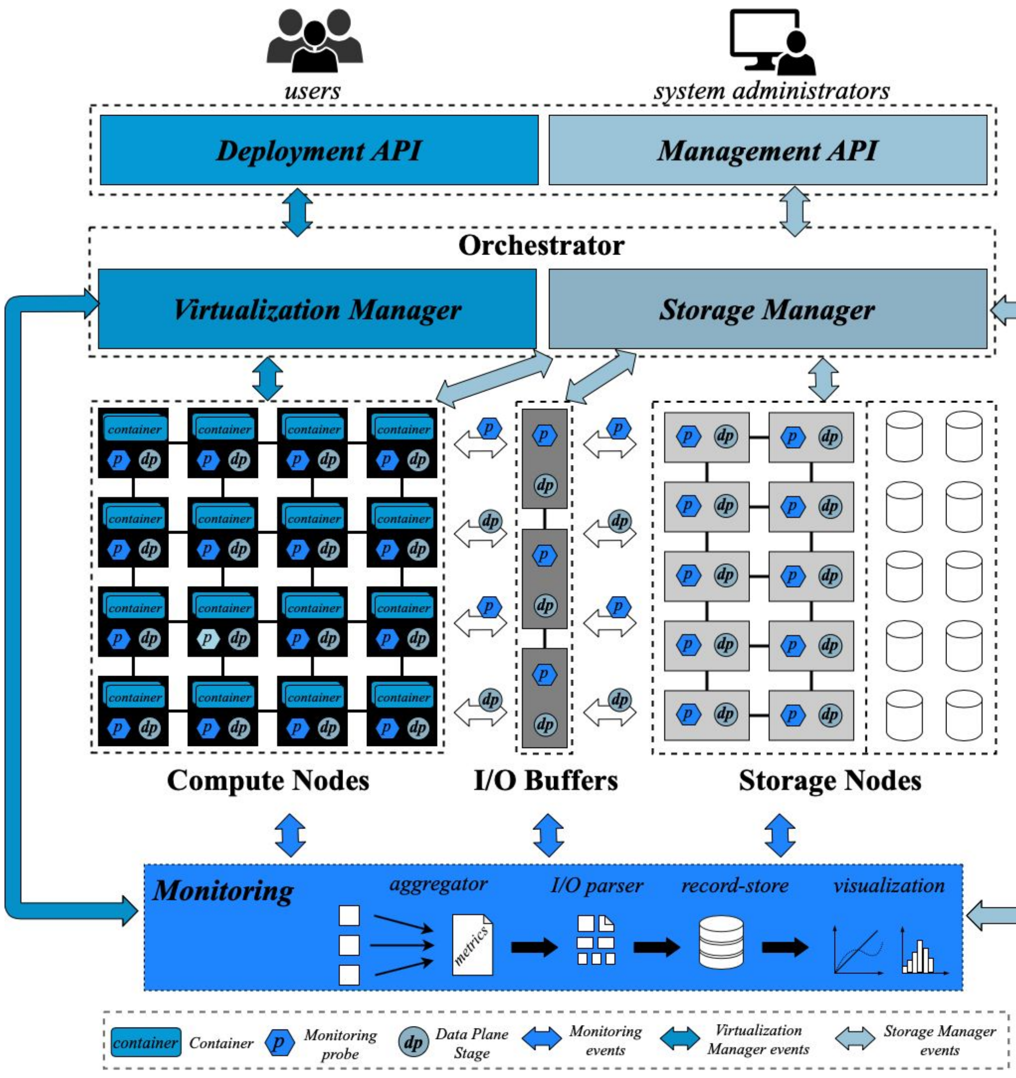

Figure 1: Overview of BigHPC platform

As depicted in Figure 1, The framework will support a deployment API that will be implemented through a Command Line Interface (CLI). Implementation of such high-level interfaces will ease the adoption and use of the platform by users to deploy their HPC and Big Data analytics applications. Moreover, a management API, likewise implemented through a Command Line Interface (CLI), will enable system administrators to manage the overall infrastructure and deployed applications.

These external APIs are provided by a modular Orchestrator component in which different management modules can be seamlessly integrated by respecting a common internal API. In BigHPC, the Orchestrator will support two main modules, namely a Virtualization Manager and a Software-Defined Storage (SDS) Manager. However, by keeping the design of this component modular, the project aims at ensuring a straightforward integration of other management modules in the future (e.g., Software-Defined Networking Manager [KRV+15]). This is a core component of the framework that will be responsible for managing the deployed applications and infrastructure resources in a holistic fashion. Such will require collecting application and cluster resource metrics from the Monitoring component, further described next.

The Virtualization Manager is responsible for abstracting the heterogeneous physical resources and low-level software packages (e.g., compilers, libraries, etc) that currently exist in an HPC cluster and to provide a common abstraction so that both HPC and Big Data applications can be easily deployed on such infrastructures. To achieve such a goal, this manager will resort to virtual containers. Containers will be running user applications in an isolated fashion and their placement, across the computational nodes, will take into account the performance and reliability requirements of each application while providing optimized usage of the infrastructures' overall resources (i.e., CPU, RAM, network, I/O, energy).

The Storage Manager will ensure optimized configuration of shared storage resources provided to applications and jobs running at the HPC infrastructure. This component will follow a Software-Defined Storage [MPP+20] approach while providing the building blocks for ensuring end-to-end control of storage resources, in order to achieve QoS-provisioning, performance isolation, and fairness between HPC applications.

The Storage manager will decouple the control and data flows into two major components, control and data planes. The data plane is a programmable multi-stage component distributed along the I/O path that enforces fine-grained management and routing properties dynamically adaptable to the infrastructure status (e.g., rate limiting, I/O prioritization and fairness, bandwidth aggregation and flow customization). Such a component can be employed over compute, I/O, and storage servers. The control plane comprehends a logically centralized controller with system-wide visibility that orchestrates the overall storage infrastructure (i.e., data plane stages and storage resources) in a holistic fashion. The control plane is scalable and enforces end-to-end storage policies, tailored for attending the requirements of exascale computing infrastructures.

Finally, BigHPC will also provide a Monitoring component that will collect resource usage metrics (e.g., CPU, RAM, network and storage I/O) for Big Data and parallel computing applications deployed at the HPC infrastructure. The solution will collect both short-term and long-term resource metrics to enable better management of deployed containers and applications, and to understand their I/O patterns, which will be crucial for the Orchestrator component and respective modules.

These components (i.e., the Orchestrator, Virtualization Manager, SDS Manager, and Monitoring) will be integrated into a single software bundle that will be validated by resorting to the infrastructures of both TACC and MACC center, and productized and explored commercially by Wavecom.

3. General Requirements

This section describes the requirements for scientific and Big Data workloads, while contextualizing these with the infrastructures and software currently supported by MACC and TACC supercomputers.

3.1. Scientific Applications

Types of Applications. Traditional scientific modeling and simulation tasks require large slices of computational time, are CPU- or memory bandwidth- bound, and rely on iterative approaches (parametric/stochastic modeling). Modern applications require tens to thousands of nodes and significant, often tightly-coupled communication among distributed processors and accelerators. MPI, CUDA, and shared-memory programming models facilitate the applications' intra-node and inter-node communication in an optimal way, utilizing the hardware's capabilities. These applications often have infrequent, streaming IO enabling checkpoint and restart mechanisms for partial program executions. Traditional applications originate from a broad range of scientific disciplines such as materials science, fusion, geosciences, astrophysics, particle physics, and fluid mechanics. Examples of commonly used applications with established user communities include NAMD, GROMACS, MILC, CHROMA, OpenFOAM, and VASP.

Functional Requirements. BigHPC should provide the mechanisms to compile and execute traditional HPC applications such that they can utilize available hardware capabilities and are run in a performant manner. Hardware capabilities include processor-specific ISAs and optimizations, RDMA supporting networks, and accelerator technologies such as GPUs. Thus BigHPC should support multiple compilers and compilation options, MPI implementations and shared-memory APIs (OpenMP), and libraries for accessing accelerators.

BigHPC should be able to run these applications within an HPC system's existing workload manager such as SLURM, without modifying the existing HPC infrastructure. It should mitigate the interference between simultaneous executions of applications by combining information about current resource usage and the expected resource usage of a scheduled application execution, and running that application accordingly.

Non-functional Requirements. The execution of applications on distributed computing systems is prone to errors in compilation and runtime configurations that result in less than optimal performance. BigHPC should provide sufficient information to the user to indicate whether their application execution is using available hardware or is misconfigured in a manner that all requested resources are not used. It should also not prevent the incorporation of code profilers and debuggers into user workloads.

3.2. Big Data Applications

Types of Applications. Big data applications are often IO bound and have diverse execution characteristics. They often are only supported in a restricted software ecosystem and do not provide for flexible customization of compilation or runtime configurations. The main classes of applications as defined for this project are characterized below.

Traditional Big Data: Applications in this category expect full control over and often exclusive access to compute resources. Persistent services with interactivity is also often desirable. Applications running on Hadoop, PySpark, or expecting database accessibility qualify. Applications expecting a Jupyter notebook or other interactive GUI also qualify. MACC and TACC cannot readily support these applications on HPC resources. Some Machine Learning workflows fall into this category.

Machine Learning: Tensorflow and PyTorch are currently popular. Many of these workflows can run on TACC HPC resources but often are challenging or impossible to compile, difficult to maintain due to rapid development combined with inconsistent software engineering practices (e.g. limited version-to-version API/ABI compatibility), and overly demanding on filesystem IO resources. The workflows can usually be configured to reduce load on the filesystems with expert support. These applications generally can be greatly accelerated when configured to use GPUs. There is an ongoing effort for distributed deep learning training frameworks, such as Horovod, to accelerate training on multiple nodes. Currently, most applications only support single-node execution, although many can leverage multiple GPUs if they are collocated on a single node. As another optimization, data processing and machine learning are often splitted in two different pipelines. Enabling machine learning frameworks to integrate seamlessly with ETL jobs allows for more streamline production jobs, with faster iteration between feature engineering and model training. This possible approach breaks down the barriers between ETL and continuous model training.

Life Sciences: These tend to be highly varied and bespoke applications with little community by-in. The TACC Lifesciences Group maintains container images for thousands for these applications. They are typically difficult or impossible to build in a software environment other than where they were written. They are also often designed with pathological IO patterns with little initiative or capability from the user community to improve the behavior. These applications are supported by TACC but typically without taking advantage of native hardware capabilities. They are also prone to overloading the filesystem with IO.

HTC Workloads: TACC has many projects with high throughput computing (HTC) workloads, even on its capability systems. The workloads are characterized by customized work schedulers running within the primary job scheduler (e.g. SLURM), persistent services requirements, containerized applications, short executable runtimes, and relatively high IO requirements. The applications and fields of science that could be classified as HTC are diverse, ranging from analyses of particle physics data to functional brain mapping. TACC supports these applications with difficulty.

Functional Requirements. BigHPC should enable the execution of the above described classes of Big Data applications on traditional HPC infrastructure. It will provide the mechanisms, where possible, to compile these applications such that they utilize existing hardware capabilities. Their specific software requirements will be supported through container technologies. Execution of persistent services such as databases will be possible within the primary job scheduler of the host HPC system by standing up and tearing down such services at the initialization and finalization of a job.

Intensive IO demands will be supported through a combination of the BigHPC monitoring service and storage manager components. It should mitigate the interference between simultaneous executions of applications by combining information about current resource usage and the expected resource usage of a scheduled application execution, and running that application accordingly.

Non-functional Requirements. BigHPC will provide sufficient information to the user for them to determine whether their Big Data workloads are utilizing existing hardware and requested resources. Quality of service may be maintained for both remote and local IO needs.

3.3. Infrastructure and Software

TACC is currently operating 6 major compute cluster resources, comprising nearly 16,000 nodes in total. TACC also maintains extensive storage resources and cloud infrastructure resources. Of these resources, 4 have been identified as most appropriate for BigHPC development and testing. These resources have been chosen due to their availability, hardware heterogeneity, applicability to general HPC systems, and scale. These characteristics ensure the design of BigHPC will address the broadest possible range of use cases. The 4 resources chosen as test beds are shown in Table 1. Node count and type, interconnect, and filesystems for each resource are included in the table. Detailed descriptions of each node type are provided in Table 2, while Table 3 details the storage backends supporting such nodes. A description of each system follows.

Resource	**Stampede2**	**Frontera Primary Compute**	**Frontera Liquid Submerged System**	**Longhorn**
Nodes	4,204 Knights Landing 1,736 Intel Skylake	8,008 Cascade Lake	90 Broadwell + 4 Single Precision GPU	96 IBM Power9 + 4 Double Precision GPU
Interconnect	Intel Omnipath (100Gb/s)	Mellanox Infiniband HDR-100 (100GB/s)	Mellanox Infiniband FDR (50Gb/s)	Mellanox Infiniband EDR (100Gb/s)
Filesystem	HOME 1 PB Lustre SCRATCH 18 PB Lustre	HOME 0.5 PB Lustre SCRATCH 44 PB Lustre	HOME 0.5 PB Lustre SCRATCH 44 PB Lustre	HOME 11 TB IBM Spectrum Scale SCRATCH 5 PB IBM Spectrum Scale

Table 1: Overview of TACC compute clusters

Resource	***Node type***	***CPUs***	***RAM***	***GPU***	***Disk***
Stampede2	Skylake	Intel Xeon 8168 2 2.1 GHz CPUs 24 cores /CPU 2 HW threads/core	192 GB (2666 MT/s) DDR4	N/A	200GB SSD 144 GB /tmp
Stampede2	Knights Landing	Intel Xeon Phi 7250 1 1.4 GHz CPU 68 cores/CPU 4 HW threads/ core	96 GB (2666 MT/s) DDR4 16 GB MCDRAM	N/A	200 GB SSD 107 GB /tmp
Frontera Primary Compute	Cascade Lake	Intel Xeon 8280 2 2.7 GHz CPUs 28 cores/CPU 1 HW threads/core	192 GB (2933 MT/s) DDR4	N/A	240 GB SSD 144 GB /tmp
Frontera Liquid Submerged System	Broadwell + 4 Single Precision GPU	Intel Xeon E5-2620 V4 2 2.1 GHz CPUs 8 cores/CPU 1 HW thread/core	128 GB (2133 MT/s) DDR4	4 NVIDIA Quadro RTX 5000 GPUs	240 GB SSD 144 GB /tmp
Longhorn	IBM Power 9 + 4 Double Precision GPU	IBM Power 9 2 2.3 GHz CPUs 20 cores/CPU 4 HW threads/core	256 GB (2666 MT/s) DDR4	4 NVIDIA Tesla V100 GPUs	900 GB /tmp

Table 2: Detailed description of TACC Compute Nodes.

Resource	**Filesystem type**	**#MDS/#OSS**	**OST Capacity**	**Network**	**Filesystem IO BW**
Stampede2	Cray ClusterStor Lustre	HOME 2/4 SCRATCH 4/66	HOME 280TB SCRATCH 280TB	Intel Omnipath 100	SCRATCH 300 GB/s
Frontera (primary compute & liquid submerged system)	ES-18K-HDR Lustre	HOME 2/4 SCRATCH1 2/16 SCRATCH2 2/16 SCRATCH3 4/32	HOME 125 TB SCRATCH1 680TB SCRATCH2 680TB SCRATCH3 680TB	Mellanox Infiniband HDR-100	SCRATCH1 60 GB/s SCRATCH2 60 GB/s SCRATCH3 120 GB/s
Longhorn	IBM Spectrum Scale (GPFS)	HOME+SCRATCH 1/2	HOME 11 TB SCRATCH 4.5 PB	Mellanox Inifiniband EDR	N/A

Table 3: Detailed description of TACC Storage backends

Stampede2: Stampede2 is the flagship supercomputer for the NSF's Extreme Science and Engineering Discovery Environment (XSEDE) program. Stampede2 provides HPC capabilities to thousands of researchers across the U.S. and their international collaborators. It entered full production in Fall 2017. Its peak Linpack performance (Rmax) was measured at 10.7 PFlops/s and currently ranks as the #21 fastest supercomputer in the world.

Stampede2 is composed of roughly 70% Intel Xeon Phi "Knights Landing" nodes and 30% Intel Xeon "Skylake" nodes. The Knights Landing nodes have 16 GB of high speed MCDRAM, capable of providing 450+ GB/s bandwidth to the processor. The Knights Landing nodes efficiently support highly parallel workloads, while the Skylake nodes are capable of supporting more general workloads.

Frontera Primary Compute: Frontera is the leadership-class system in the NSF's cyberinfrastructure ecosystem. It entered production in August 2019 and debuted as the #5 fastest supercomputer in the world at 23.5 PFlops/s. It provides resources to the most compute intensive NSF projects.

The primary compute component of Frontera consists of 8008 Intel Xeon “Cascade Lake” nodes. These nodes are capable of efficiently supporting diverse workloads, ranging from the predominantly serial applications used in high throughput computing to tightly coupled, massively parallel capability computations.

Frontera Liquid Submerged System: Frontera is augmented with a collocated 90 node GPU subsystem. The commodity NVIDIA Quadro RTX 5000 GPUs are suspended in mineral oil in 4 Green Revolution Cooling ICEraQ racks. This configuration, unique at the time of it’s deployment, allowed 4 commodity (fan-cooled) GPUs to be attached to the same node while maintaining a sustainable thermal density. These nodes provide high-density, single precision compute capability to GPU-accelerated workloads that have a significant fraction of single-precision or less flops, such as Machine Learning and Molecular Dynamics applications.

Longhorn: Longhorn is associated with the Frontera project and built in partnership with IBM to support GPU-accelerated workloads requiring double precision or less flops. It is a self-contained system with 96 IBM AC922 nodes with 4 NVIDIA Tesla V100 GPUs per node.

An overview of the above TACC systems’ software infrastructure is in Table 4 below.

	Stampede2	**Frontera Primary Compute**	**Frontera Liquid Submerged**	**Longhorn**
Operating System	CentOS 7.8			Redhat 7.6
HPC Software suite	Custom	Custom	Custom/NVIDIA	Custom/IBM/NVIDIA
Provisioning	LoSF	LoSF		xcat
User Management	LDAP			
Job resource manager	SLURM			
Node Health Check	Custom			
End-user portal	TACC VisPortal			N/A
Cluster Monitoring	Nagios/TACC Stats			TACC Stats

Job usage reporting	TACC Stats/XDMoD			
Containers	Singularity	Singularity or Charliecloud or UDocker		Singularity
Numerical/Scientific Libraries	Custom + BLAS, LAPACK, SCALAPACK, FFTW	Custom + BLAS, LAPACK, SCALAPACK, FFTW	Custom + BLAS, LAPACK, SCALAPACK, FFTW,NVIDIA	Custom + BLAS, LAPACK, SCALAPACK, FFTW,NVIDIA
I/O Libraries	HDF5 (pHDF5), NetCDF (including C++ and Fortran interfaces), Adios			
Compiler Families	GNU (gcc, g++, gfortran) Intel (icc, icpc, ifort)	GNU (gcc, g++, gfortran) Intel (icc, icpc, ifort)	GNU (gcc, g++, gfortran) Intel (icc, icpc, ifort) NVIDIA (nvcc)	GNU (gcc, g++, gfortran) IBM (xlc, xlc++, xlf) NVIDIA (nvcc)
MPI Families	Intel MPI, MVAPICH2			IBM Spectrum MPI, MVAPICH2
Development Tools	GNU GDB,VTune, ARM FORGE			GNU GDB
Power/Energy monitoring management	IPMI Sensors			IBM

Table 4: Overview of TACC Software Infrastructure.

MACC is the Portuguese open-science driven advanced computing facility, primarily devoted to foster research and innovation in supercomputing, computational sciences, engineering and artificial intelligence. MACC is currently operating Bob, a less than 1PFlop system with 600 nodes and is acquiring a new machine Deucalion with two computational clusters: a x86-64 cluster including GPU compute nodes, and ARM compute nodes with Scalable Vector Extension(SVE) targeting at 10PFlop peak, a total of 2165 nodes and supporting a dependable high-performance 10PB storage. As Deucalion is expected to start production during the lifetime of this project, end of 2020, start of 2021, we describe both resources (Tables 5 and 6) to be used for BigHPC development and testing.

Resource	**Bob**	**Deucalion ARM**	**Deucalion X86**	**Deucalion**

				ACCEL
Nodes	800 Intel Sandy Bridge	1632 Fujitsu FX700 with A64FX	500 AMD EPYC Rome 7742	33 AMD EPYC Rome 7742
Interconnec t	Mellanox Infiniband FDR (50Gb/s)	Mellanox Infiniband HDR-100 (100Gb/s)	Mellanox Infiniband HDR-100 (100Gb/s)	Mellanox Infiniband HDR-100 (100Gb/s)
Filesystem	HOME 110 TB Lustre SCRATCH 350 TB Lustre	HOME 50 TB NFS SCRATCH 11 PB Lustre	HOME 50 TB NFS SCRATCH 11 PB Lustre	HOME 50 TB NFS SCRATCH 11 PB Lustre

Table 5: Overview of MACC compute clusters

Resource	**CPUs**	**RAM**	**GPU**	**Disk**
Bob	Intel Xeon CPU E5-2680	32 GB DDR3	N/A	256GB HDD
Deucalion ARM	Fujitsux A64FX 1 2 GHz CPU 48 cores/CPU	32 GB HBM2 @ 1 GHz	N/A	512 GB NVMe
Deucalion X86	AMD EPYC Rome 7742 2 2.25 GHz CPU 128 cores/CPU	256 GB DDR4-3200	N/A	240 GB SSD
Deucalion ACCEL	AMD EPYC Rome 7742 2 2.25 GHz CPU 128 cores/CPU	512 GB DDR4-3200	4 NVIDIA Ampere A100-4	240 GB SSD

Table 6: Detailed description of MACC Compute Nodes.

Bob: Bob is a supercomputer installed in a state-of-the-art data centre facility located in Riba de Ave fully powered by sustainable energy sources. It entered full production in Summer 2019. Bob is part of the former Stampede 1 of TACC.

Deucalion: Deucalion will include several computing technologies namely leading edge x86 and ARM processors, as well as accelerating coprocessors. It consists of components that offer high levels of efficiency to maximize application performance and minimize operational costs by including high-performance low energy processors available on the market today. Composed of three processing options, X86, X86+GPU, ARM it is able to address a wide range of applications spanning traditional HPC to AI and data analytics and more. It is expected to enter production at the end of 2020, start of 2021.

Table 7 describes the software supported by MACC supercomputers.

	Bob	**Deucalion ARM**	**Deucalion X86**
Operating System	CentOS 7.8	CentOS 8.1	
HPC Software suite		OpenHPC 2.x	
Provisioning	Custom	Warewulf	
User Management	LDAP		
Job resource manager	SLURM		
Node Health Check	NHC		
End-user portal	No	Open OnDemand	
Cluster Monitoring	Prometheus/Grafana		
Job usage reporting	Prometheus/Grafana/Open XDMoD		
Containers	No	Singularity or Charliecloud or UDocker	
Numerical/Scientific Libraries	OpenHPC libs + BLAS, LAPACK, SCALAPACK, FFTW	OpenHPC libs + Fujitsu optimized BLAS, LAPACK, SCALAPACK, FFTW	OpenHPC libs + Intel MKL**
I/O Libraries	HDF5 (pHDF5), NetCDF (including C++ and Fortran interfaces),		

	Adios		
Compiler Families	GNU (gcc, g++, gfortran)	GNU (gcc, g++, gfortran), Fujitsu Compiler suite	GNU (gcc, g++, gfortran), Intel Parallel Studio Cluster Edition
MPI Families	OpenMPI,MVAPICH2	OpenMPI, Fujitsu MPI,MVAPICH2	MVAPICH2, OpenMPI, Intel MPI**
Development Tools	GNU GDB,VTune	GNU GDB, Fujitsu debugger/profiler	GNU GDB, Intel Inspector, VTune
Power/Energy monitoring management	IPMI Sensors	Atos Smart Power Efficiency Management Suite	

Table 7: Detailed description of MACC Software.

4. Components Requirements

This section describes the specific requirements for each of the BigHPC components. Moreover, we also briefly discuss the integration requirements for the platform.

4.1. Orchestrator

The Orchestrator will be the main entry point for users and system administrators. Users resort to this component to deploy both HPC and Big Data jobs, while system administrators use this component to manage virtualization and storage resources provided by BigHPC.

General requirements. A deployment API will be needed so that users can deploy both HPC and Big Data jobs without requiring major changes to the way such is traditionally achieved. This API will be based on current de facto standards used in HPC and Big Data. Notably, this will require merging two distinct APIs, which is a challenge to be addressed by the BigHPC project. For instance, on the HPC side, Slurm and LSF are widely used batch system schedulers for clusters [YJG+03, I+19]. On the other hand, for Big Data platforms it is important to support the deployment of applications through widely used container technologies (e.g., UDocker [GCB+18] and Singularity [KSB+17]) and the execution of analytical tasks via traditional APIs (e.g, Hadoop Spark jobs, TensorFlow, Pytorch).

Moreover, a management API, also resorting to de facto standards (e.g., Ansible, Puppet, LOSF [R+19,P+19,L+19]), must provide system administrators with the necessary tools for managing the virtualized and storage resources provided by the BigHPC platform.

Integration Requirements. The Orchestrator will be collecting application and cluster resource metrics from the Monitoring component, which will be provided to two modules - the Virtualization Manager and the SDS Manager. These modules will be integrated into the orchestrator through an internal API that will be generic so that other specialized management modules, outside the scope of this project, can be easily supported in the future.

Virtualization Manager requirements. The virtualization (container) manager will use resource and past job performance data collected from the monitoring component to correctly place containers on one of the resources that is currently available. This management component will choose container placement based on a variety of variables including but not limited to, CPU architecture, storage space requirements, memory requirements, interconnect, RDMA hardware/software stack, availability of specific accelerators, theoretical or past performance metrics, and the availability of software to support the above.

Storage Manager requirements. The Storage Manager will allow users and system administrators to submit storage policies within the scope of a specific job, a set of jobs or for the overall infrastructure. Namely, the BigHPC project will need to define a high-level language so that users can specify storage policies for I/O prioritization, rate-limiting and data placement. These are further discussed in Section 4.4.

4.2. Monitoring

The monitoring component is the element that will be responsible for keeping all the systems under close and continuous observation, thus allowing HPC managers to keep track of the clusters occupancy and their health and to the users their job execution.

General requirements. The monitoring of HPC is intended to be: a) non-intrusive by not requiring the re-implementation or re-design of current HPC cluster software; b) efficient while monitoring the resource usage of thousands of nodes without imposing significant overhead in the deployed HPC workloads; c) able to store long-term monitoring information for historical analysis purposes; and, d) to provide real-time analysis and visualization about the cluster environment.

Integration requirements. The collected metrics and post analysis provided by this component will be consumed by the Orchestrator and then used by the management tools of the clusters. Monitoring will also comprise the aggregating of the gathered information at different infrastructural levels. At the server level, with statistics for the node state such as node up/down status, CPU, memory, network/storage and I/O usage; at the system/hardware level, collect environment statistics such as power consumption, temperature, or AC unit operation logs. These metrics will be correlated with the resource usage of specific jobs, applications, users and projects.

From the analysis and correlation of these metrics it will be possible to acquire knowledge on the cluster performance; improve scheduling and applications management; and predict malfunctions. Furthermore, such analysis will trigger real-time alarms for system administrators identifying potential performance or reliability issues.

To provide such a solution BigHPC will need to address distinct challenges. First, it will provide support for persisting and analyzing long-term monitoring data which will allow leveraging machine learning techniques to improve the prediction of possible performance bottlenecks and failures.

Secondly, the solution design will need to be scalable, efficient and non-intrusive, thus allowing it to efficiently monitor large-scale and heterogeneous infrastructures while imposing a negligible performance overhead and avoiding the need to change (re-implement or patch) deployed software and hardware components. Finally, the monitoring component will export a rich API so that the Orchestrator modules and system administrators, the latter through the proper visualization tools (e.g., Kibana, Grafana), are able to analyze and take advantage of the collected metrics

Storage Monitoring. The Storage Manager module will collect infrastructural metrics from the BigHPC's monitoring component. Storage related metrics such as I/O bandwidth, throughput, latency, space quotas, and access patterns will be collected at different tiers and granularities of the HPC center. Namely, these metrics will be collected at the application, compute node's local storage mediums, and at the shared parallel file system level.

Table 8 further details the monitoring requirements for BigHPC.

Monitoring		
Challenge	**Requirements**	**Current Approach**
Real time metrics	Provide metrics that present the node state at a given instant	The current polling intervals are not real time metrics and others only provide past job metrics
Non intrusive	Impact as little as possible the performance of running jobs	Capture metrics with bigger intervals to lessen the impact
Long time	The data resulting from the storage of logs will be stored for	Logs are discarded after specific intervals making it

	future analysis	impossible to perform post job analysis
Granularity	Provide short and long time metrics and also cluster vs node vision according to the user or manager needs	The granularity depends on the type of tool used to monitor the cluster
Visualization	Provide visualization either for managers or job owners in a simple web environment without the need to install additional tools	Self tailored applications with complex installation scripts and long learning curves
Heterogeneity	Work in all type of x86 and ARM clusters	Most HPC centers develop their own set of tools as a monitoring solution. There are some proprietary solutions from hardware vendors
Storage and Network Monitoring		
Challenge	**Requirements**	**Current Approach**
Measure I/O performance of the cluster	Monitor the IO performance of the system identifying possible bottlenecks and malfunctions	There are approaches on the Lustre server and also on the node but not targeted to the job manager or job owner but to the luster server manager
Network	Determine the network performance and diagnose possible faults while moving data from the storage servers to the nodes	This monitoring is usually made at network level and not integrated in the node monitoring

Table 8: BigHPC monitoring requirements.

4.3. Virtualization

The primary goal of creating the Virtualization Manager is to be able to run jobs on multiple different clusters with different hardware and software transparently for the user. The challenge of building the Virtualization Manager is combining all of the different parameters and data that go into building an application on one machine and then using that to create an image and place a job on one of a few types of machines. The two primary components of the Virtualization Manager are the Container Curator and the Container Deployment manager.

Container Curation: In order to construct a container that can run on many HPC clusters we must first enumerate the differences between the clusters we are targeting. Based on these differences we will create container templates for different types of systems. Some of these parameters are easily implemented on the command line of an executing container. However, some will require changes to the application compile commands. Hardware specific details such as the CPU vector instruction support or the RDMA hardware can necessitate compile time changes. Depending on the application it may be worthwhile to compile multiple versions of an application and then choose the appropriate version at runtime.

Another per-cluster difference is typically how the storage is tiered. Not all clusters have local disks or even a work environment. Regardless of that, small changes, like mount point targets can cause confusion in containers. This can be mitigated by a standardized container mount point location and bind mounting the relevant file systems within the container, however, we must first create a matrix of filesystems at each site and how they map to the container defaults so we can programmatically change the file system bindings at job run time.

As a solution to the problems above we will characterize each important parameter as hardware-specific or generalizable within the Virtualization Manager. Then container templates for each cluster containing the generalized and the hardware-specific parameters will be provided to application experts that can implement their specific application within our BigHPC containers.

Container Deployment: With a working BigHPC container the container deployer must have enough information in order to intelligently place a container on one of the supported HPC resources. This data, ostensibly from the monitoring tool, will need to encompass both static (CPU Sku, RDMA hardware, storage mount targets, etc) and dynamic (resource usage data, resource availability, storage space availability, etc) parameters in order to: a) choose the resources most closely aligned with the needs of a job; b) select the correct BigHPC container and; c) correctly configure the container for the specific cluster.

Some examples of possible allocation decisions based on resource usage data are:

- Prefer empty nodes

- Prefer nodes/sockets with available memory
- Prefer socklet with idle cores
- Prefer nodes/sockets with higher or lower memory bandwidth
- Prefer nodes with specific fabric types

Because of the variety of different applications and the large expense of HPC clusters it is incumbent upon us to use the available resources as efficiently as possible. In addition to the resource usage data this effort includes characterizing each workload and feeding that data into the virtualization manager.

Containers		
Challenge	**Requirements**	**Current Approach**
Container software have specific OS requirements and some features of containers are only enabled with some kernel versions	The containers in use in this project will only work on specific versions of the Linux Kernel	One is confined by the resources that they have available if the Linux Kernel is not new enough containers cannot be run
Containers are locked into a specific architecture	Must be able to deploy to a non-x86 cluster (arm64).	Containers are typically created on the cluster they are used on so the architecture is implied during the creation.
Data Movement and Storage		
Challenge	**Requirements**	**Current Approach**
Data ingress/egress can be large and consume large amounts of time and resources	Data locality has to be built into the decisions of the virtualization manager. It is impractical to copy a large dataset multiple times so this may confine jobs to a cluster where the dataset exists.	Typically users run on one cluster so once the data is there this is not an issue.
Data copying to local disks from local shared storage	Need to be able determine the size of the input and output data in order to make a decision about where data should be for the job (local disk, shared storage, somewhere offsite)	On a job by job basis this is balanced between time to copy and IO performance of different tiers of file systems.

Resource Utilization		
Challenge	**Requirements**	**Current Approach**
Collecting Cluster-level performance metrics	Have enough information about either **a.** theoretical capacity or **b.** regularly collected performance metrics (preferably both)	These types of metrics are typically done during acceptance during bring up and/or after a maintenance and they are rarely automated
Multi-node jobs within containers	The virtualization manager must be able to match versions of MPI with RDMA hardware as well as ethernet devices.	There are typically good defaults for most clusters that users can load to use MPI.
Performance		
Challenge	**Requirements**	**Current Approach**
Different CPUs have different vectorization instruction sets	The virtualization manager must be able to map specific CPU SKUs to vectorization instruction sets (AVX512, AVX2, SVE, etc)	This is typically done at compile time for each application. Some applications will have support for these and some will not.
Accelerators have software drivers that need to be exposed within the container	The Virtualization Manager must be able to map a specific accelerator to a specific partition in a cluster, then be able to run a container with the accelerator software stack bound by the container	Containers are typically built on the cluster where the software stack exists.

Table 9: BigHPC virtualization requirements.

Broadly, and as depicted in Table 9, there are four major areas of challenges for the Virtualization Manager: Containers, Data Movement and Storage, Resource Utilization and Performance.

Container Challenges: Containers were originally built upon the Linux LXC extensions, however, some implementations have created their own libraries to use the LXC extensions along with other extensions (libvirt, cgroups, selinux, etc). This ecosystem means that there are many options for running containers, however, finding the correct container for the correct kernel versions and library versions is a non-trivial task. Luckily, it is only needed once from each participating cluster. A partial solution to this is to pick one (or two) container types we use and request each site enable the functionality that is needed. Containers are also, implicitly, Linux based. While docker does run on Windows, for our use case Linux is the only choice.

Data Movement and Storage Challenges: Data locality is core to HPC. Having the data, in a fast (close) place is intrinsic to most tightly coupled (MPI) jobs. This can create a problem if the input or output of the application is very large. Additionally, intercontinental data transfer has been slow, historically. Moving large datasets from site to site across the internet needs to be carefully considered. We may need to consider some applications with datasets that are too big to transfer, local to a specific site.

Another challenge in data movement is that often clusters do not have matching file system mounts. The Virtualization Manager will have to map each clusters' mount point to a set of generalized mount points within the BigHPC containers.

Resource Utilization Challenges: Most resource utilization challenges require us to make generalizable abstractions for HPC-centric hardware such as RDMA networks and accelerators. As described above, this will be mitigated by either including the site-specific hardware access and drivers within site-specific containers or we will find a general way to access the hardware and software needed without a site-specific container.

Another challenge revolves around how we get up-to-date performance information from all of the clusters to inform the job placement decisions within the Virtualization Manager. In order to mitigate this. There are a number of solutions to this problem ranging from getting theoretical peak data and leaving the variables static to getting dynamic performance measurements on a regular basis.

Performance Challenges: The last challenge is getting full CPU and Accelerator performance of different CPU SKUs in an automated way. One option would be to compile for multiple instruction sets and choose the correct version during runtime. Another would be to have site-specific containers that have any vectorization built in.

4.4 Storage

In this section, we will provide the challenges and requirements for BigHPC's storage component, while focusing on state-of-the-art infrastructures from MACC and TACC, namely the Deucalion and Frontera supercomputers. However, most of these requirements are applicable to other supercomputers from these and other advanced computing centers.

HPC infrastructures are vertically designed, composed of several layers along the I/O path that provide an assortment of compute, network, and storage functionalities. Storage resources are organized into different tiers with different purposes and requirements.

RAM and local storage tier. At the higher tiers, compute nodes have access to local RAM disks and persistent storage (e.g., SSD or HDD disks). These act as temporary storage mediums that can be used by jobs and applications during runtime. After the job's completion, these storage devices are cleaned up and data is no longer persisted.

Requirements. These local storage mediums are typically accessed by a single job, running at the compute nodes. In the BigHPC project, we aim at supporting multiple jobs in the same node, by leveraging virtual containers, that will be sharing these resources.

The sizes of volatile disks depend on the amount of RAM at compute nodes that can be spared by deployed jobs (i.e., currently the RAM size is 192 GB for TACC's Frontera and 256 GB for MACC's Deucalion compute nodes). The size of persistent local disks ranges from 144GB to 512GB for TACC and MACC supercomputers. Frontera supports both SSD and HDD disks while Deucalion will have only SSD disks. The speed of each disk is highly dependent on the hardware configurations. Nevertheless, it is important to note that this speed varies across faster mediums (RAM disks, SSD NVMes) and slower mediums (SATA SSDs and HDDS).

Shared file system tier. In the case where data does not fit these local storage mediums, it needs to be available across different compute nodes, or it needs to be persisted between the execution of job's, a lower storage tier exposed as a shared filesystem is provided to users. In both TACC and MACC supercomputers, the Lustre distributed file system, which is deployed at the storage nodes, is used to export this shared interface to users.

HPC users have access to three different Lustre storage environments. The *Home* environment is used mainly for storing the binaries of applications and scripts to execute them as jobs. The *Work* (sometimes referred to as project or campaign storage) environment is used to store medium term data from jobs and applications, while the *Scratch* environment is used to store runtime data of jobs and applications.

Requirements. The Lustre deployment is shared by thousands of jobs from different users. In Deucalion, the Lustre deployment provides more than 11 PB of storage space with a sustained throughput of 260 GB/s. In Frontera, it provides more than 44 PB of storage space with a sustained throughput of 240 GB/s.

Finally, in terms of interfaces the RAM and local disks of compute nodes provide a block-device interface, while the Lustre volumes provide a POSIX compliant API.

Ideally, an user's job would leverage the different tiers depending on the necessary speed and persistency requirements. In practice, the shared filesystem is usually overutilized thus creating I/O fairness issues that can significantly cripple the experience of HPC users. Such problems are amplified by the emerging support of data-centric applications for data analytics and machine learning that further stress the available storage resources. This is one of the main reasons why storage performance (i.e., the access and retrieval of data from jobs and applications) is now a major bottleneck that is limiting the performance of current HPC applications [ECS+17, YDI+16].

In Table 10, we further detail the storage challenges and requirements of state-of-the-art HPC centers. BigHPC has the goal of addressing these requirements in order to alleviate the current performance bottlenecks of HPC storage solutions.

Performance		
Challenge	**Requirements**	**Current Approach**
Ensuring I/O fairness when multiple jobs are accessing shared storage resources (e.g., compute nodes local storage or the shared file system)	Runtime and dynamic distribution of storage resources' bandwidth across multiple jobs Sustained throughput/latency across multiple jobs	Typically this is a manual process that is triggered when an application is overloading the storage resources Automatic tools are static and cannot adapt to changes in workloads and job scheduling
Ensuring I/O prioritization policies for applications requiring different storage performance levels	Specialized distribution of storage resources' bandwidth across multiple jobs Different levels of sustained throughput/latency given the job's priority requirements	This requirement is currently not met by the tools available in HPC centers

Improved storage performance for small file workloads	Small file workloads can easily overload the shared file system metadata servers. Random access workloads on small files, with several metadata operations (e.g., *open* and *close*) are major performance bottlenecks	Adding more metadata servers and sharding metadata across these for improved scalability and performance
Transparency and Automation		
Challenge	**Requirements**	**Current Approach**
Transparent and automatic data placement across storage tiers	Ensuring that data placement across the different storage tiers is transparent to users Efficient and automatic data placement for increased storage performance and for reducing the pressure over the shared file system	Data placement is currently done by users which requires specialized knowledge about the application storage requirements and the HPC infrastructure
Transparent and automatic enforcement of I/O policies	I/O prioritization and fairness must also be done transparently to users and system administrators	The amount of storage bandwidth used by a given job is typically done manually by users and HPC administrators
Integration		
Challenge	**Requirements**	**Current Approach**
Data plane integration with storage APIs	BigHPC's data plane must be integrated with applications and HPC storage layers in order to control their I/O flows and ensure the desired performance and data placement policies This will require supporting standard POSIX and block device interfaces	Current HPC storage solutions are not SDS-enabled and only target the shared file system layer, thus not requiring this type of integration
SDS control and data planes	The data and control flows of HPC storage resources will be managed by a SDS control and data plane, respectively. The integration of these	Current HPC storage solutions are not SDS-enabled and only target the shared file system layer

	modules in terms of design and APIs needs to be accounted for	
Orchestrator and job manager	The SDS control plane must be integrated with BigHPC's Orchestrator component. Integration with the HPC job manager (e.g., SLURM) is also a requirement for ensuring efficient data placement and I/O policies features	Current HPC storage solutions are not SDS-enabled and only target the shared file system layer thus, not requiring this type of integration
Monitoring	The SDS control plane will collect runtime metrics from the HPC infrastructure. Namely, from BigHPC's Monitoring component Metrics related to storage usage (e.g., bandwidth, throughput, latency, space quotas) must be collected at the jobs and storage resources granularity	These metrics are already collected but are provided to system administrators' dashboards and not to an SDS controller

Table 10: BigHPC storage requirements.

Requirements can be divided in three major groups, namely Performance, Transparency and Automation, and Integration.

Performance. In order to alleviate the I/O interference currently observed for HPC shared storage resources, while improving the performance of data-intensive jobs, BigHPC will need to address the following requirements.

Multiple jobs may be running at the same compute node thus concurrently accessing the same local storage mediums. Moreover, these jobs will be storing and retrieving data concurrently from the shared HPC file system. Thus, BigHPC SDS solution will need to provide mechanisms for minimizing I/O interference between jobs and provide a fair usage of shared storage resources. This will be achievable through novel I/O rate-limiting and scheduling policies that will control the storage throughput and/or latency of jobs while having a global visibility of the HPC infrastructure and deployed jobs. The policies will be enforced at runtime and in a dynamic fashion thus, considering the lifetime of jobs and variations on their storage workloads.

As another goal, the previous policies will be extended to provide different levels of I/O performance according to each job's requirements and the available capabilities of HPC storage resources. Namely, it will be possible to define minimum, maximum and sustained I/O throughput and latency requirements at the job granularity (I/O prioritization).

Finally, the SDS solution will also have the goal of alleviating the performance bottleneck currently imposed by jobs considering random access workloads on thousands of small files. These workloads are metadata intensive and can easily overload the HPC shared file system metadata servers.

These requirements will be achieved by leveraging on the work on SDS systems, which has been successfully applied to cloud computing infrastructures, but it is still in a very early stage in the HPC field [MPP+20]. The outcome will be a SDS storage solution that is highly programmable and can automatically adapt to different storage workloads. By having an holistic vision of HPC jobs and infrastructural resources, this solution will ensure quality-of-service and performance policies according to the requirements of each job/application while providing fairness and prioritization in the access to shared storage resources.

Transparency and Automation. Currently, most of the storage optimizations applied over HPC storage resources are done manually by the users or system administrators. This requires a deep knowledge about the applications/jobs being deployed and the HPC infrastructure and software supporting these. BigHPC will advance the automation and transparency of storage resources management by meeting the following requirements.

The decision on what storage tiers (i.e., local storage mediums at compute nodes or shared file system) should be used for improving the storage performance of a given job is done manually. Moreover, this decision may require significant code changes to the application being deployed, and requires deep knowledge about the persistency guarantees offered by each HPC data tier and on how data can be migrated across these. Thus, the simplest solution is to avoid using the temporary storage tiers while storing all data in the shared file system, where data persistency and access across compute nodes is always guaranteed. However, this solution creates further pressure on the shared storage backend, leading to a severe I/O bottleneck.

Therefore, another goal of BigHPC is to ensure that data placement can be done automatically and in a transparent fashion thus, avoiding the manual intervention of users and system administrators, and the need to modify the code of user applications. Moreover this placement needs to be done efficiently in order to increase the storage performance of applications while reducing the pressure over the shared HPC file system.

Finally, transparency and automation will also be key requirements for the I/O prioritization and fairness policies described previously. Also, the SDS solution being devised in this project will provide the necessary tools for accomplishing these requirements.

Integration. As explained previously, BigHPC storage solution will decouple the I/O control and data flows into two major components, control and data planes. The data plane is a programmable multi-stage component distributed along the I/O path (i.e., placed between the applications and distinct storage tiers) that enforces fine-grained management and routing properties dynamically adaptable to the infrastructure status, such as rate limiting, I/O prioritization, bandwidth aggregation and data placement.

The control plane (also referred as Storage Manager in this document) comprehends a logically centralized controller with system-wide visibility that orchestrates the overall storage infrastructure (i.e., data plane stages and storage resources) in a holistic fashion. The control plane is scalable and enforces end-to-end storage policies, tailored for attending the requirements of exascale computing infrastructures.

These two planes will require integration among them in terms of design and communication APIs.

Moreover, since the data plane will be deployed between applications and storage layers, it will require a transparent integration with these. Ideally this will be a non-intrusive integration which will require exporting standard POSIX and block-device APIs.

The SDS control plane must be integrated with BigHPC's Orchestrator since this component will be responsible for managing jobs and virtualization at the HPC cluster. Namely, the integration with the Orchestrator and the HPC job manager (e.g., SLURM) is necessary for knowing where (i.e., in which compute nodes) jobs will be deployed and to ensure the desired data placement and I/O policies features for those jobs. The Orchestrator will also be responsible for exporting the storage management API that will allow users and system administrators to define storage policies for different jobs and the overall infrastructure.

Finally, in order to efficiently enforce storage policies, the control plane will collect infrastructural metrics from the BigHPC's monitoring component. Storage related metrics such as I/O bandwidth, throughput, latency, space quotas, and access patterns will be collected at different tiers and granularities of the HPC center. Namely, these metrics will be collected at the application, compute node's local storage mediums, and at the shared file system level.

4.5. Platform Integration

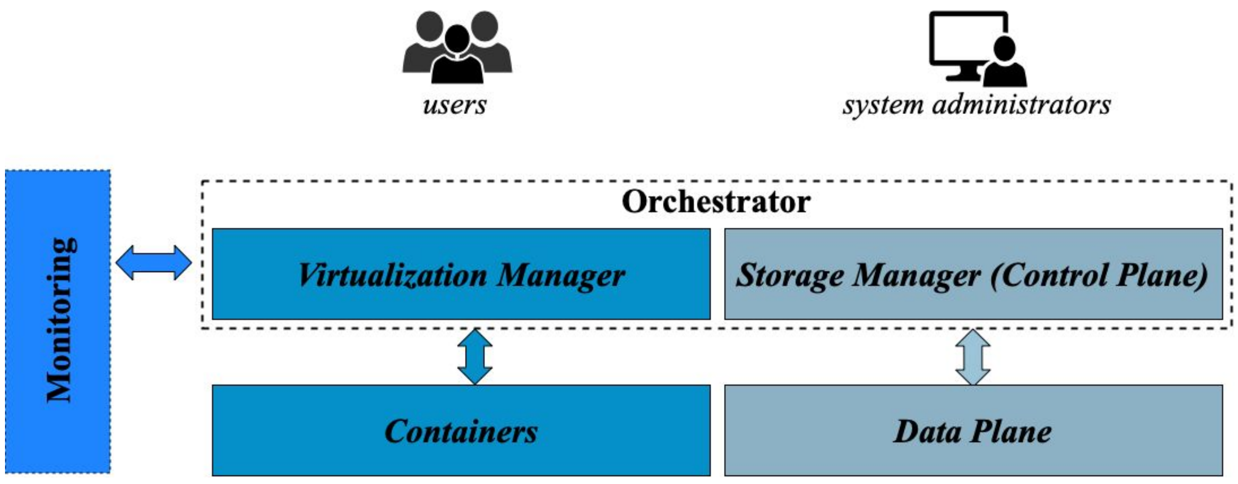

Figure 2: Integration dependencies between BigHPC components

The Orchestrator, Virtualization Manager, Storage Manager and Monitoring solutions will be provided as fully-functional prototypes. These prototypes will be integrated into a single software framework that will be validated by resorting to the infrastructures of both TACC and MACC centers. For this to be possible, as detailed in Figure 2, the Virtualization Manager and Storage Manager (SDS control plane) will be bundled into the Orchestrator component, thus exporting a unified point of contact for users and system administrators that wish to manage their jobs and infrastructural resources. Moreover, the Monitoring component will also require integration with the Orchestrator in order to provide real time metrics for jobs, virtualization and storage management.

Also, the project will provide real use-cases and a pilot for testing the proposed framework in a real setup. They will provide real compute and data intensive applications that are already being used in production and whose behaviour is well understood. These pilot applications will be representative of the users' communities and will allow the validation and comparison of the BigHPC framework against conventional HPC systems. This integration and validation step will ensure that a coherent and cohesive software prototype is available at the end of the project and can be later productized and explored commercially by Wavecom.

5. Conclusion

This deliverable details the requirements for the BigHPC platform. The major goal of the project is to ensure that the proposed solution is compliant with the management needs of tradicional HPC jobs while meeting the novel functional and non-functional requirements brought by Big Data applications.

The solution must also be compliant with state-of-the-art infrastructures, such as the ones supported by TACC and MACC, and the legacy software supported at these centers. Furthermore, it must simplify the deployment and management tasks done by users and system administrators, while maintaining easy-to-use and familiar interfaces.

As another contribution, this deliverable discusses the challenges and requirements that must be addressed by BigHPC's Monitoring, Orchestration, Virtualization and Storage components in order to improve the performance and management of current HPC supercomputers.

The analysis presented at this document, for the general platform and for each of its main components, will be key to drive the next major step of the project, namely the design, architecture and APIs of the BigHPC framework.

References

[ECS+17] Joseph, E., Conway, S., Sorensen, B., Thorp, M., 2017. Trends in the Worldwide HPC Market (Hyperion Presentation). HPC User Forum at HLRS.

[GCB+18] Gomes, J., Bagnaschi, E., Campos, I., David, M., Alves, L., Martins, J., Pina, J., López-García, A. and Orviz, P., 2018. Enabling rootless Linux Containers in multi-user environments: The udocker tool. Computer Physics Communications, 232, pp.84-97.

[I+19] IBM, Armonk, NY, USA. IBM Spectrum LSF Suites. (2019) [Online]. Available: https://www.ibm.com/us-en/marketplace/hpc-workload-management. Accessed on: 2020.

[KRV+15] Kreutz, D., Ramos, F.M., Verissimo, P., Rothenberg, C.E., Azodolmolky, S. and Uhlig, S., 2015. Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), pp.14-76.

[KSB+17] Kurtzer, G.M., Sochat, V. and Bauer, M.W., 2017. Singularity: Scientific containers for mobility of compute. PloS one, 12(5), p.e0177459.

[KWG+13] Katal, A., Wazid, M. and Goudar, R.H., 2013, August. Big data: issues, challenges, tools and good practices. In 2013 Sixth international conference on contemporary computing (IC3) (pp. 404-409). IEEE.

[L+19] K. Shultz, “LosF: A Linux operating system Framework for Managing HPC Clusters”, [Online]. Available: https://github.com/TACC/losf. Accessed on 2020.

[MPP+20] Macedo, R. Paulo, J. Pereira, J. and Bessani, A. 2020. A Survey and Classification of Software-Defined Storage Systems. ACM Comput. Surv. 53, 3, Article 48 (June 2020), 38 pages.

[NCR+18] Netto, M.A., Calheiros, R.N., Rodrigues, E.R., Cunha, R.L. and Buyya, R., 2018. HPC cloud for scientific and business applications: Taxonomy, vision, and research challenges. ACM Computing Surveys (CSUR), 51(1), p.8.

[OG+15] Osseyran, A. and Giles, M. eds., 2015. Industrial applications of high-performance computing: best global practices (Vol. 25). CRC Press.

[P+19] Puppet, Portland, OR, USA. Puppet.(2019) [Online]. Available: https://puppet.com. Accessed on: 2020.

[R+19] Red Hat Ansible, Durham, NC, USA. Red Hat Ansible. (2019) [Online]. Available: https://www.ansible.com. Accessed on: 2020.

[RD+15] Reed, D.A. and Dongarra, J., 2015. Exascale computing and big data. Communications of the ACM, 58(7), pp.56-68.

[YDI+16] Yildiz, O., Dorier, M., Ibrahim, S., Ross, R. and Antoniu, G., 2016, May. On the root causes of cross-application I/O interference in HPC storage systems. In 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (pp. 750-759). IEEE.

[YJG+03] Yoo, A.B., Jette, M.A. and Grondona, M., 2003, June. Slurm: Simple linux utility for resource management. In Workshop on Job Scheduling Strategies for Parallel Processing (pp. 44-60). Springer, Berlin, Heidelberg.