Sunfish Management of Lustre On-Demand FAM-based Filesystem

Michael Aguilar, PI, Senior Computer Scientist,
HPC Research and Development, Sandia National Labs

Catherine Appleby, Andreas Dilger, Matthew Curry, Shyamali Mukherjee,
Erica Armijo, Christian Pinto, Phil Cayton, Russ Herrell, Michele Gazzetti

Sandia National Labs, Whamcloud, OpenFabrics Alliance, CXL Consortium

Lustre User Group, 2024

Wednesday, May 8, 2024

Lubbock, Texas

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2024-05363C

Sunfish Management of Lustre On-Demand FAM-based Filesystem

1. Goals and Motivation for Lustre-on-Demand
2. Composable Disaggregrated Infrastructure (CDI)
3. Sunfish Composability Manager for HPC systems
4. Combining Lustre-on-Demand with Sunfish to create a versatile and dynamic burst buffer
5. Further links and Q&A

Goals and Motivation for Lustre-on-Demand

- On-Demand Community-Based Lustre Burst Buffer
- Localized parallel burst-buffer file IO
- Optimized CDI Burst Buffer integration
- Virtual Cluster Manager integration
- New capabilities that we add into the Lustre tree will allow new implementation ideas----'If we build it, they will come'

Why is this implementation potentially better than other implementations?

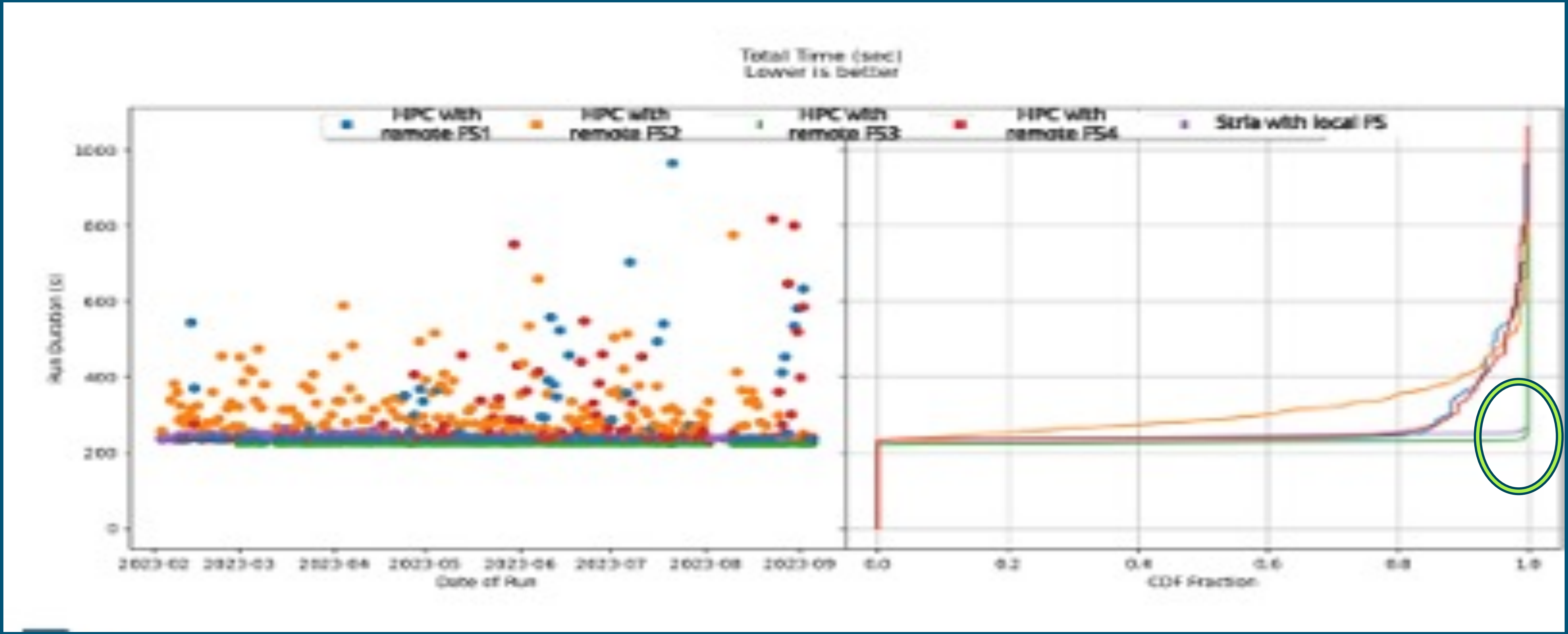
- Open Source version of Lustre-on-Demand, in the community tree
- New Lustre OSD that can take better advantage of the ephemeral nature of our proposed burst buffer, especially when using Fabric Attached Memory
- Centralized implementation of Reinforcement Learning to impact resource allocation of Fabric Attached Memory can be better integrated with Workload Managers (eg. Flux) and Container Deployment Services (eg. Kubernetes)
- Improvements in dynamic deployment of software-defined nodes can help mitigate current HPC and Cloud IO issues
- Integration with Compute Express Link and RDMA, for new HPC architectures

Overview of a Lustre-on-Demand Burst Buffer implementation

- Dynamic start-up that will bring up Management, Metadata, and Storage, in order
- Support for dynamic addition of Storage OSTs, when requested
- Allow for varying quantities of MDTs and OSTs, as requested, upon start-up
- Implements RAM disk OSDs
- Capable of Staged In/Staged Out operations, when requested
- Capable of shared remote filesystem ‘local caching’, if requested

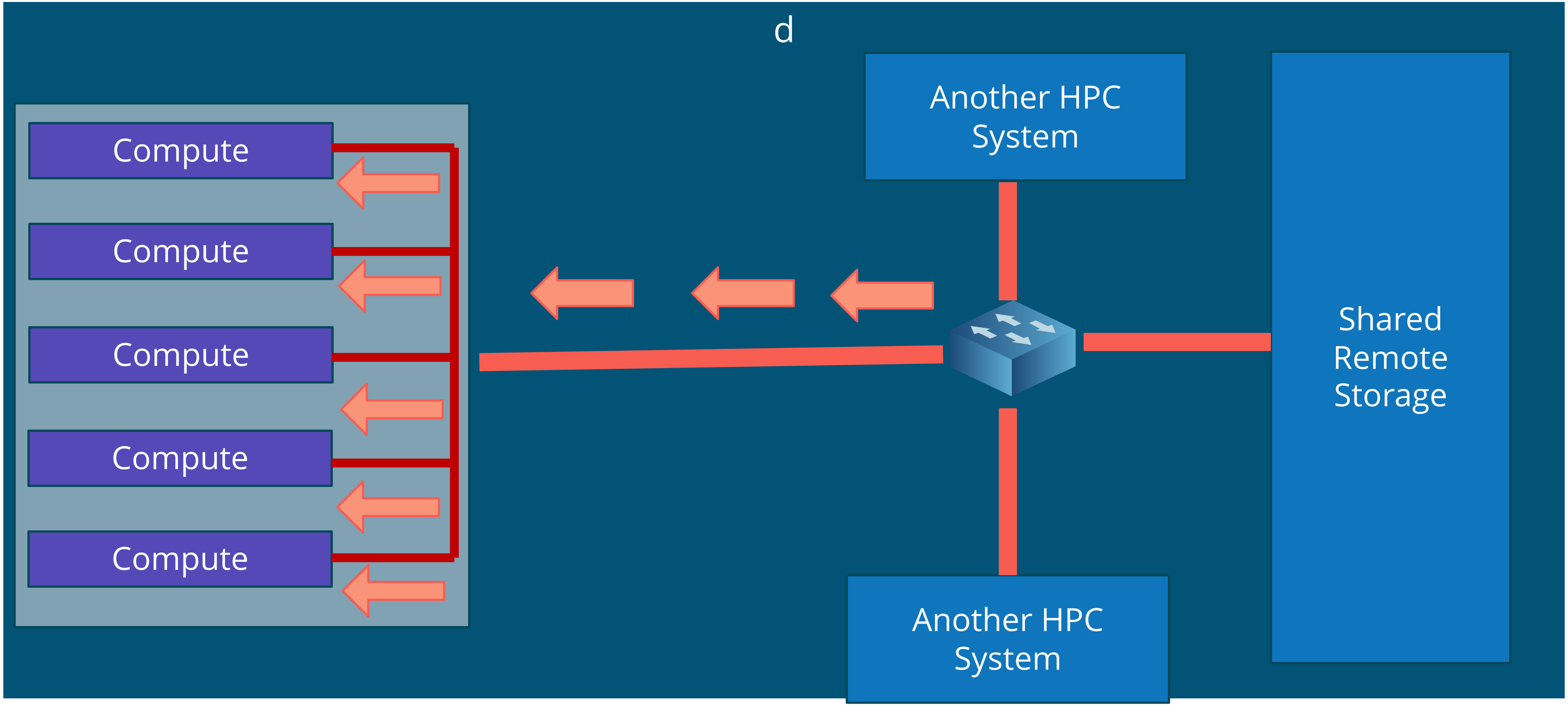
Overview of a Lustre-on-Demand Burst Buffer implementation

Burst Buffers give fast and consistent IO performance



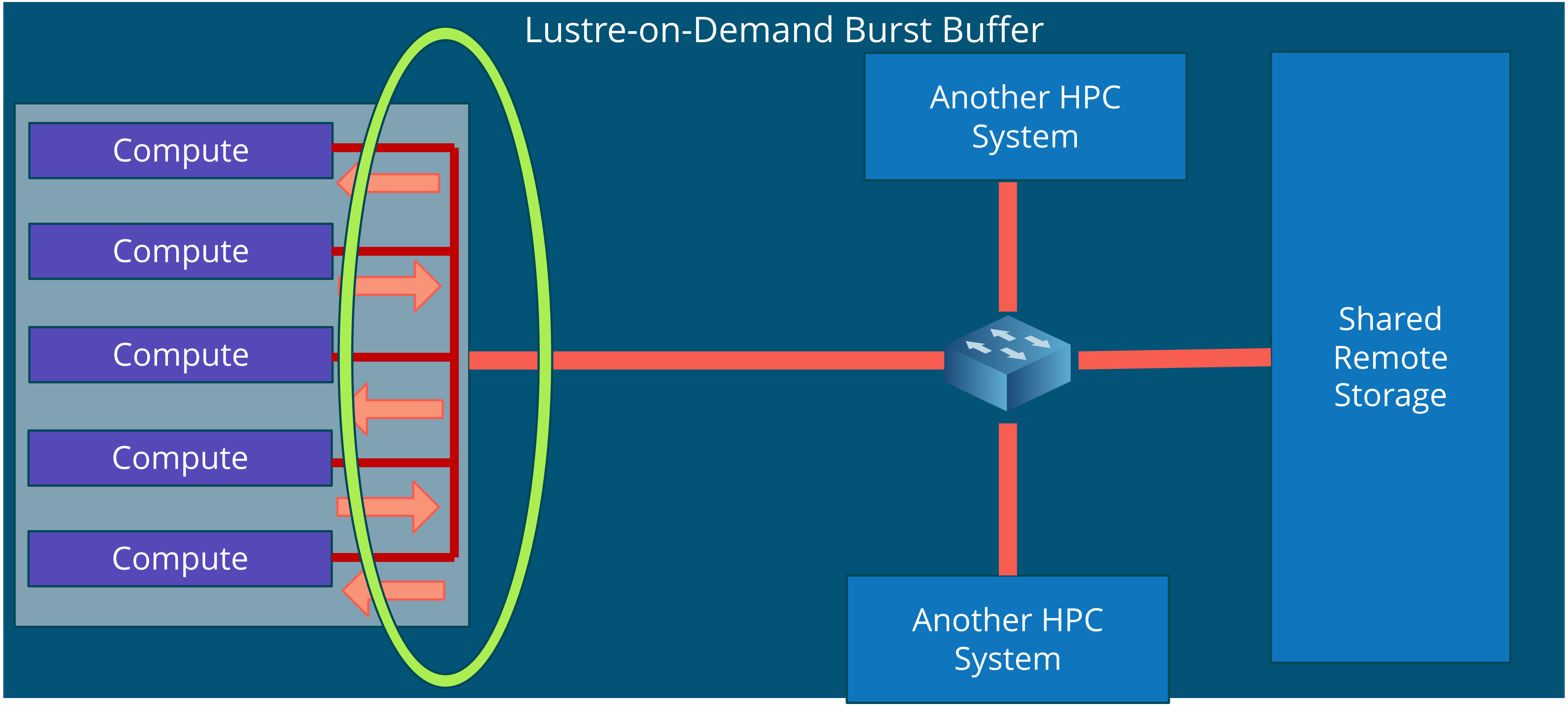
Goals for a Lustre-on-Demand Burst Buffer

We copy in the data from the remote storage.



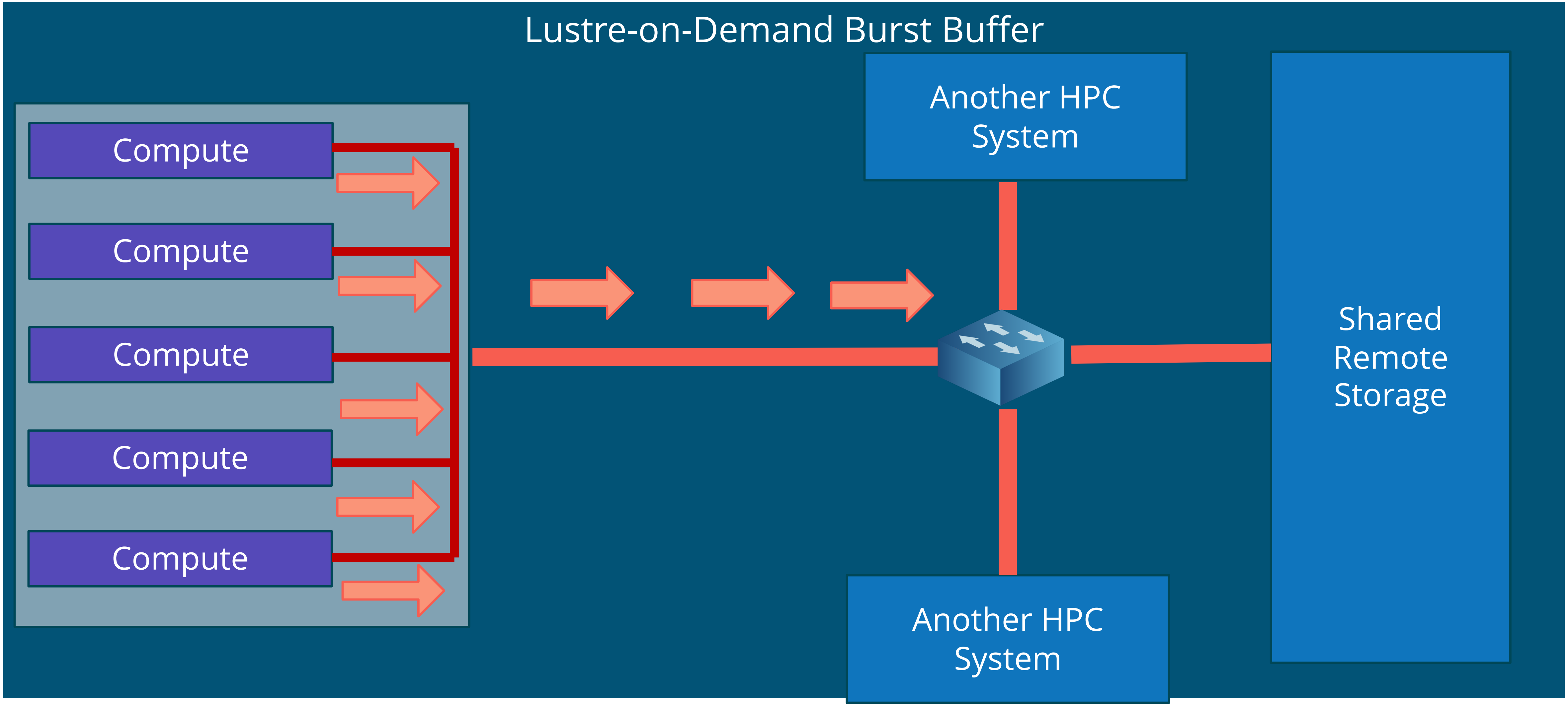
Goals for a Lustre-on-Demand Burst Buffer

Localized and personal parallel filesystem----We can expect better performance because the filesystem IO has fewer hops and reduced congestion to deal with



Goals for a Lustre-on-Demand Burst Buffer

We copy the data out to the remote storage.



Goals for a Lustre-on-Demand Burst Buffer

RAM-based OSD----Byte addressable storage means more efficient IO transactions

- Byte-addressable, not Block IO
  - Skipping Block IO aggregation helps with small file IO, directory operations, inode sizes, etc.
    - (e. Kernel requesting file metadata from the MDS)
  - IO wait queues
- A dedicated RAM OSD is ephemeral
  - We don’t need to provide support for data recovery
  - We can be more efficient in terms of object allocation, data structure sizing

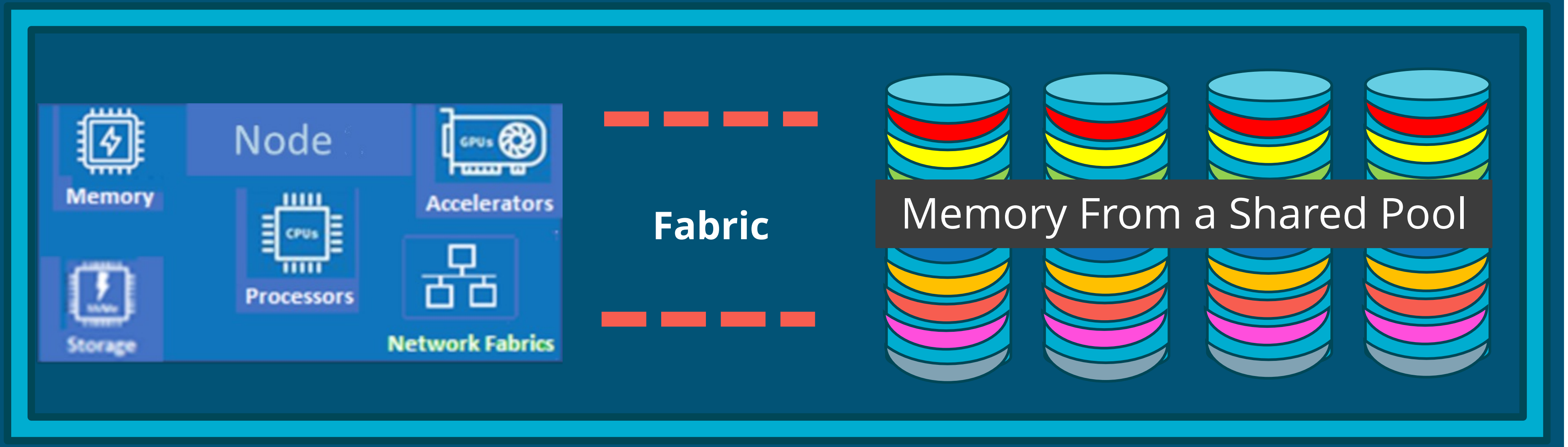
Goals for a Lustre-on-Demand Burst Buffer

Ready to take advantage of Composable Disaggregated Infrastructure

Burst buffer memory from our remote memory pools alleviates stranded memory resources

Fabric Attached Memory

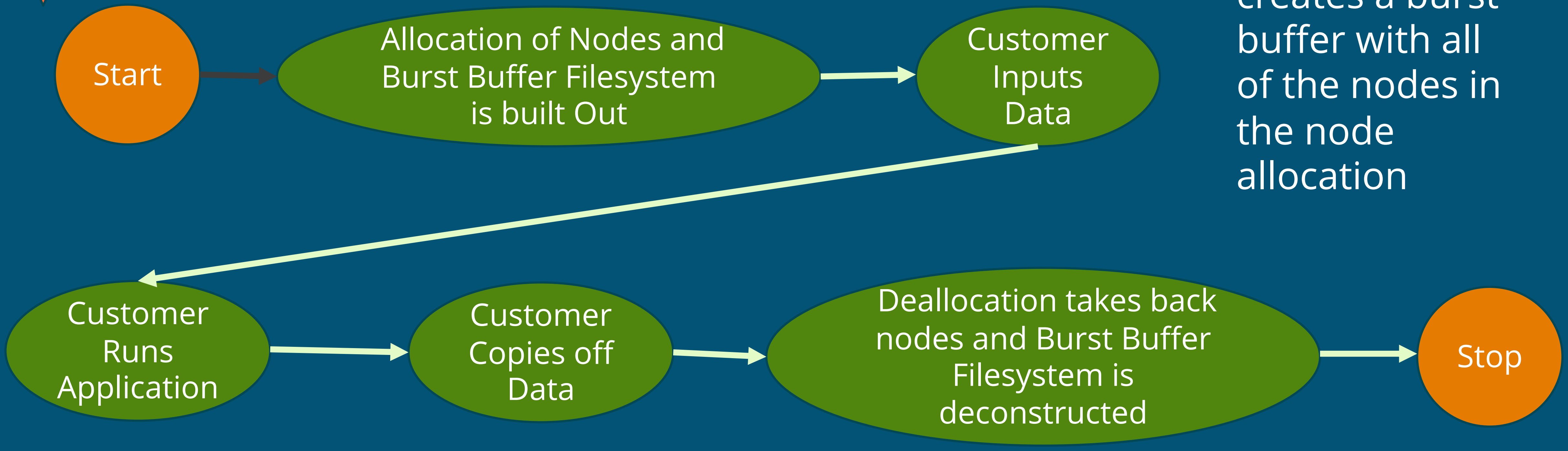
Burst buffer filesystem is included as a key request of a software-defined node, during a Virtual Cluster Manager allocation



Goals for a Lustre-on-Demand Burst Buffer

Automatic Parallel Burst Buffer

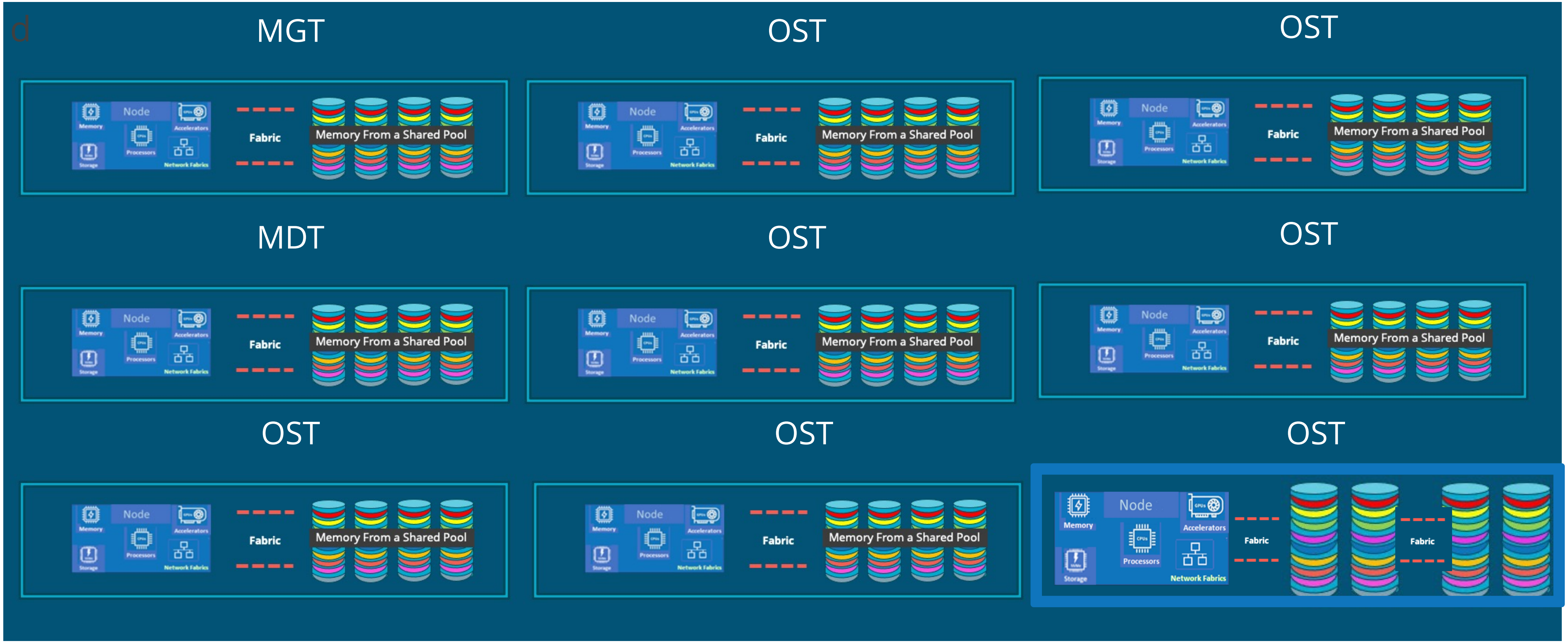
Seamless for Customers



Flux or Kubernetes creates a burst buffer with all of the nodes in the node allocation

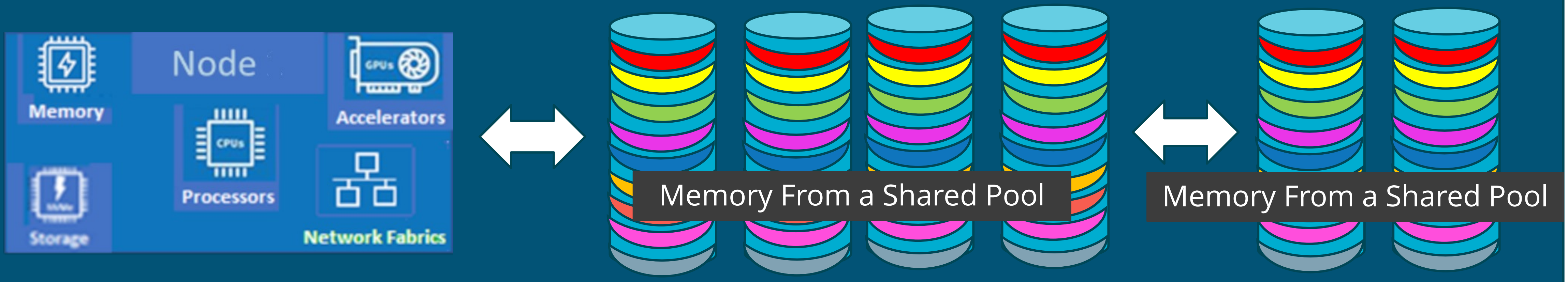
Composable Disaggregrated Infrastructure (CDI)

Dynamically Adding in more OSTs



Composable Disaggregrated Infrastructure (CDI)

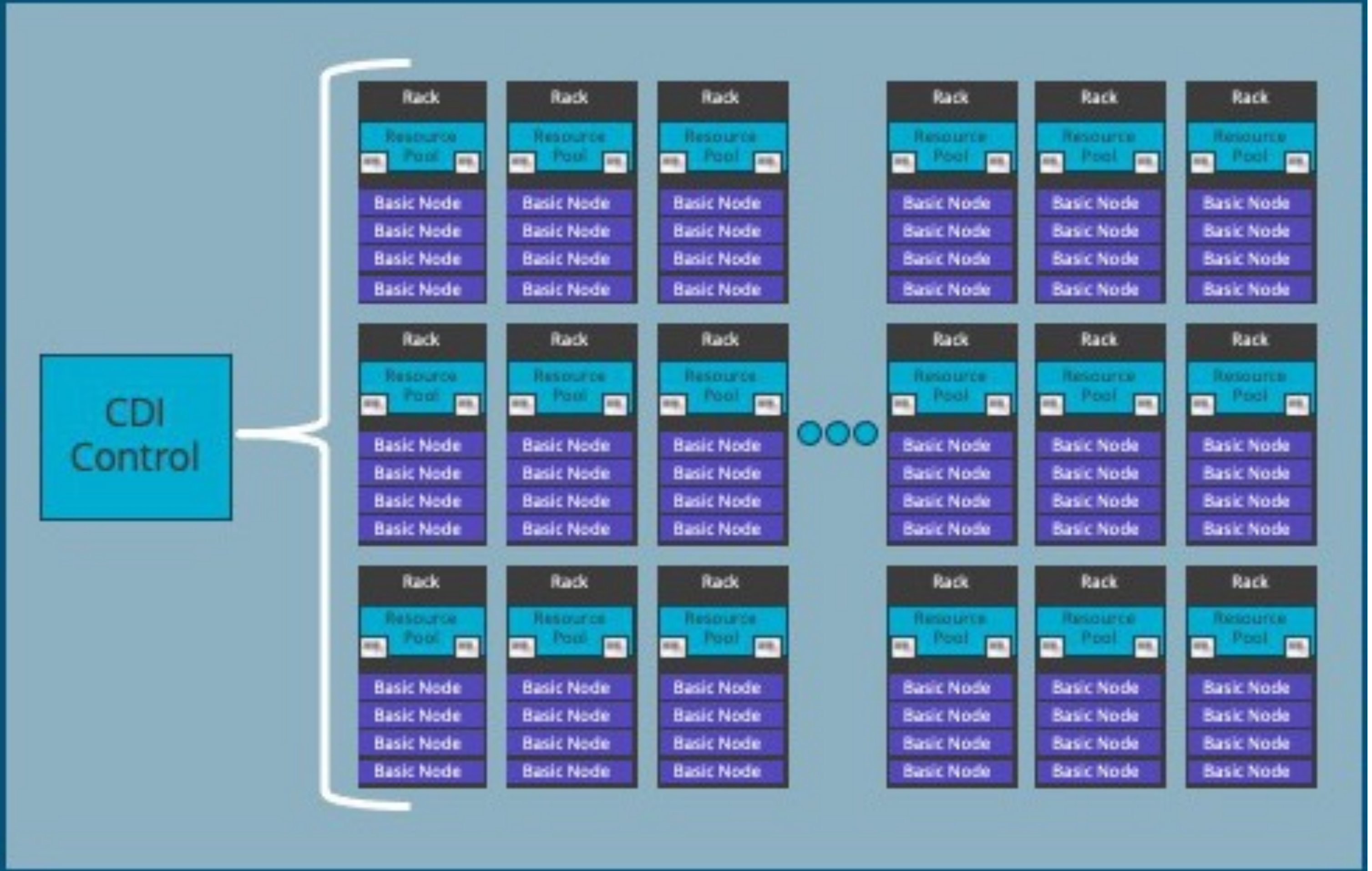
Dynamically attaching more Fabric Attached Memory, if needed, for caching



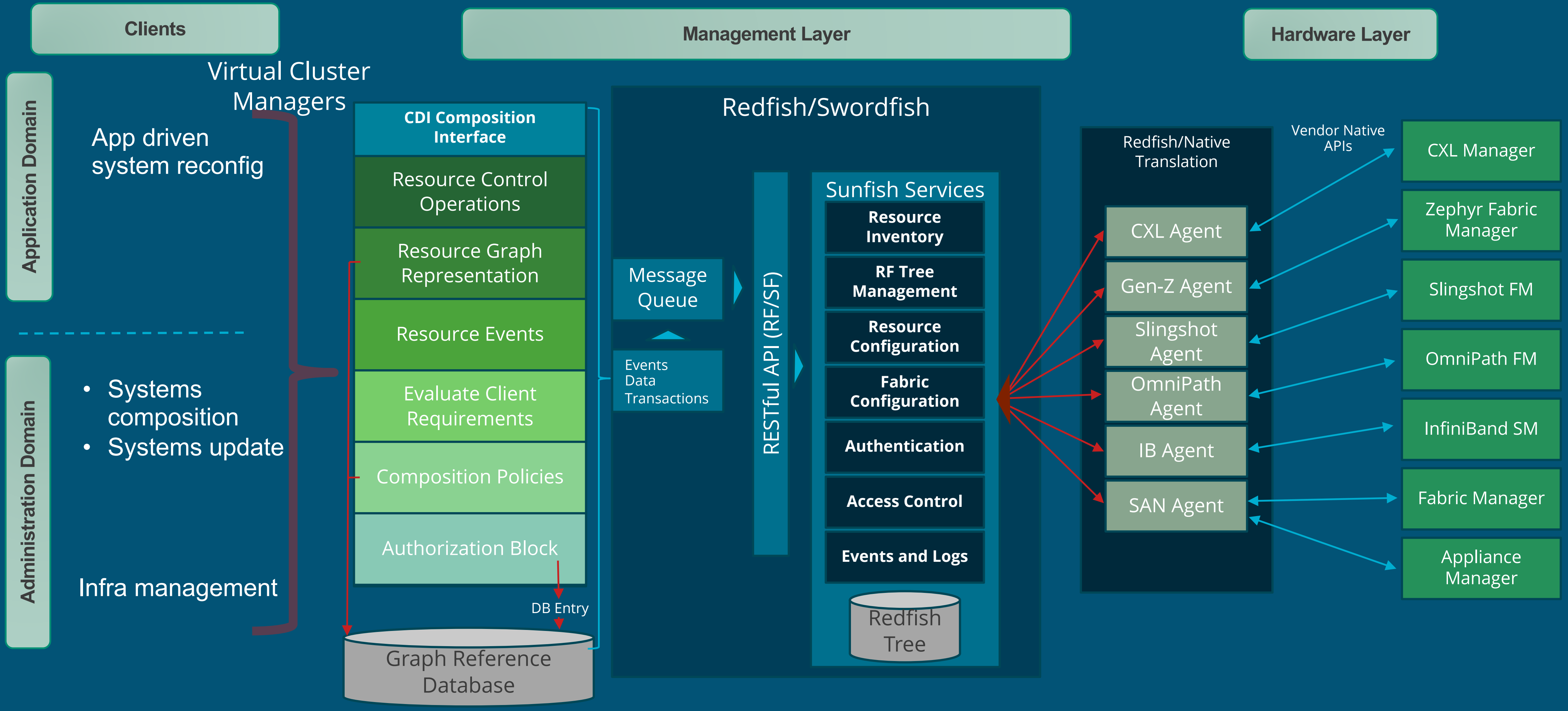
Design Considerations for a Composability Manager on a Large-Scale HPC System

CDI Control

- Need to keep track of a huge number of concurrent resources
- Need to keep management and query communications down to a reasonable quantity
- Need to be able to execute timely changes to the HPC system as those changes are requested



Introducing Sunfish



Introducing Sunfish

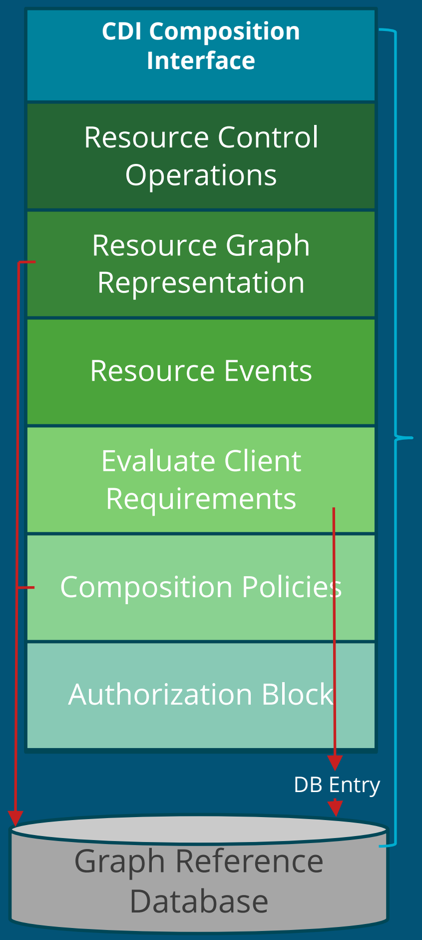
OFA
Sunfish

OpenFabrics Alliance
Oak Ridge National Laboratory
intel
Red Hat
Cornelis Networks
SNIA
Sandia National Laboratories
IBM
H3
IntelliProp
DMTF
Lawrence Livermore National Laboratory
Hewlett Packard Enterprise
Fsas Technologies
CXL Compute Express Link
Dell Technologies
scalemem

The Sunfish Composability Management Framework

For our HPC System, the Composability Manager reduces transactions to Sunfish Core

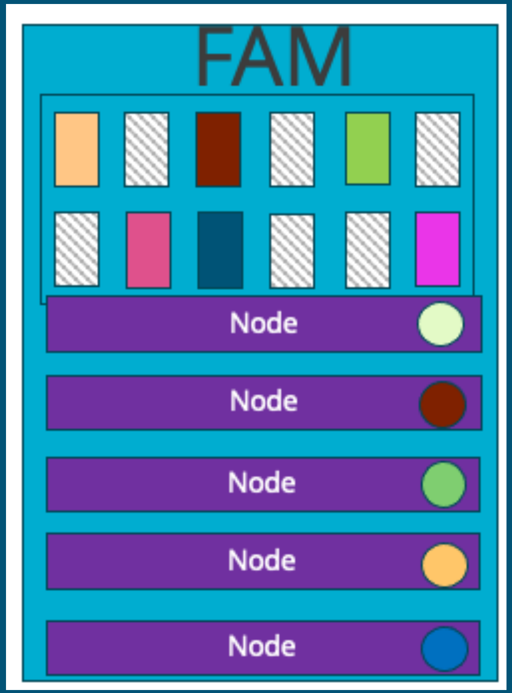
- The Composability Manager provides a platform for decision making, for picking the FAM, for each node
- Clients can check availability of resources, before scheduling our nodes
- Verification of the success of our burst buffer creation is reported back to clients
- If we don't succeed in constructing our burst buffers, then we can try to allocate new software-defined nodes and we record the failed allocation
- We can lock out FAM for possible stage-in/stage-out, for our burst buffers, even after the application run



Sunfish Hardware Agents

4-Dimensional Software Defined Node Allocation

- In this example, the FAM pools are located at the top of each rack.
- The Resource Pools, for available orchestration, are going to be grouped together, in certain physical locations, over heterogeneous fabrics
    - Orchestration could look a little bit like Tetris, across the Resource Pools



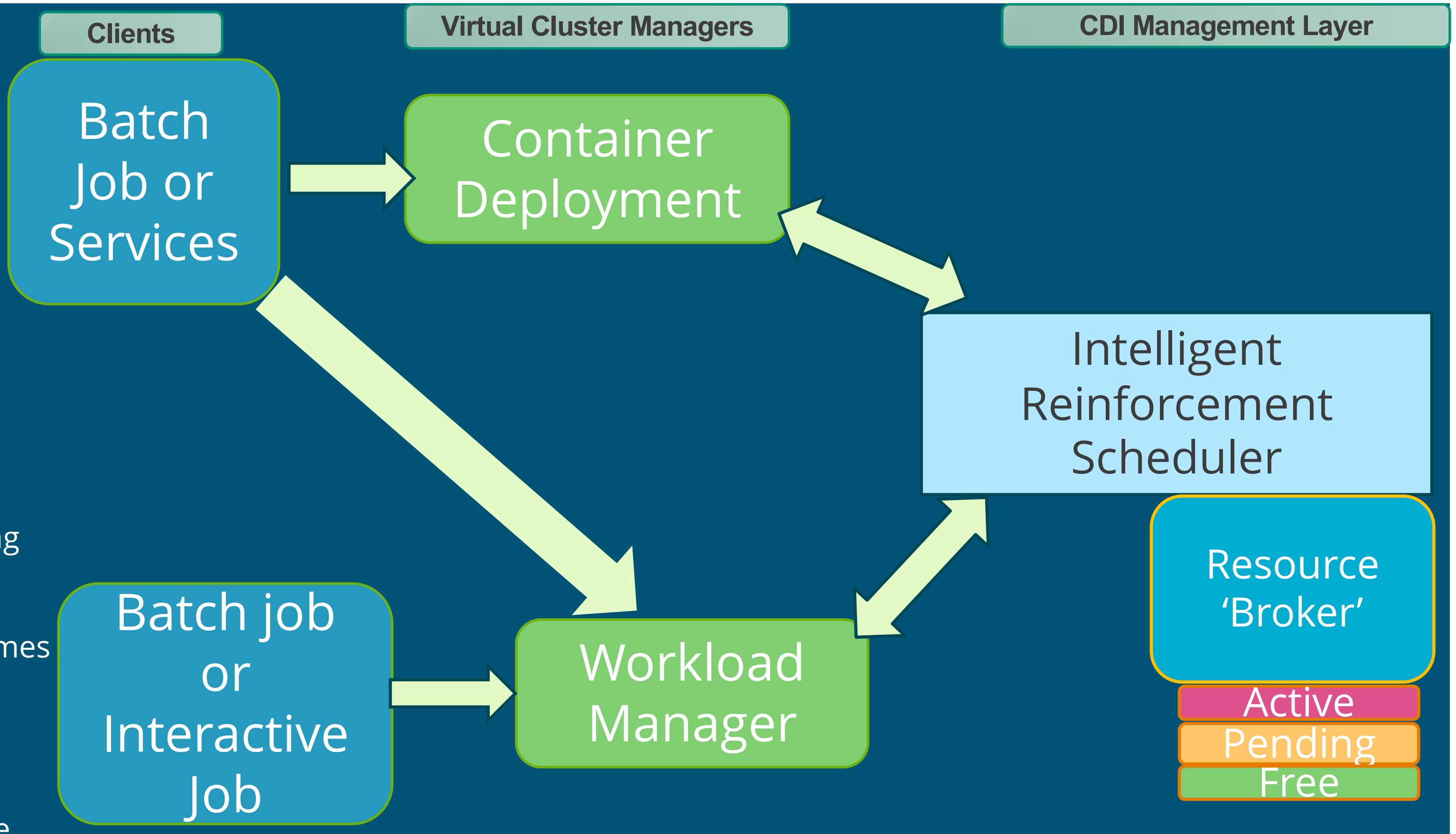
How Machine Learning can help us allocate CDI Resources and Algorithm Design

Reinforcement Learning

Heuristics
- Can be fair, but often aren't the most efficient

Optimization Algorithms
- Must be highly tailored to specific machines

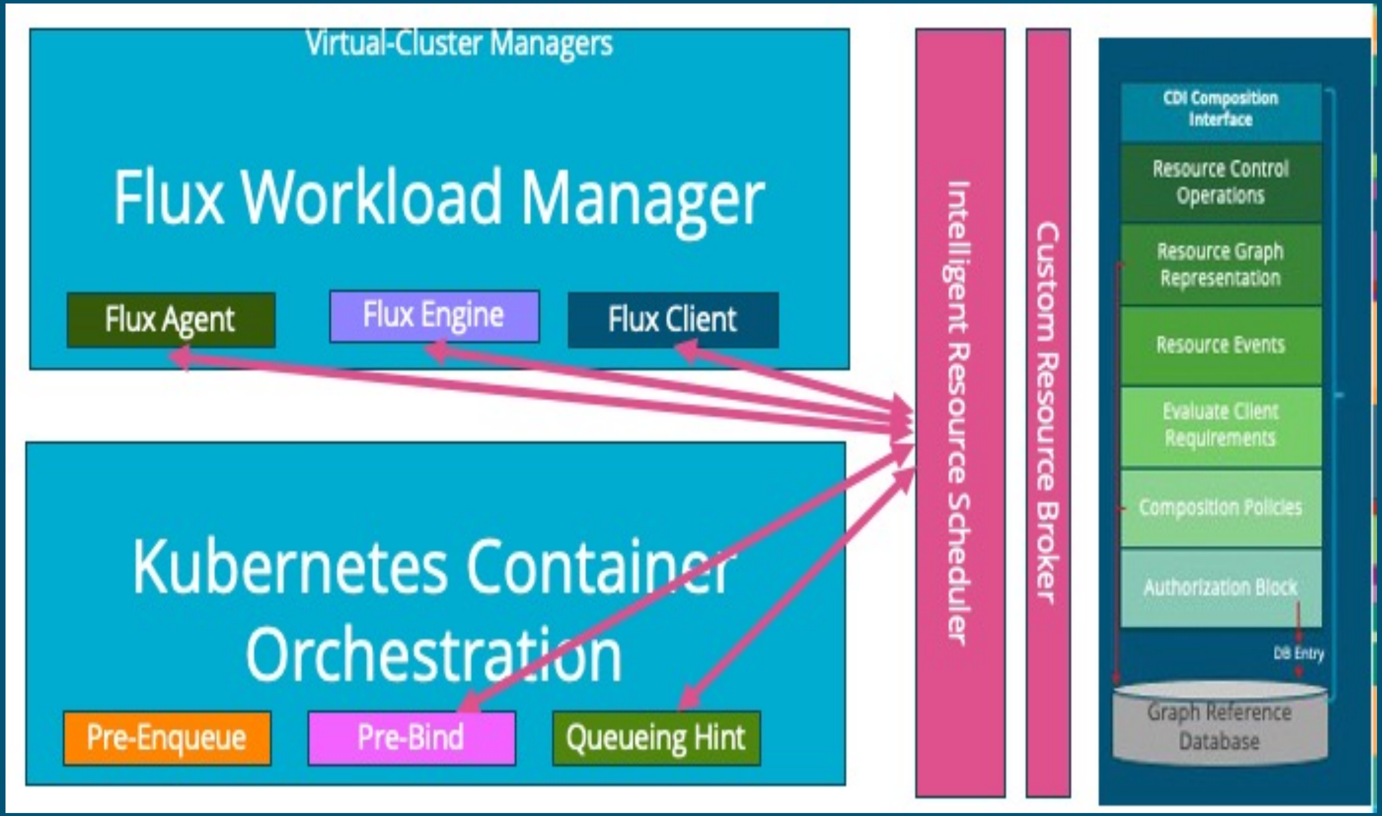
Reinforcement Learning
1. Customized rewards function
   1. Prioritize fairness
   2. Penalize undesirable scheduling
2. Machine agnostic
   1. Adapts to changing resources
   2. Adapts to different traffic volumes
   3. Learns a better algorithm over time
3. Potential cons
   1. Prone to job starvation
   2. May need lots of compute/time



How Machine Learning can help us allocate CDI Resources and Algorithm Design

Integration with Flux and Kubernetes

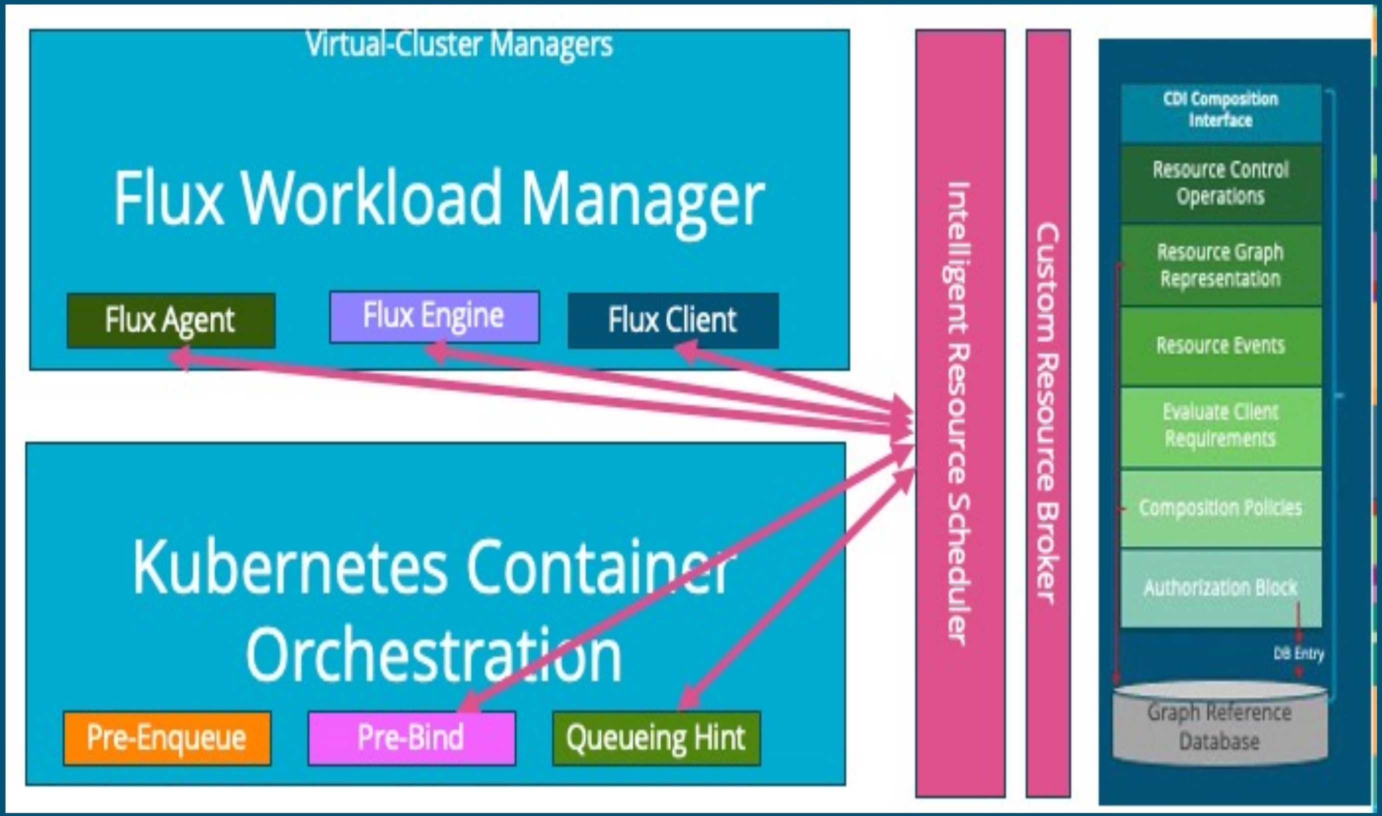
- Flux
  - Flux Engine----API to implement workflows and workflow scripts
  - Flux Client CLI----Gives manipulation control of the Flux Engine
  - Flux Agents----Process Action Agents invoke command-line programs and scripts
- Kubernetes
  - Queueing hint----An event occurs that makes a Pod available for scheduling
  - Pre-Filter----What conditions are needed for the Pod operation?
  - Pre-Bind----What does Sunfish need to do before we prep out the software-defined nodes for the Pod?
  - Bind-----Schedule the Pod for the nodes



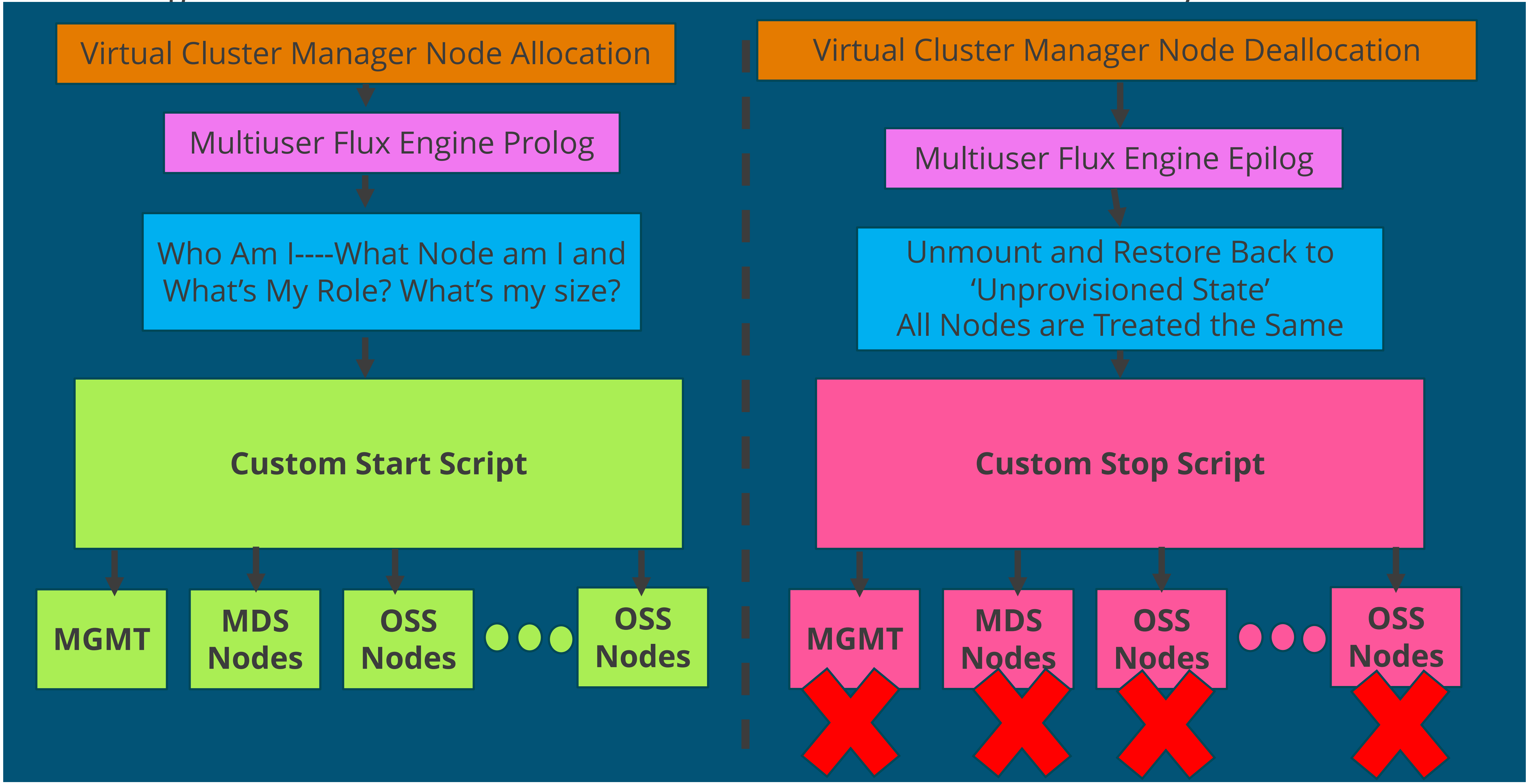
How Machine Learning can help us allocate CDI Resources and Algorithm Design

Integration with Flux and Kubernetes

- Current Workload and Kubernetes Schedulers implement back-flow strategies
- Current Workload and Kubernetes Schedulers assume node limitations, inside the box
- Portions of the Resource Pools will be available, at specific portions of time

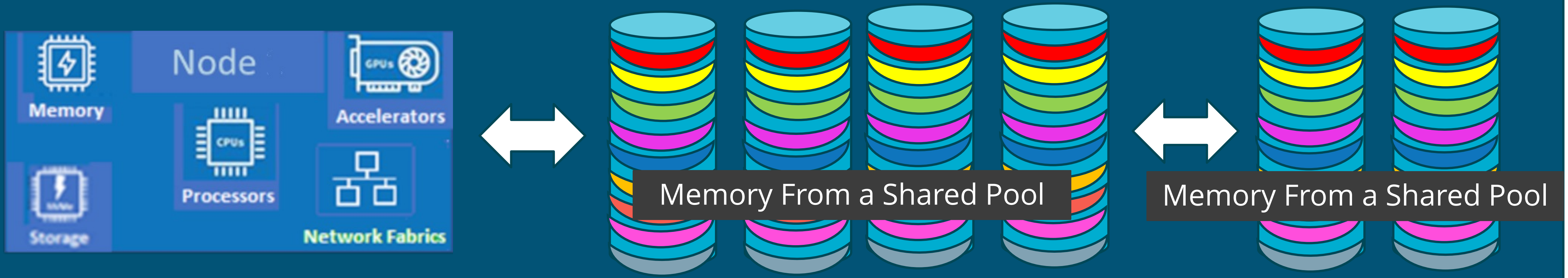


Combining Lustre-on-Demand with Sunfish to create a versatile and Dynamic Burst Buffer



Composable Disaggregated Infrastructure (CDI)

Dynamically attaching more Fabric Attached Memory, if needed, for caching, we will need to learn what the limits are available for memory additions



Further links and information

- Lustre
- GitHub - OpenFabrics/sunfish_docs: Documentation for the Sunfish Project
- opensfs.org/wp-content/uploads/Fast-IO-El-Capitan-Rabbits.revised.pdf
- Scheduling Framework | Kubernetes
- Command Line Interface | Flux Docs
- GitHub - hpc/mpifileutils: File utilities designed for scalability and performance.

Acknowledgements and Questions

- [Whamcloud – Enterprise-grade technical support for Lustre]
- Lustre Working Group - OpenSFS Wiki
- OpenFabrics Alliance – Innovation in High Speed Fabrics
- Home | DMTF
- SNIA | Experts on Data
- About CXL® - Compute Express Link
- https://www.llnl.gov/
- Sandia National Laboratories