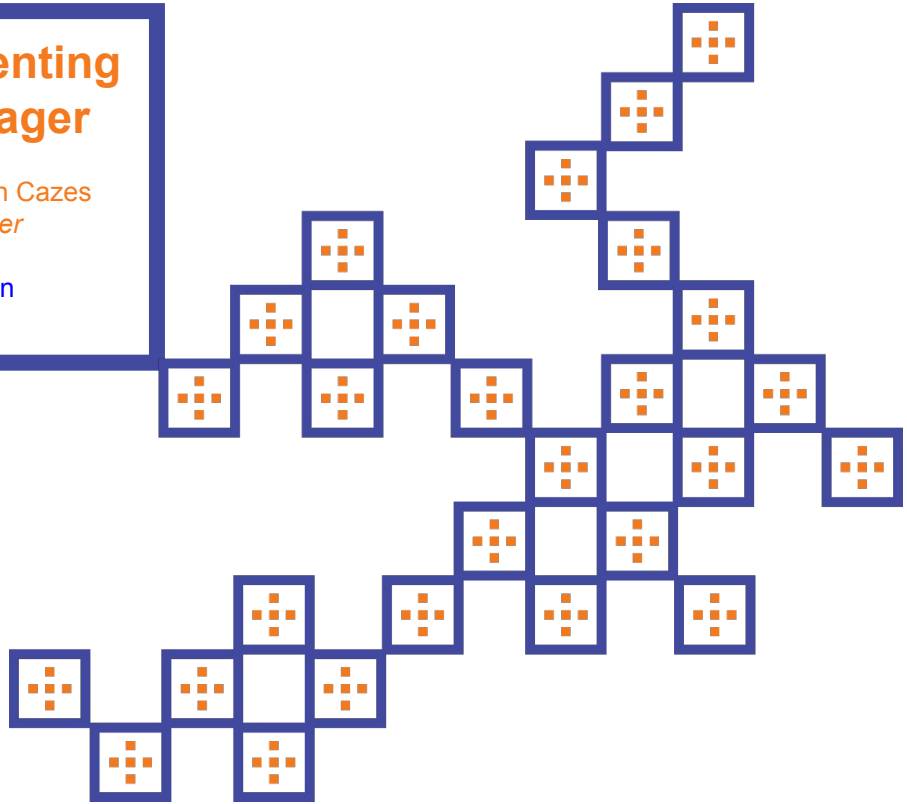
Experiments in Implementing the BigHPC Virtual Manager

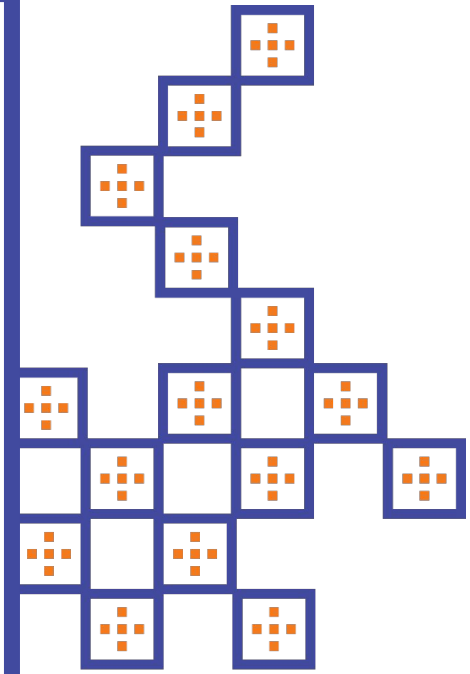
Amit Ruhela, Stephen Lien Harrell, John Cazes
Texas Advanced Computing Center

Sangamithra Goutham, Om Saran
UT Austin, Texas

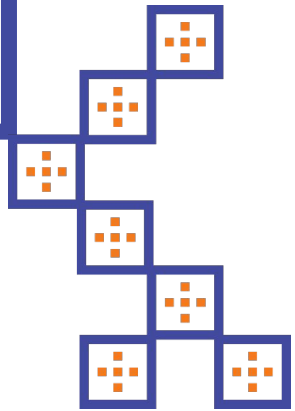


Outline

- HPC, Big Data, and BigHPC
- Virtualization Manager (VM)
- Design Alternatives
- Selected Design
- Future Plans

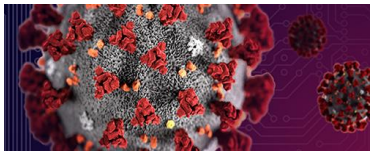


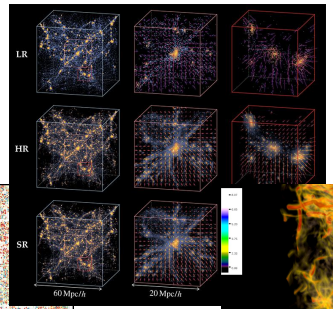
HPC, Big Data, and BigHPC

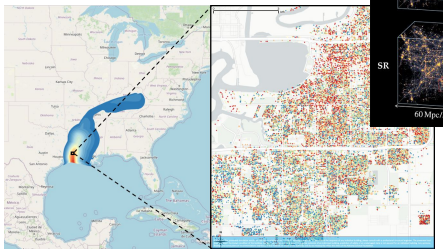


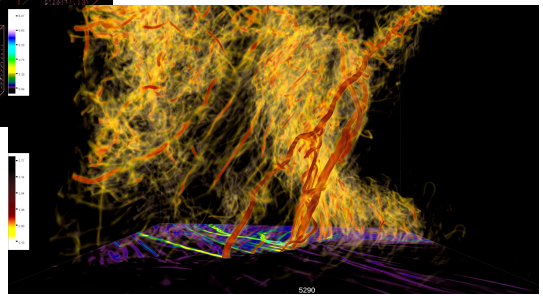
What is High Performance Computing (HPC)?

- Hundreds to thousands of tightly connected servers (compute nodes)
- Tens to thousands of researchers simultaneously running distinct applications
- Computational power to solve the largest problems
- Traditional HPC Applications include but not limited to
    - Weather and Climate (hurricanes, tornados, ice sheet melting, forecasting)
    - Astronomy, Cosmology, particle physics (LIGO, LHC, Webb)
    - Fluid dynamics (engineering, turbulence, Formula 1, rockets)
    - Molecular dynamics (cells, proteins, viruses)
    - Material science (superconductors, semiconductors, batteries)
    - Geosciences (earthquakes, mining, drilling)



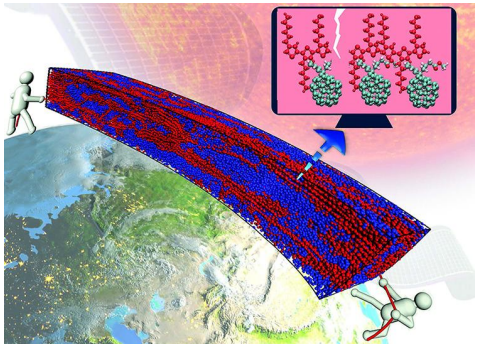


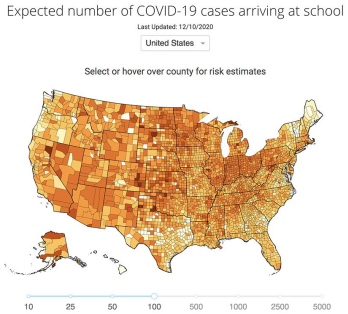


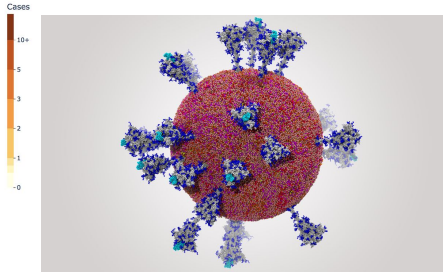


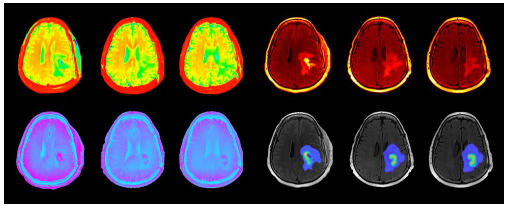
HPC and Big Data

- Big Data is increasingly integrated with HPC through Machine Learning Requirements
  - Machine Learning allows old problems to be solved in new ways
    - Optimize solar cell energy production
    - Accelerate protein folding calculations
  - Machine Learning techniques allow new problems to be solved
    - Image recognition (telescopes, MRIs, etc.)
    - Earthquake displacement prediction
- Big Data is also increasingly used in computational research
  - Data assimilation
  - Epidemiology
  - Drug Discovery









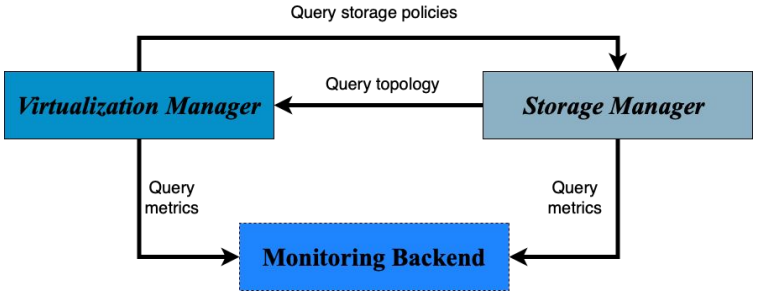
Challenges of Supporting HPC and Big Data:
Big Data Software and Workloads Don't Fit in HPC Environment

- HPC environment expects regular and predictable IO
  - Infrequent, small reads/writes
  - Very infrequent, large reads/writes
- HPC expects flexible software that can be customized to many environments (portable)
  - Modest software requirements
  - Compiled by researchers on many systems
  - Execution requires a handful of files
  - Customizable to utilize system hardware
- HPC expects workloads to share a system with other, distinct workloads
  - Never require privileged access
  - Workloads have defined runtime on defined resources
  - Workloads do not utilize persistent services

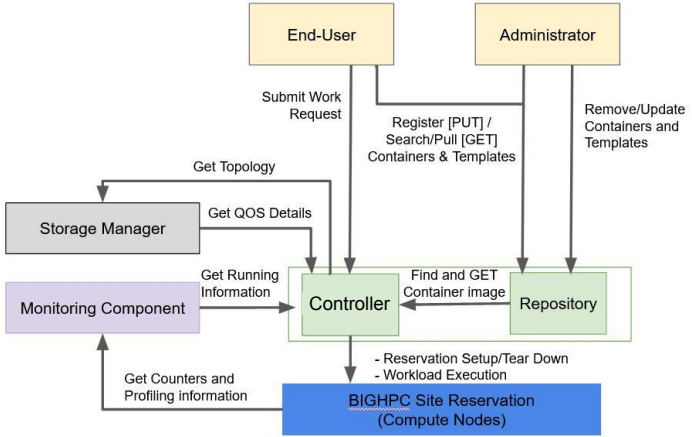
- Big Data and Machine Learning workloads have irregular IO
  - Random, frequent, small reads/writes
  - Many small files
- Big Data and Machine Learning software is highly specialized and complex (non-portable)
  - Complex software requirements
  - Cannot be compiled except by experts on specific machines
  - Execution requires many, many small files
  - Too rigid to utilize system hardware
- Big Data and Machine Learning workloads expect exclusive access to systems
  - Often require privileged access
  - Irregular workloads requiring unpredictable resources
  - Require persistent services such as databases and webservers

BigHPC: Management Infrastructure Composed of 3 Components to Address Challenges

- Monitoring Backend: What workloads are running and what resources are they consuming?
  - Facilitates Quality of Service for shared resources (nodes, network, IO, memory)
- Storage Manager: Controls IO of workloads
  - Enforces Quality of Service for IO
  - Enforces stability of storage systems
- Virtualization Manager: Curates software and places workloads
  - Enforces Quality of Service for nodes, network, memory
  - Enables workloads to utilize persistent services
  - Enables complex, rigid software to run in diverse environments and hardware in an (near) optimal way
    - *TACC alone has more than 6 entirely different processor and network architectures that users may run on!*



BigHPC: Virtualization Manager (VM)

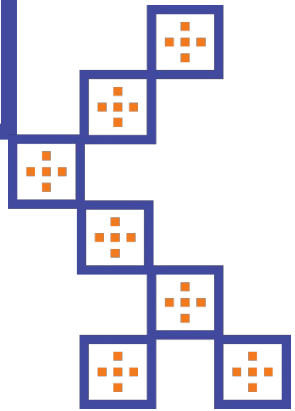


- 2 Subcomponents
  - Controller (Scheduler)
  - Repository
- VM Controller
  - Users request workloads to be run through controller/Scheduler
  - Scheduler places containerized workloads based on Monitoring & Storage Manager input
- VM Repository
  - Enables *Containerization:* Containers can help users to build and run applications optimized for specialized hardware (customized for a specific HPC system)
  - Optimized containers provided by Repository to BigHPC users at build/run time
  - Containers may be used on thousands of nodes and dozens of architectures - above tasks are not straightforward

Goal : How do we design Virtualization Manager that ensure optimal container performance to BigHPC users?

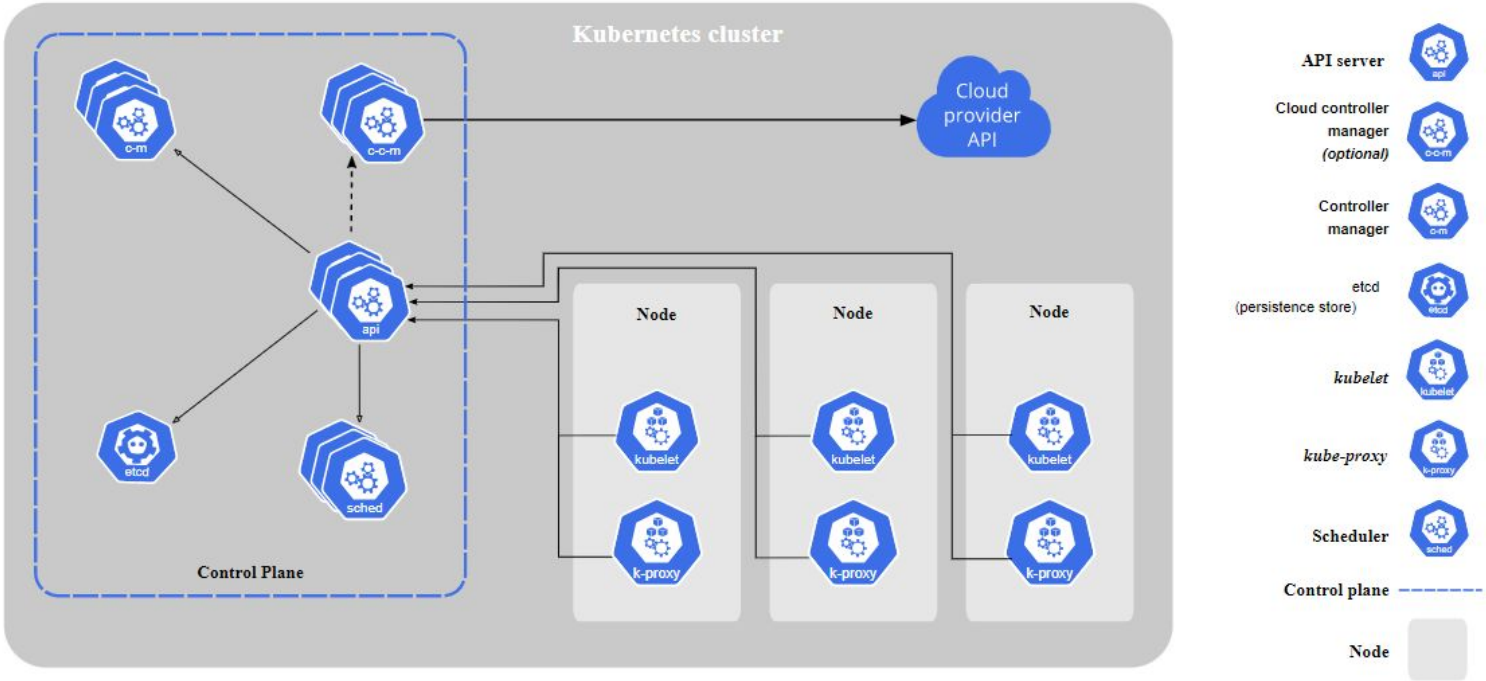
Design Alternatives - Virtualization Manager

- Kubernetes
- HTCondor
- Jenkins



Approach 1 - Kubernetes

- Open-source system for automating deployment, scaling, and management of containerized applications



Approach 1 - Kubernetes - Contd.

Limitations:

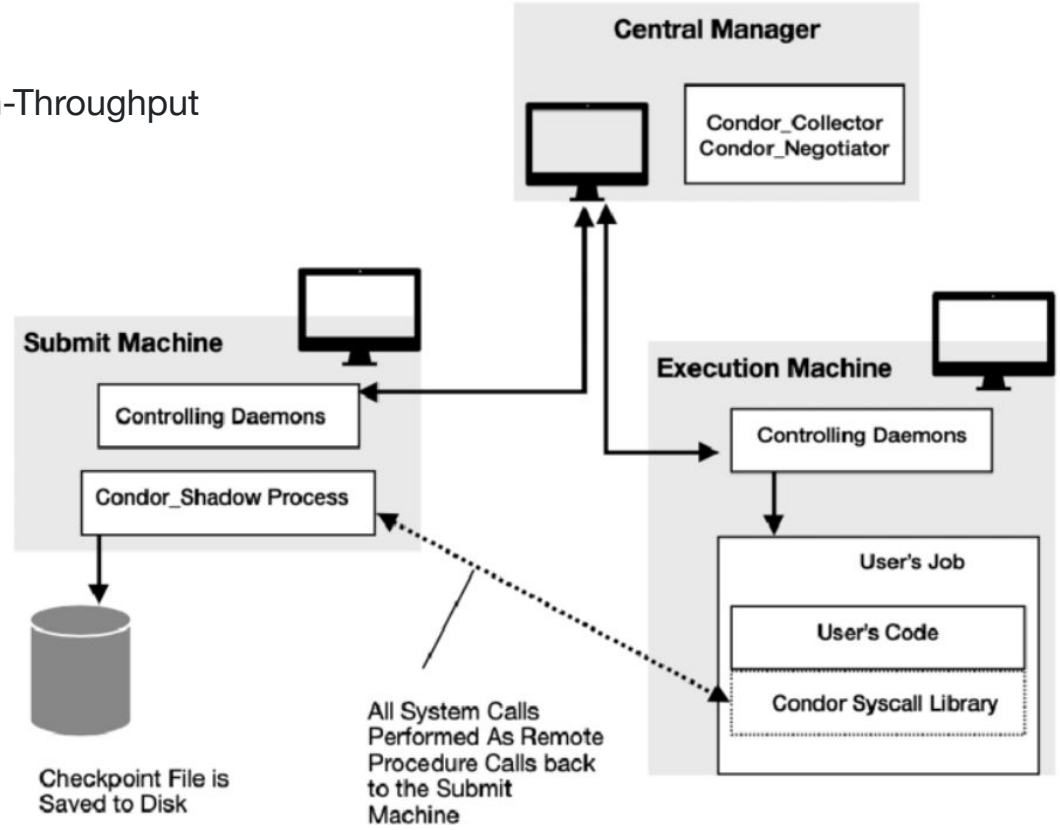
- Kubernetes executes Kublets (also called node agents) on the compute nodes which necessitates a Container Runtimes e.g Containerd, CRI-O for execution.

- The container runtime engines necessitates the privilege(root) access which is mostly prohibited by HPC administrators

- Kubernetes adds additional layer in BigHPC framework which is challenging to customize, manage, and troubleshoot in the HPC environment, leading to unreliable behavior and increased performance overheads. Furthermore, BigHPC don't require all the complex features of Kubernetes runtimes and is therefore focused for lightweight and simplicity.

Approach 2 - HTCondor

- Software system that creates a High-Throughput Computing (HTC) environment



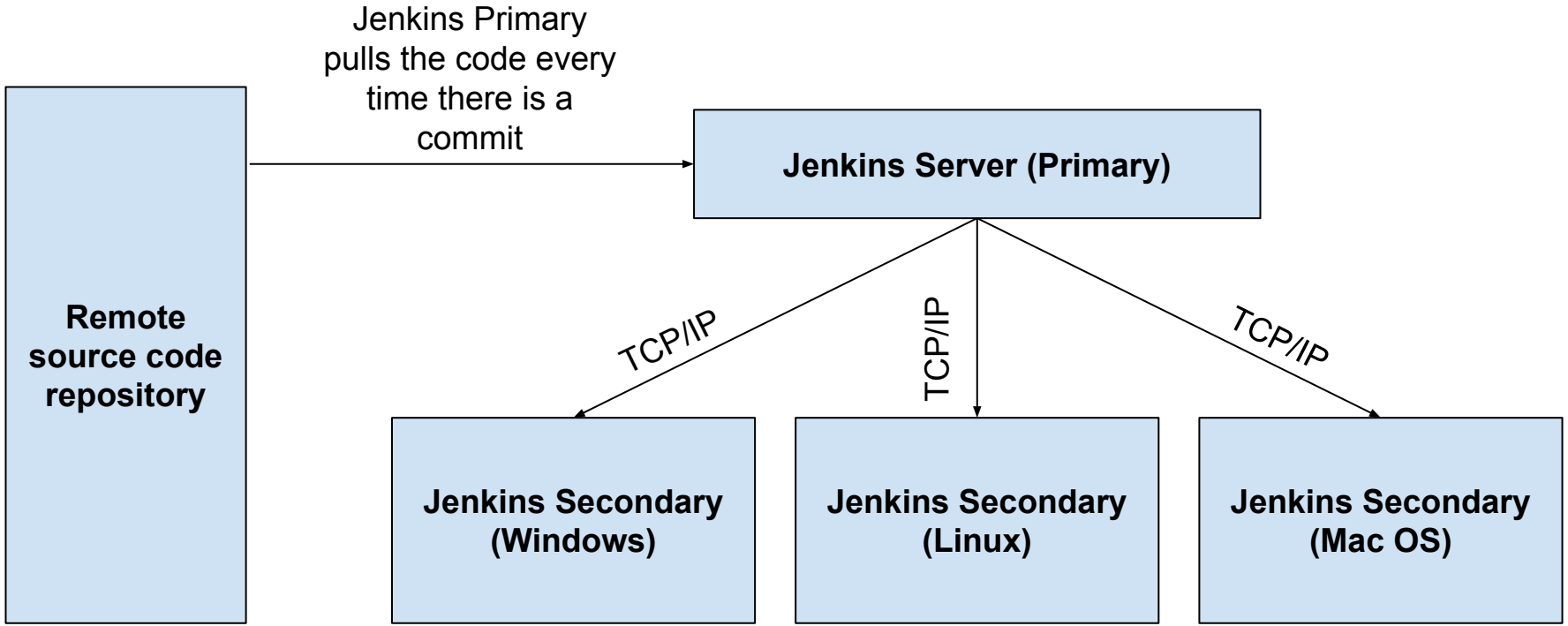
Approach 2 - HTCondor - Contd.

Limitations:

- Designed to leverage root access on compute nodes to run specific jobs as different users
- No documentation to use a single non-privileged user throughout the entire workflow/
- Only has built-in support for a few container runtimes (Not-Generic)

Approach 3 - Jenkins

- Open source automation server that helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous deliver.



Approach 3 - Jenkins - Contd.

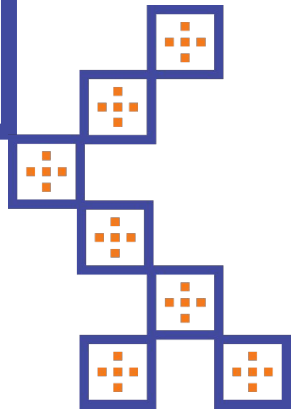
Limitations:

- Just an automation language, not an orchestrator
- Necessitates privileged access on HPC nodes

Design Constraints for the BigHPC Orchestrator

- Design for Userspace Runtime (No root access): The container engine must not demand root access on HPC Cluster nodes
- Design for Simplicity and Performance for BigHPC users
- Design for the compatibility with diverse HPC architectures
  - We couldn't use several open source solutions (Kubernetes, HTCondor, Jenkins) as target sites have incompatible specifications for such open source approaches.
- Design to provides persistence services for BigHPC workloads
  - HPC compute nodes are hidden from public network (in our case, orchestrator). Need a mechanism to reach HPC nodes from public networks

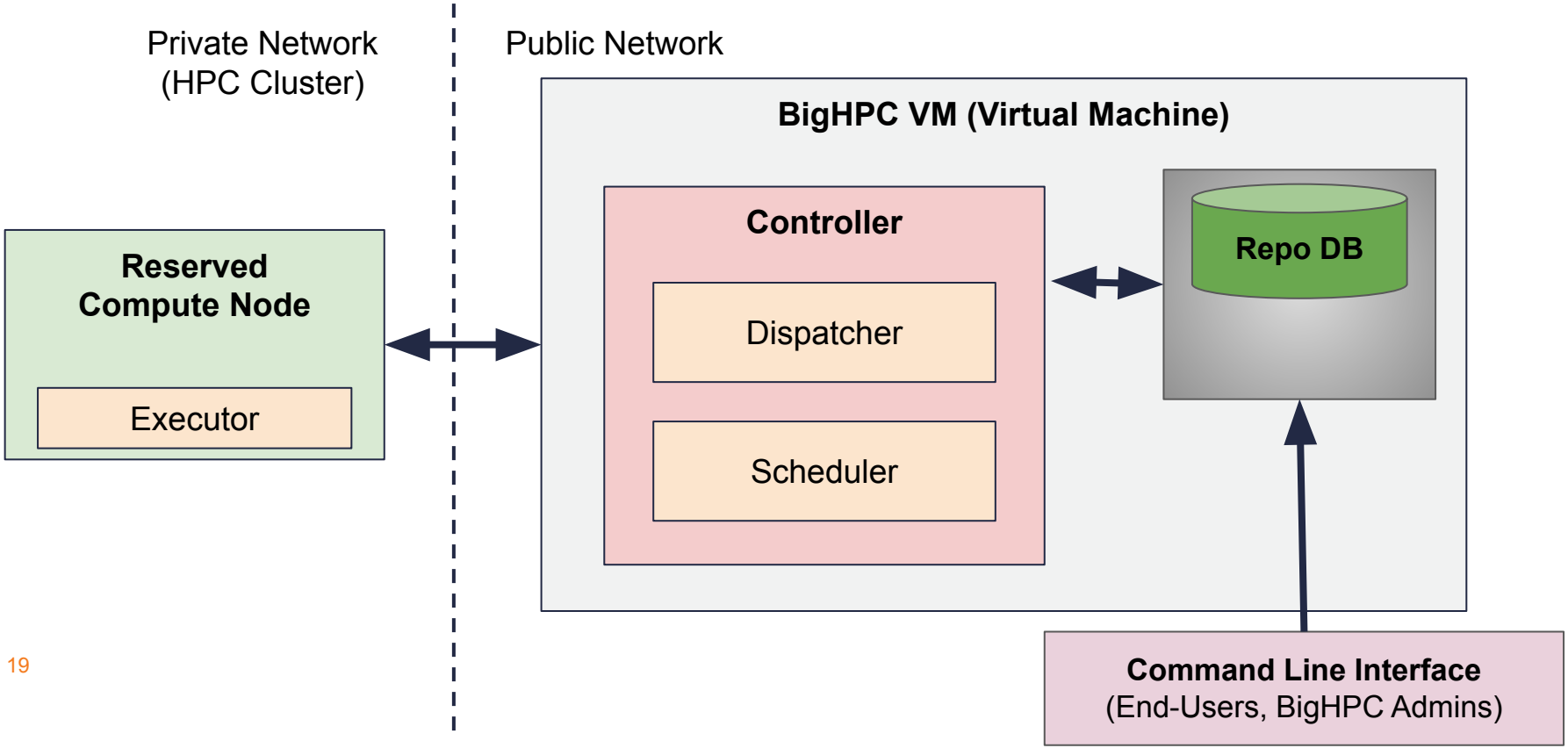
BigHPC Orchestrator



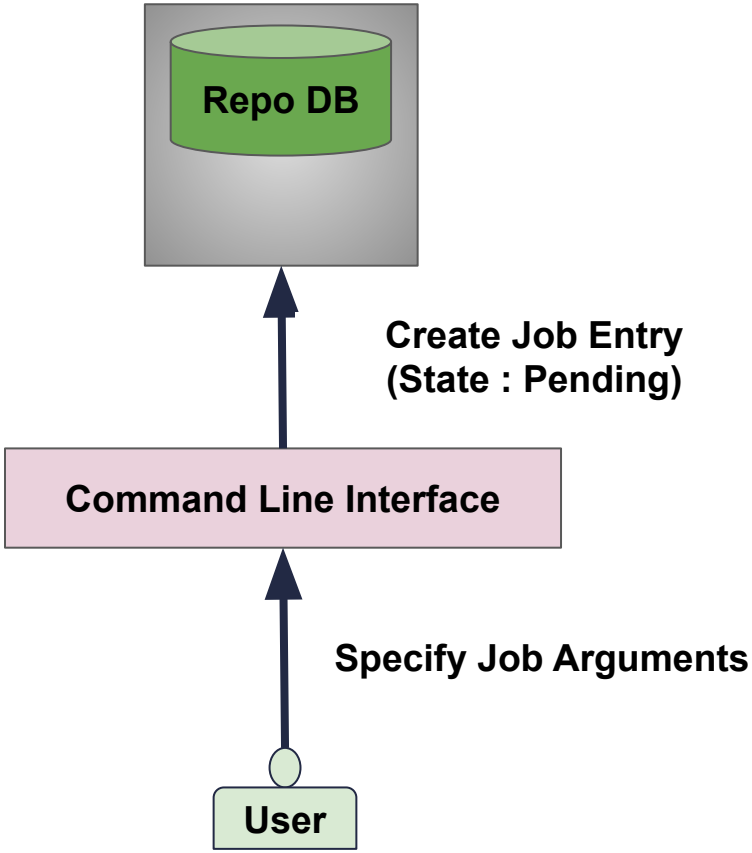
BigHPC Orchestrator - Agenda

- System Architecture
- Workload submission by users
- BigHPC Compute Node Reservation
- Compute Nodes Initialization
- Workload scheduling and dispatching

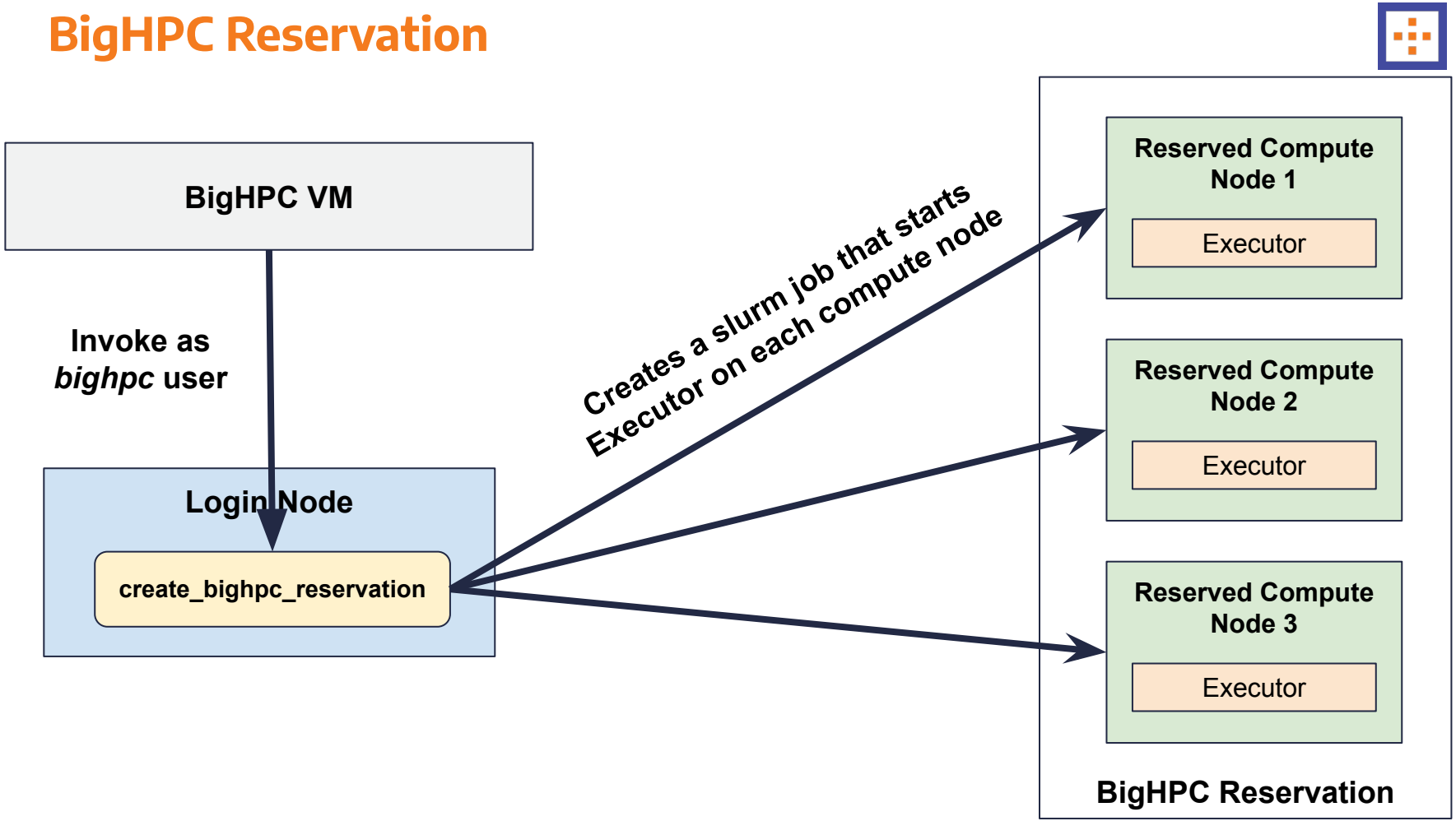
Orchestrator - Current Implementation Architecture



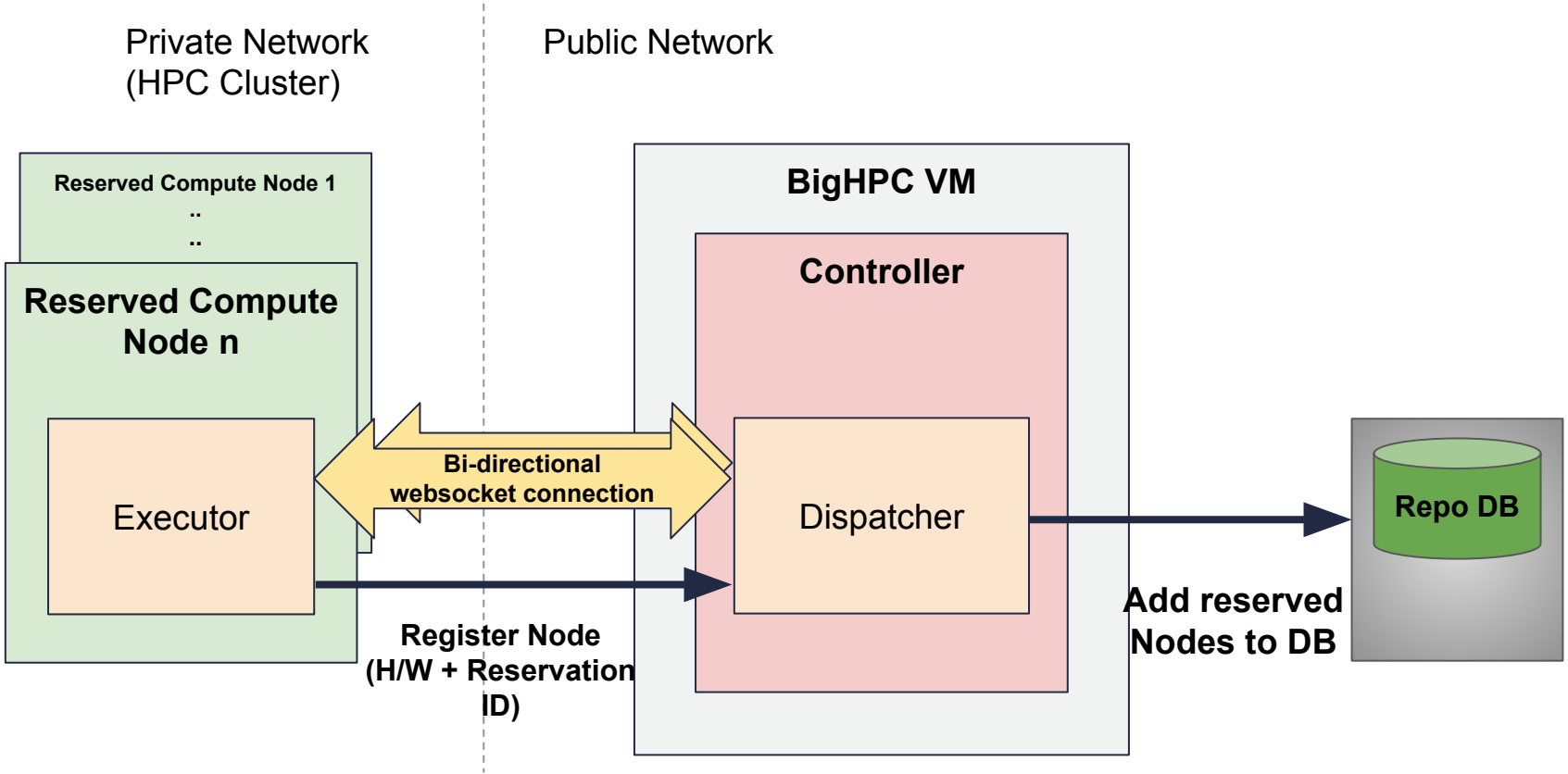
Workload Submission



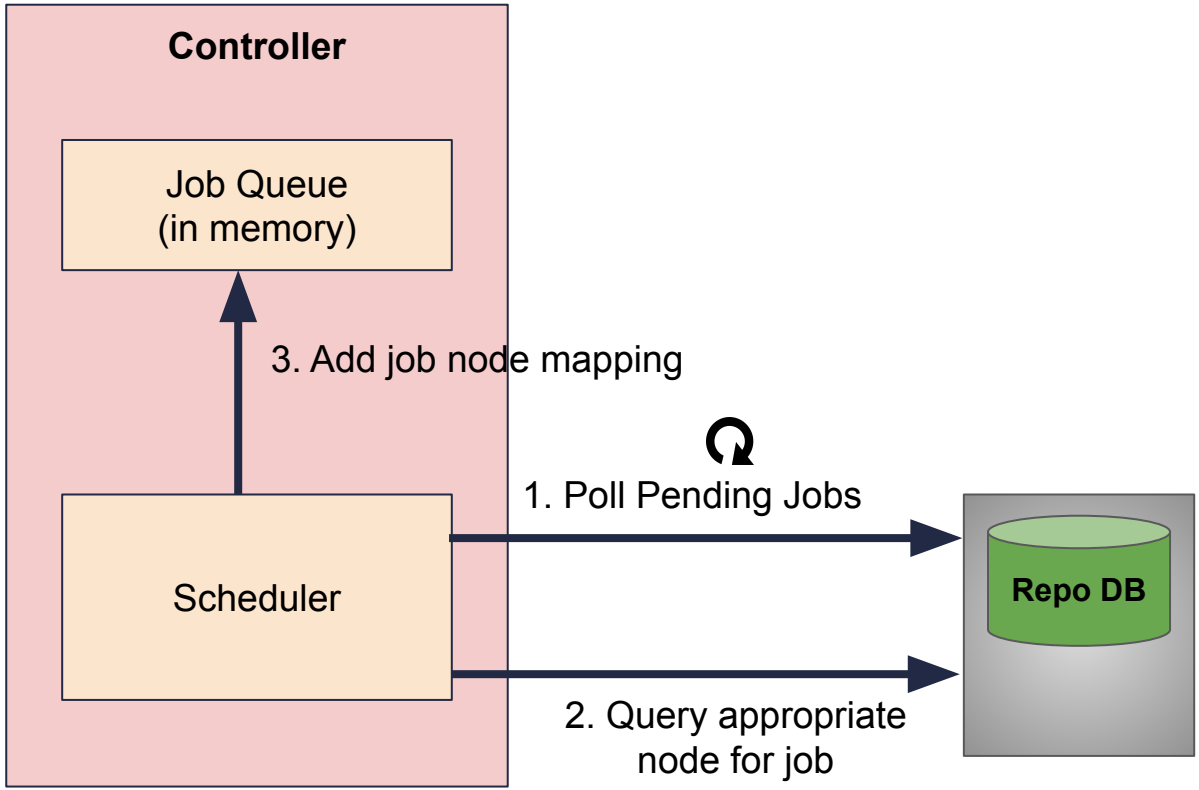
BigHPC Reservation



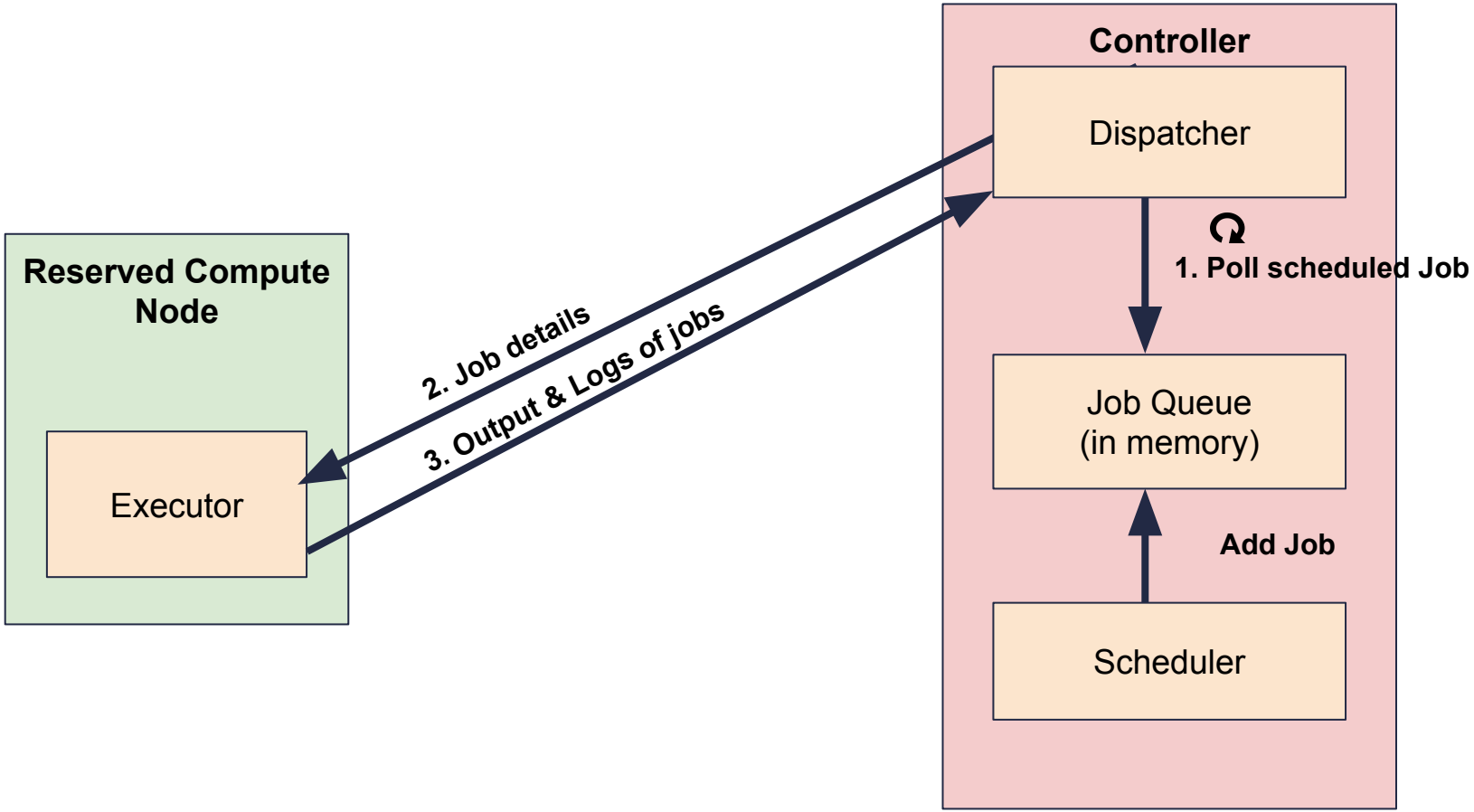
Node Initialization: Executor Startup & Node Registration



Scheduling Jobs



Dispatching Jobs



Orchestrator - Summary

- Overcomes limitations described earlier (no root access) with Controller and Executor
- Enables easy integration with existing HPC workload managers (e.g., slurm, PBS, LSF etc) via BigHPC reservations
- Overcomes networking limitation via persistent websocket connection - initiated by compute node

Orchestrator - Integration and Next Steps

- Integration with multiple components is our next step which has a few different facets
  - Database integration - all services will use the same database server
  - Executor will run persistent services inside the allocation for the life of the allocation
    - Storage Control Plane
    - Monitoring Framework
  - Executor will share BigHPC Job IDs with the other components as keys to use the same database.

- New Scheduling Algorithms will be implemented

Final Thoughts

Big Data workloads are becoming more common on HPC systems (especially with rise of Machine Learning)

- HPC and Big Data do not readily coexist -> BigHPC addresses this issue through 3 components
- Virtualization Manager provides portable AND optimized containers for diverse workloads on diverse HPC hardware and software environments

We provide performance and portability in BigHPC for users using the methods described here

- Use container technologies to provide software-level portability with very low overheads
- Provide containers for each system to match the specialized hardware to provide hardware-level portability and performance

We validated the performance of BigHPC's container solutions for HPC applications

- No significant overheads compared to bare metal runs in terms of latency and memory.
- No known general security issues in HPC environments

With this design we have shown how we can use these portable and performant HPC containers within typical HPC clusters from a central BigHPC scheduler.

Stickers

UTAustin Portugal

FCT Fundação para a Ciência e a Tecnologia

TEXAS The University of Texas at Austin

TECHNOLOGY INNOVATION AND ENTREPRENEURSHIP

MEDICAL PHYSICS

ADVANCED COMPUTING

SPACE-EARTH INTERACTIONS

NANOTECHNOLOGIES