SAND2024-01831C

Sandia National Laboratories

Breaking the Energy, Cooling, and Resource Ceiling for Future Supercomputer Architectures Through the Use of Composable Disaggregated Infrastructure

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

Catherine Appleby, Christian Pinto,

Phil Cayton, Russ Herrell, Richelle Ahlvers, Alex Lovell-Troy

New Mexico Society of Professional Engineers, E-Week Conference

Friday, February 23, 2024

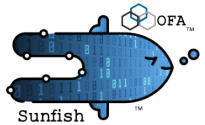
Albuquerque, New Mexico

U.S. DEPARTMENT OF ENERGY | NNSA

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.

SAND2023-11722C

Breaking the Energy, Cooling, and Resource Ceiling for Future Supercomputer Architectures Through the Use of Composable Disaggregated Infrastructure

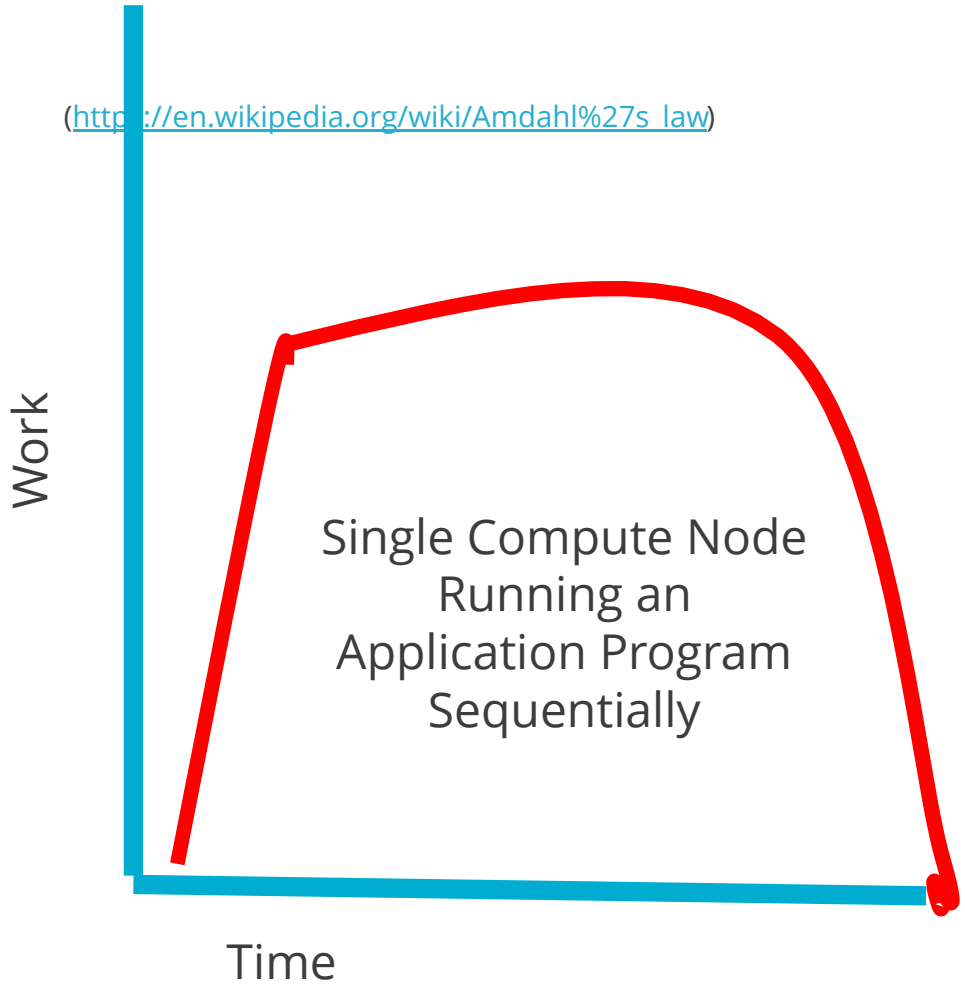
1. Quick overview of a Beowulf HPC Architecture, the majority of the HPC Systems
2. The 'Big Drawback' to current Beowulf HPC Architectures
3. Composable Disaggregated Infrastructure (CDI)---Software Derived Hardware Architectures
4. Design Considerations for a Composability Manager on a Large-Scale HPC System
5. Introducing Sunfish 

6. Sunfish Core Services
7. Sunfish Hardware Agents
8. The Sunfish Composability Management Framework
9. How Machine Learning can help us allocate CDI Resources and Algorithm Design
10. Acknowledgements and Questions

Quick overview of a Beowulf HPC Architecture, the majority of the HPC Systems

Large-Scale Parallel Execution of Compute Application Computational Units Saves Time

(http://en.wikipedia.org/wiki/Amdahl%27s_law)



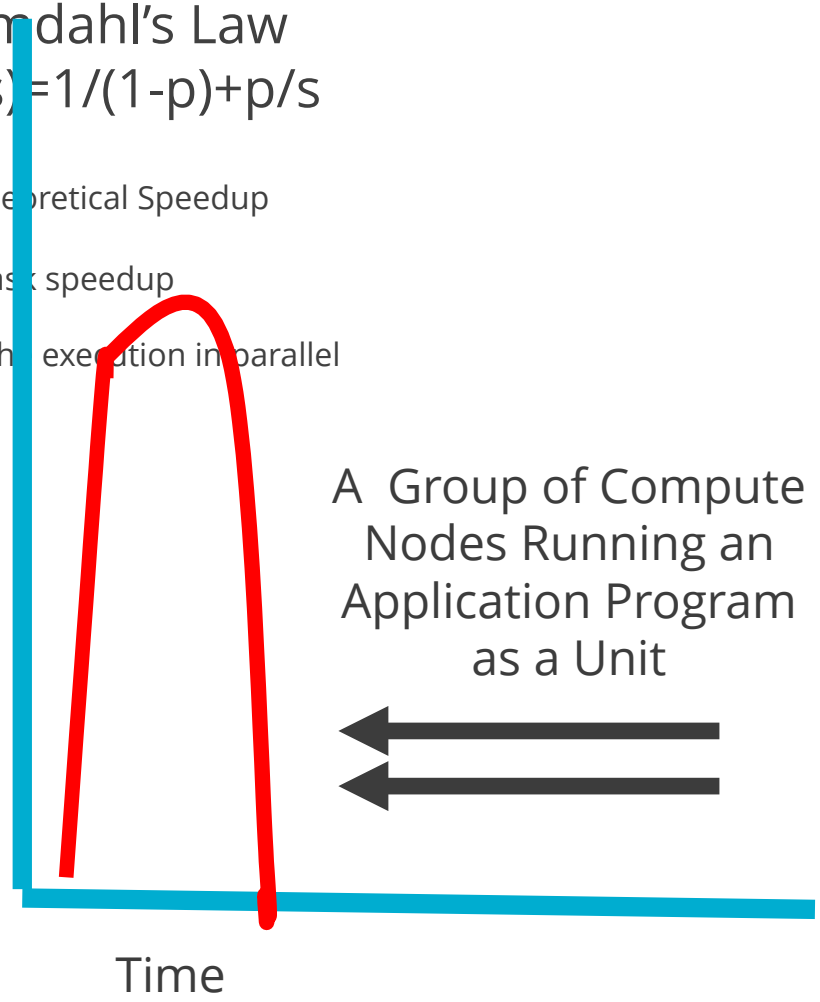
Amdahl's Law

$S_{latency}(s) = 1/(1-p)+p/s$

$S_{latency}(s)$ is the theoretical Speedup

s is the partial task speedup

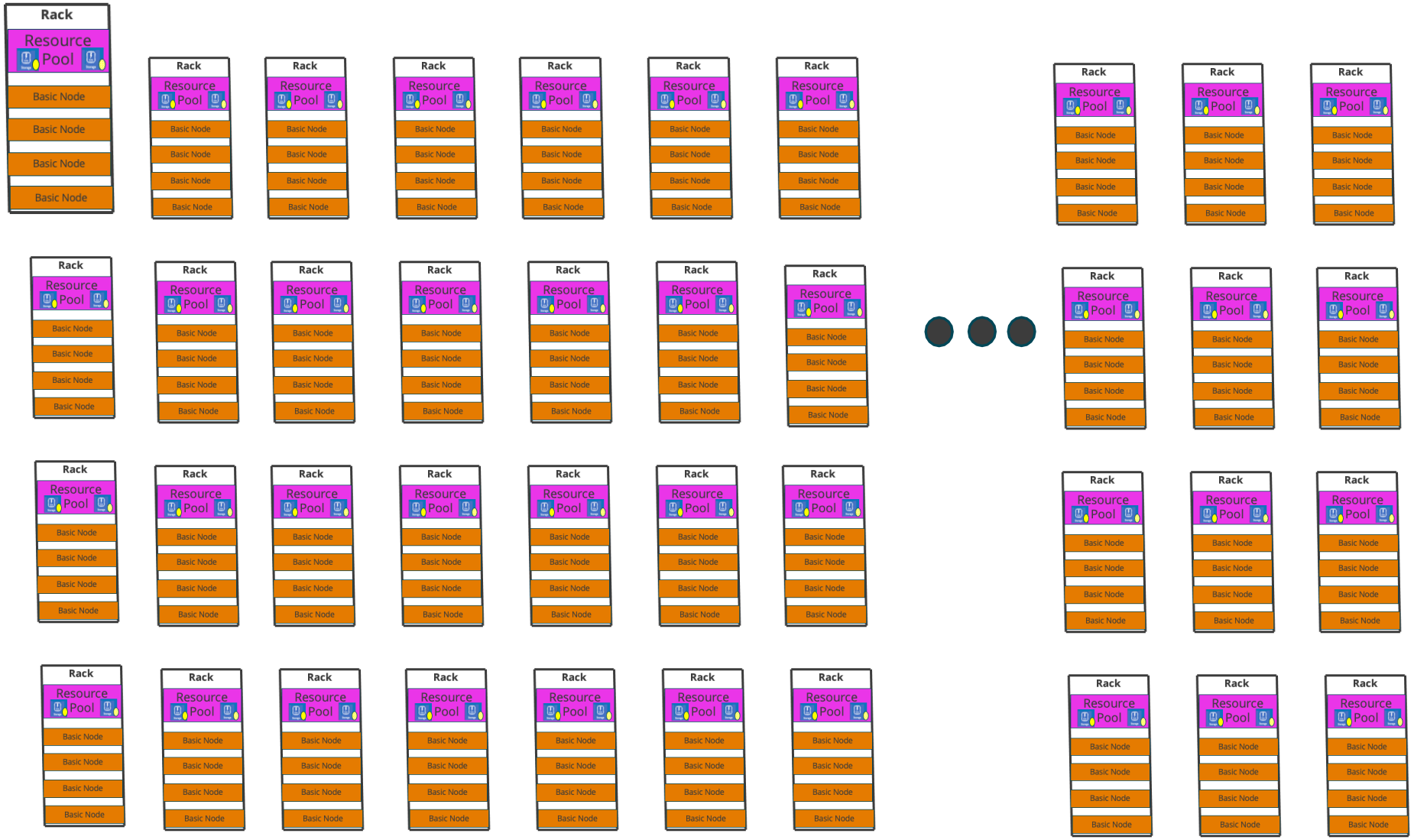
p is the part of the execution in parallel



Quick overview of a Beowulf HPC Architecture, the majority of the HPC Systems

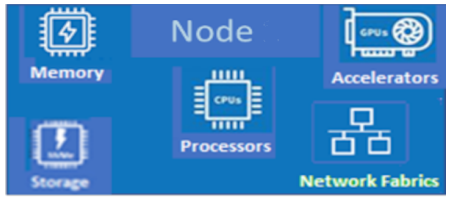
Each task is broken up into smaller subtasks

- Each node is provided with a time to start the execution (Workload Manager)
- Each node communicates with the other nodes to gather information and to do a final assemblage of the result when the computation is complete (ex. Message Passing Interface, Partitioned Global Address Space, Open Multiprocessing)

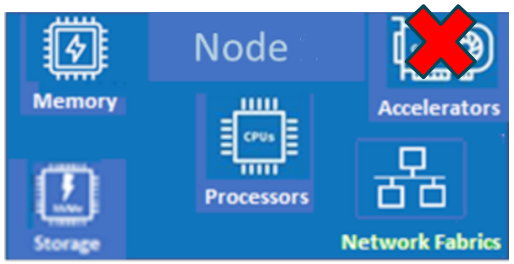


The ‘Big Drawback’ to current Beowulf HPC Architectures

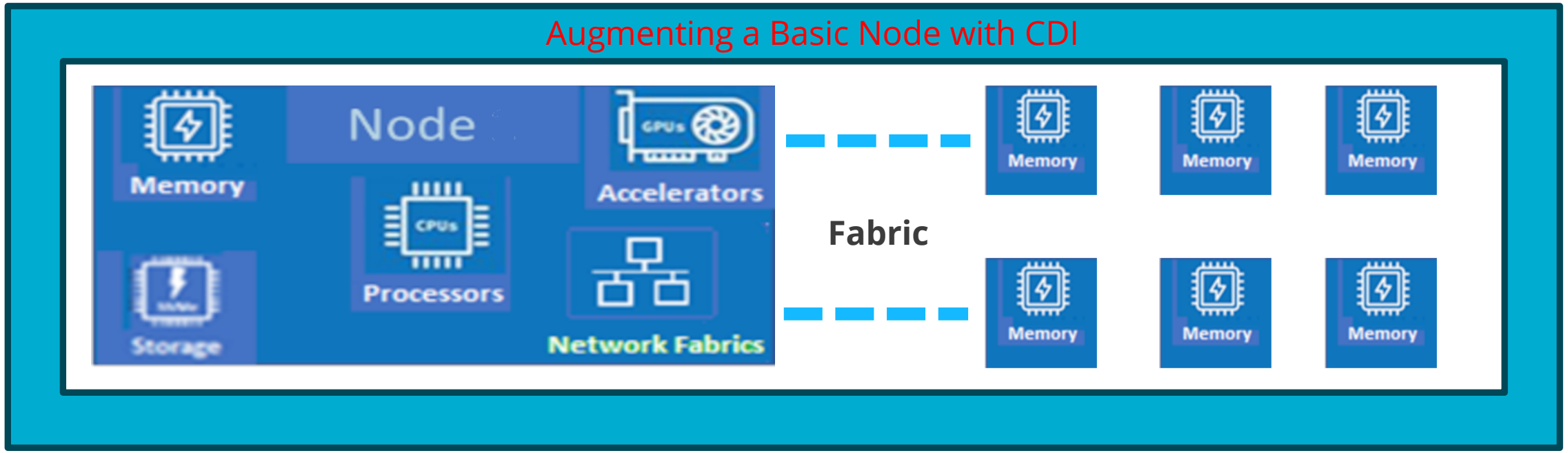
- The larger the HPC system, the greater the potential impact of wasted stranded resources:
  - ‘Stranded Resources’ are compute resources that aren’t used in computations
  - Because resource limits are fixed for each compute node----You get the hardware that the system provides and that’s it, if you need more to match a Batch application need, you are ‘out of luck’ on that system

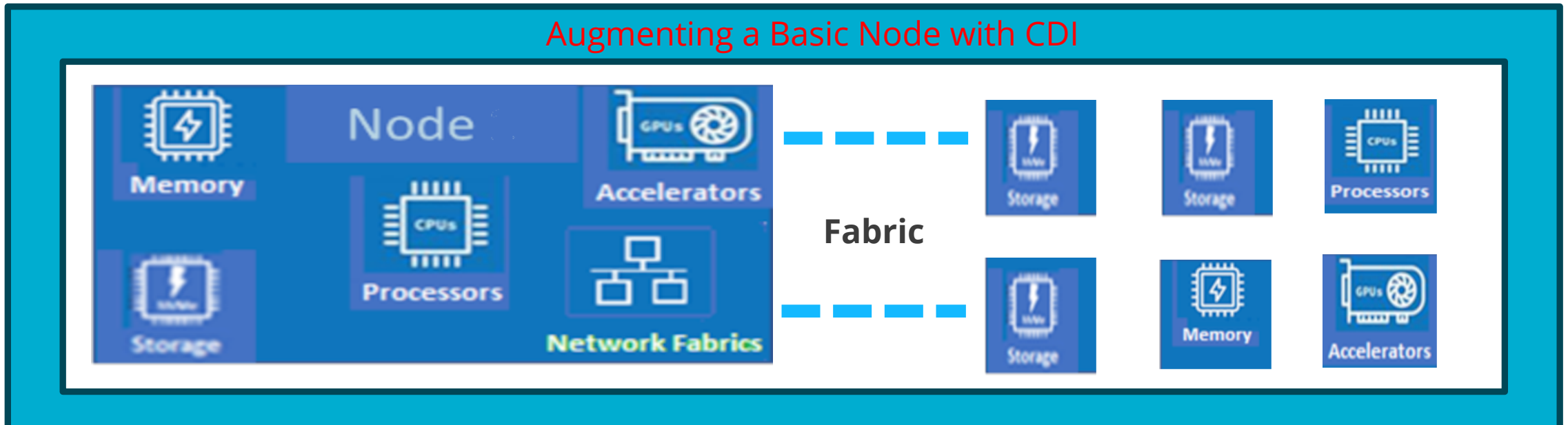


  - Wasted Stranded Resources that are using up energy and generating heat
    - 4% of the World’s Energy Consumption in input into Datacenters (https://www.energy.gov/eere/buildings/data-centers-and-servers)
  - Increased monetary resources to build out components to address all possible types of Application Codes that the HPC must support.
- Hardware failures during the run-time can kill running batch applications.



Composable Disaggregated Infrastructure (CDI)---Software Derived Hardware Architectures



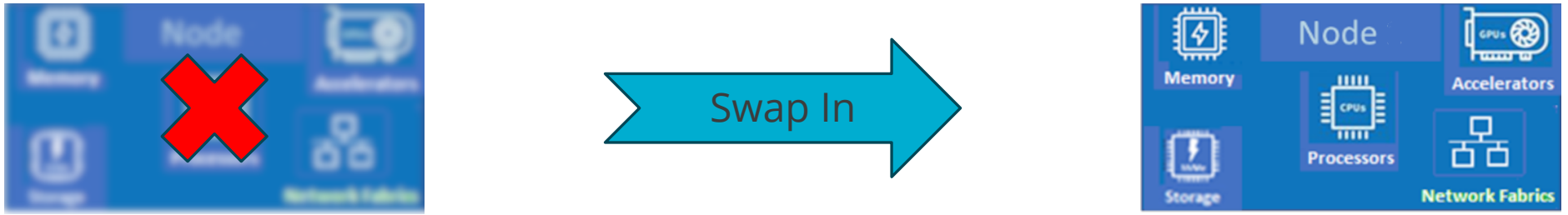


Composable Disaggregated Infrastructure (CDI)--- Software Derived Hardware Architectures

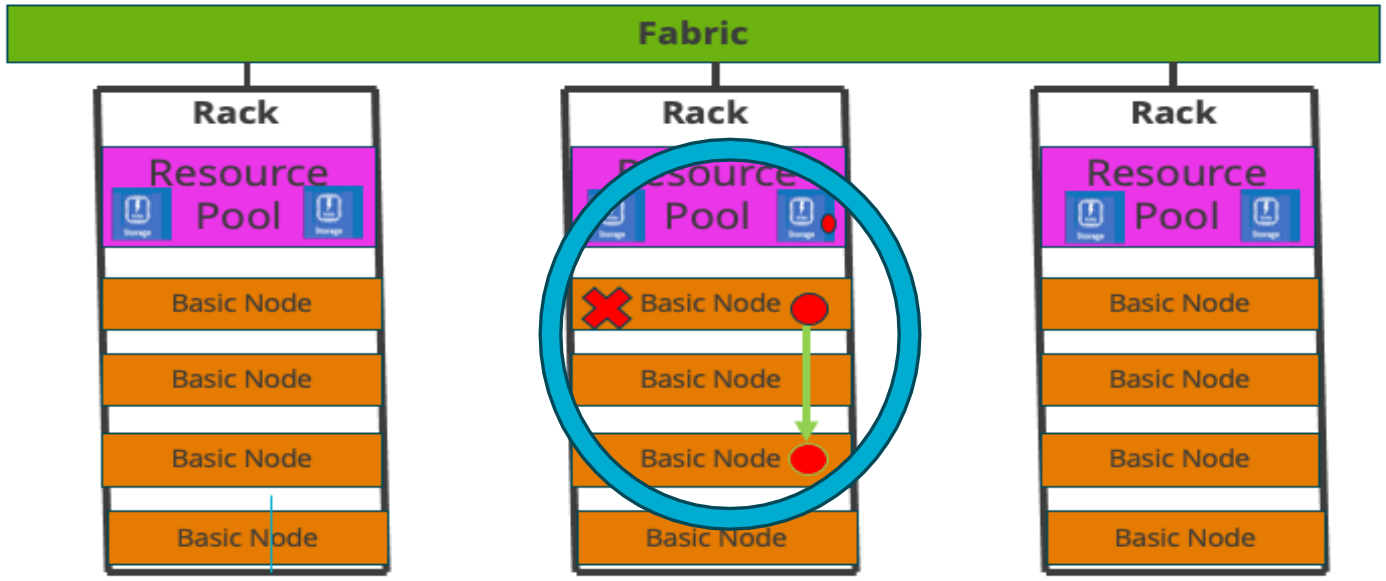
If we need more Storage servers to mitigate load issues, we can compose additional servers and automatically add them into the storage pool.

Composable Disaggregated Infrastructure (CDI)---Software Derived Hardware Architectures

- Computational Stability---A node failure, dynamically swap it out

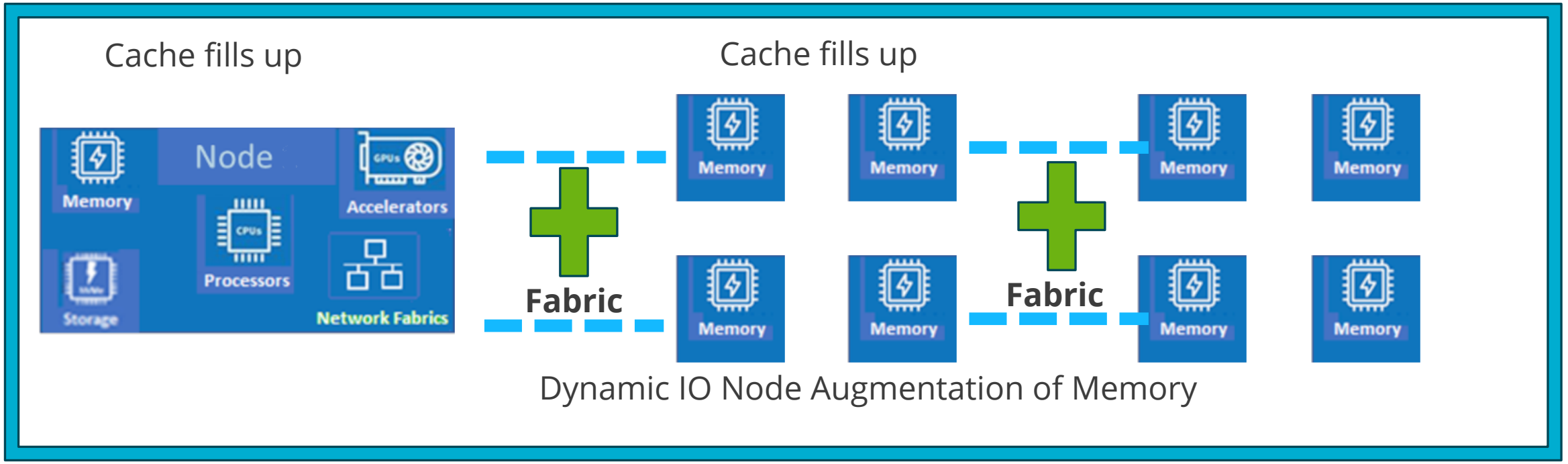


- We can leave a malfunctioning node behind, allocate another node, and **utilize the memory**, from the pool, that the other node was using, **seamlessly**.



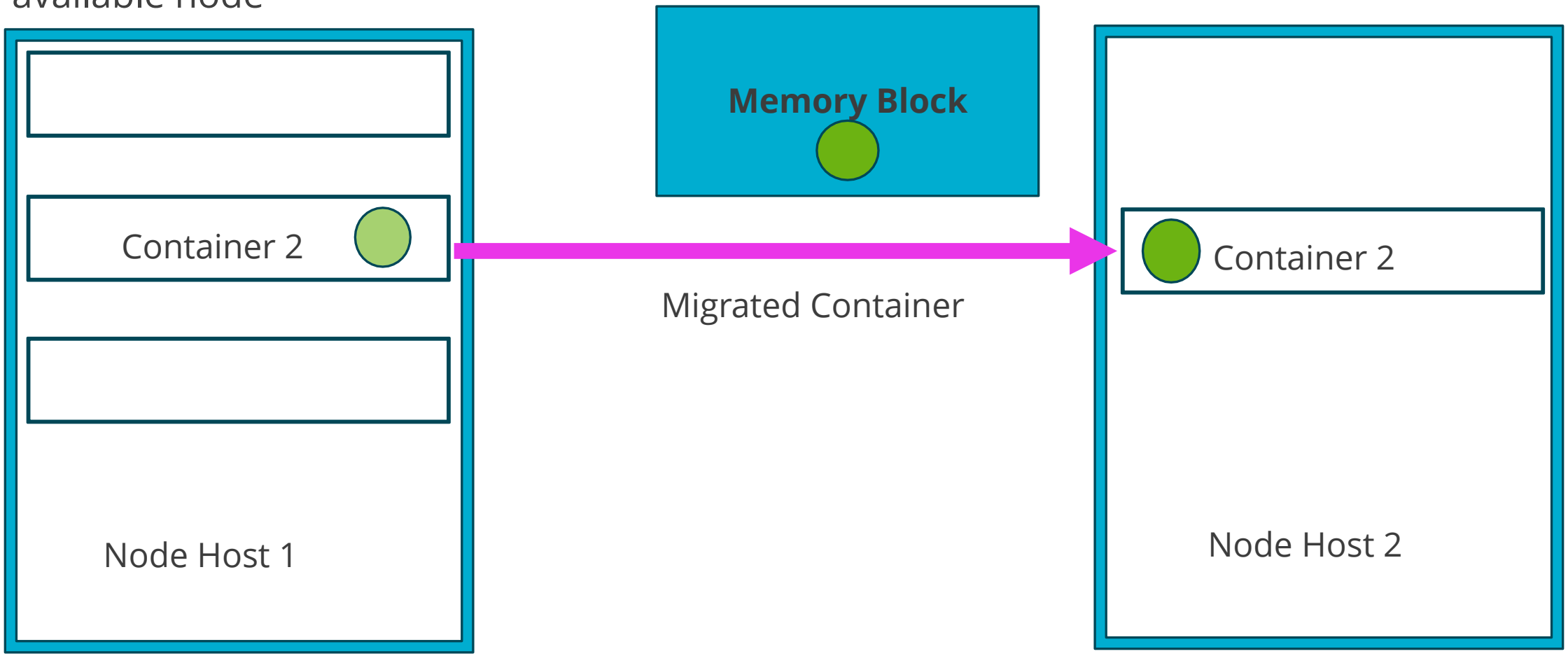
Composable Disaggregated Infrastructure (CDI)---Software Derived Hardware Architectures

Mitigate Filesystem Cache Thrashing by dynamically adding memory to the IO node
(eg. ZFS Adaptive Replacement Cache, XFS Buffer Cache, etc.) to prevent Virtual Memory Swaps from disk.



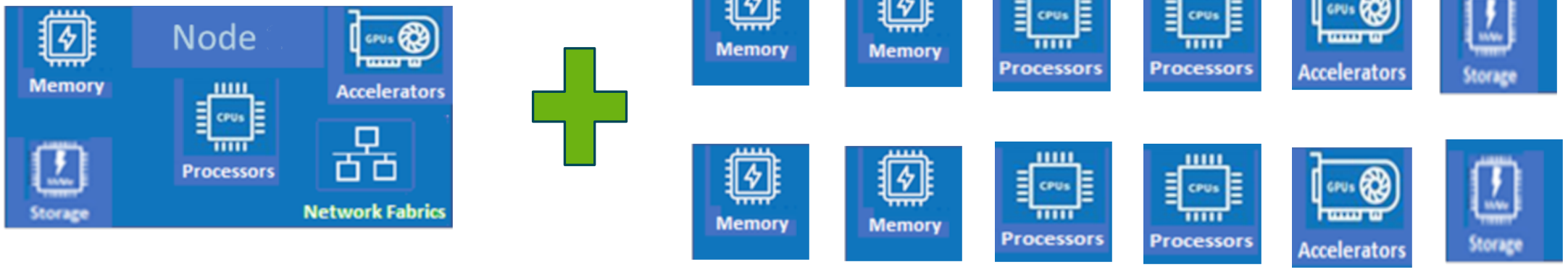
Composable Disaggregated Infrastructure (CDI)---Software Derived Hardware Architectures

Dynamic motion of a container to another available node



Versatility and capabilities for a CDI infrastructure and a Composable Burst-Buffer filesystem
What we can do with such a set-up.

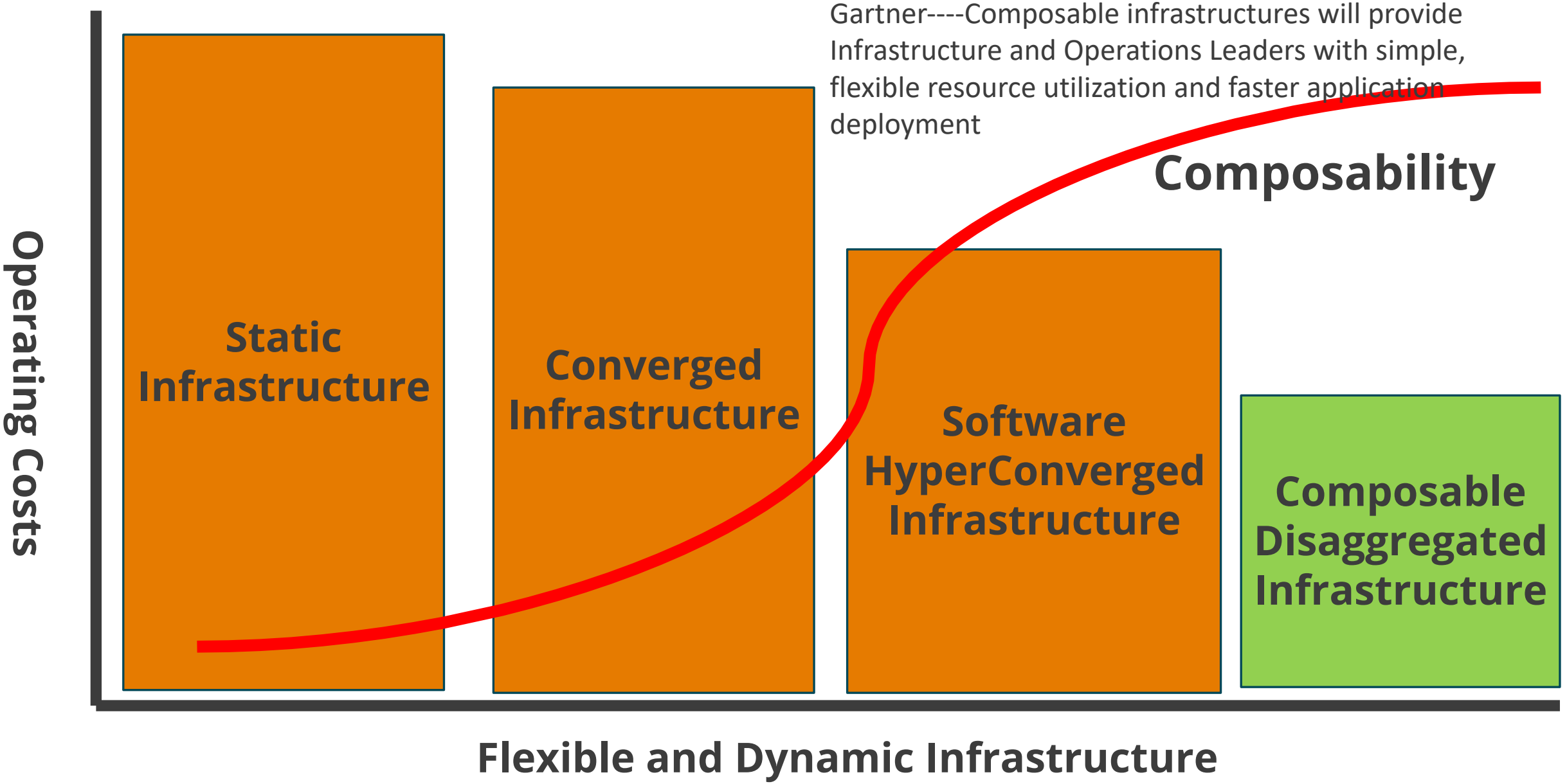
Resources are oversubscribed on a node running a bunch of application elements



Dynamic IO Node Augmentation of Node Resources

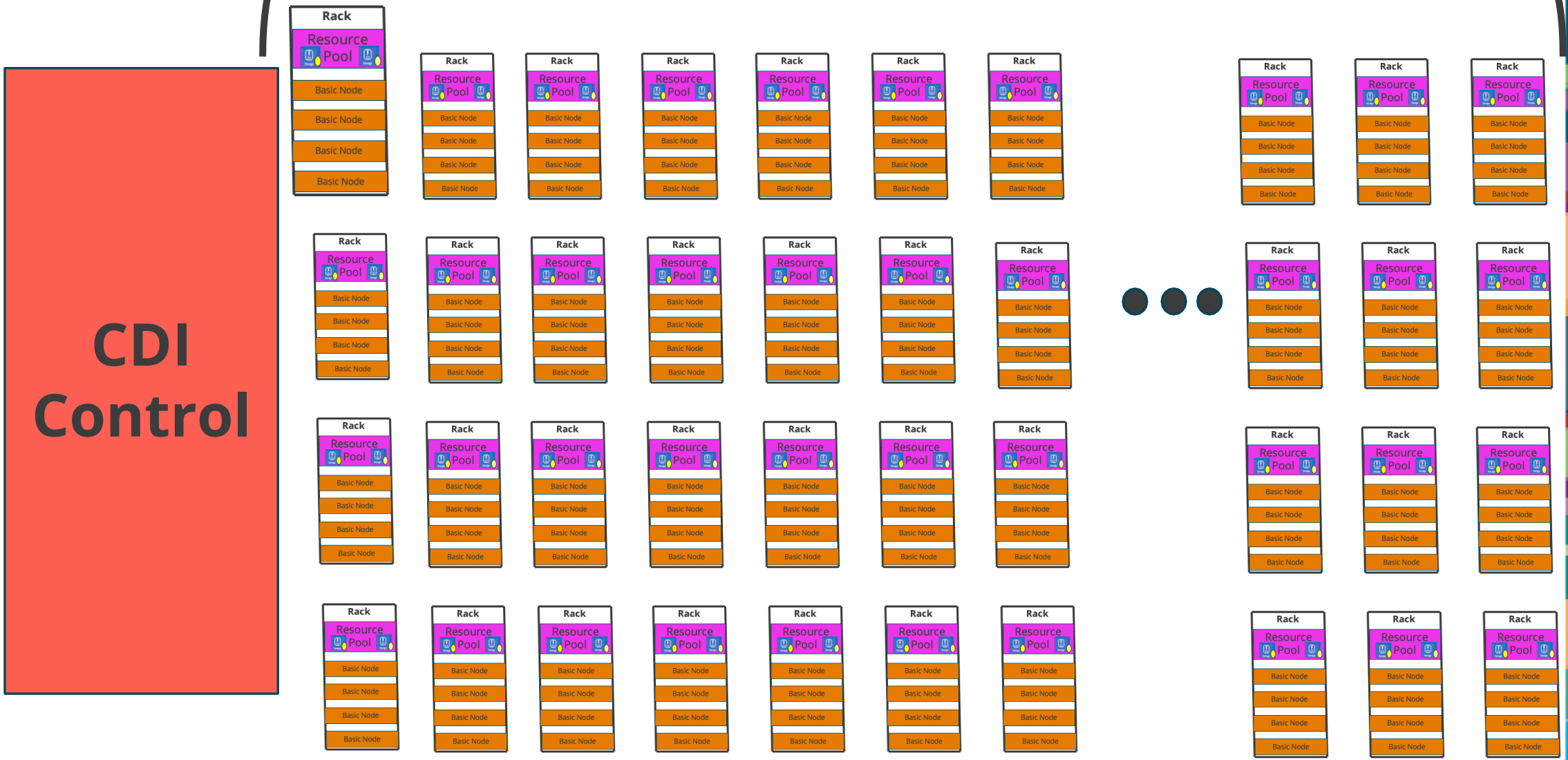
Composable Disaggregated Infrastructure (CDI)---Software Derived Hardware Architectures

Gartner----Composable infrastructures will provide Infrastructure and Operations Leaders with simple, flexible resource utilization and faster application deployment

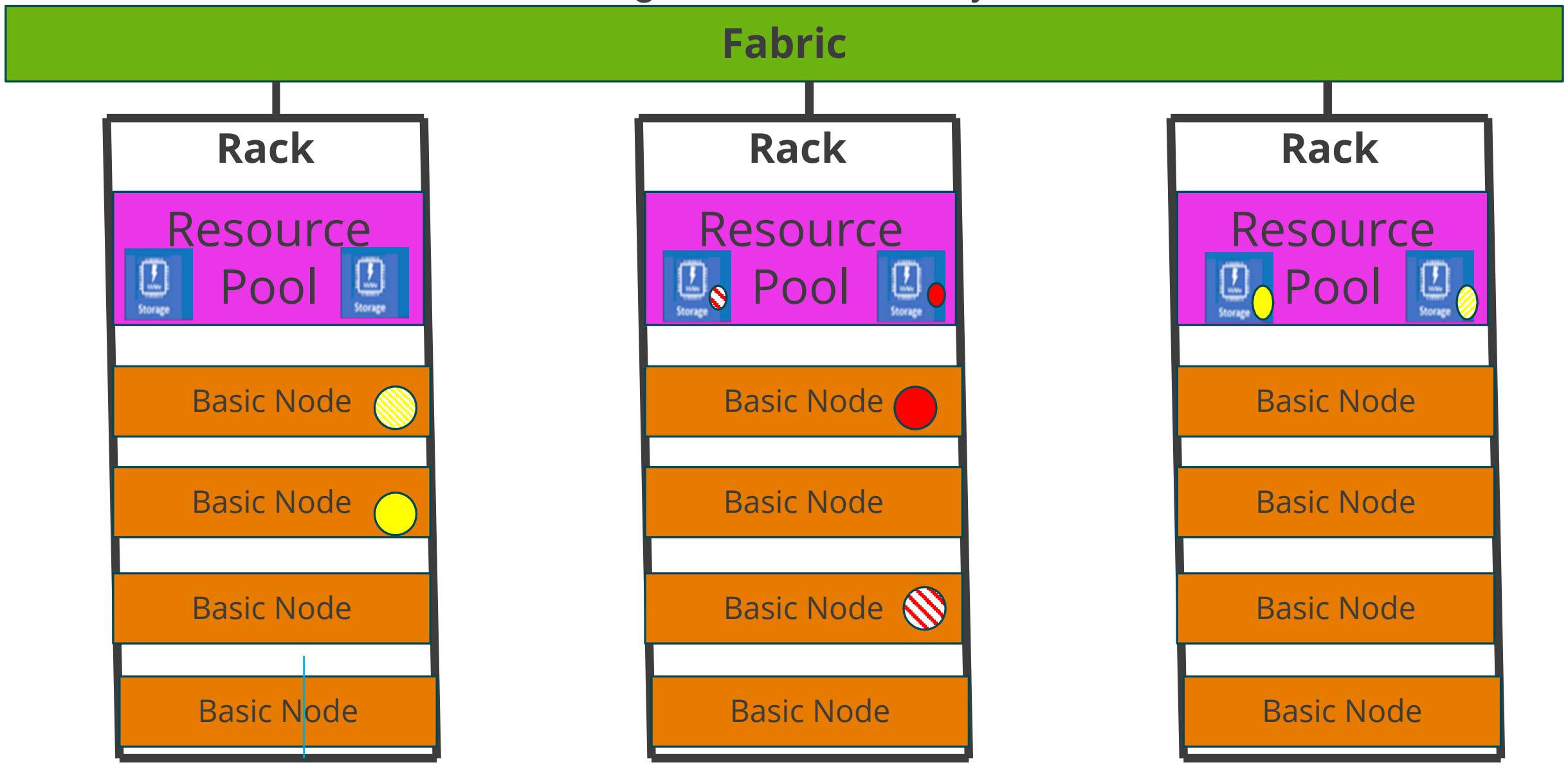


Design Considerations for a Composability Manager on a Large-Scale HPC System

Scaling the control structure To very large HPC systems

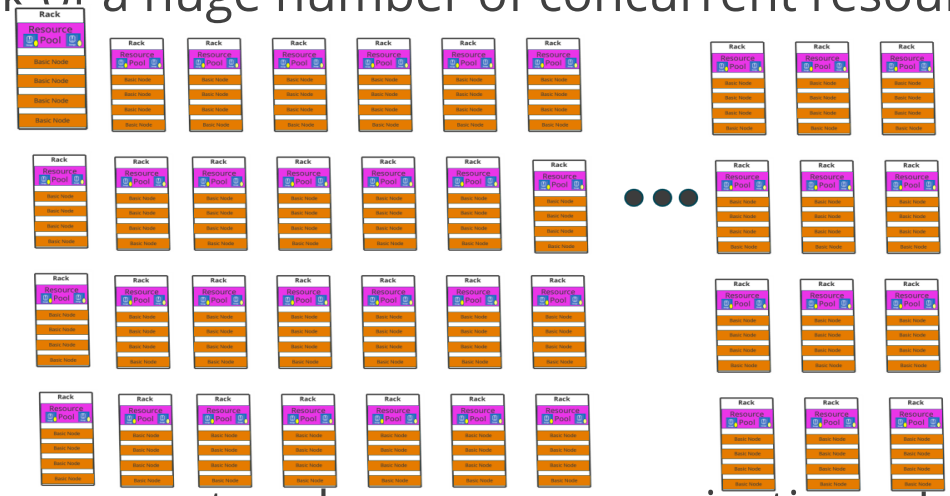


Design Considerations for a Composability Manager on a Large-Scale HPC System



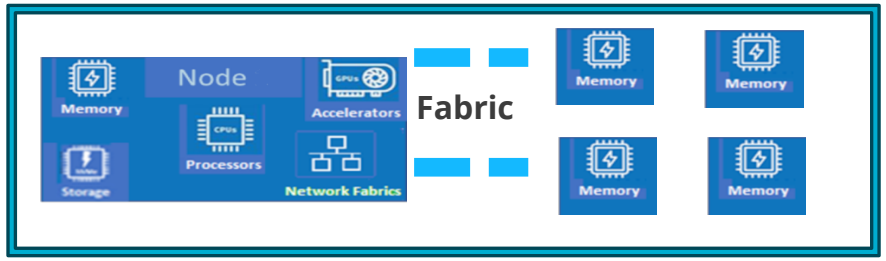
Design Considerations for a Composability Manager on a Large-Scale HPC System

- We need to keep track of a huge number of concurrent resources



- We need to keep management and query communications down to a reasonable quantity

- We need to be able to execute timely changes to the HPC system as those changes are requested



Introducing Sunfish

Clients

Management Layer

Hardware Layer

Application Domain

App driven system reconfig

Administration Domain

- Systems composition
- Systems update

Infra management

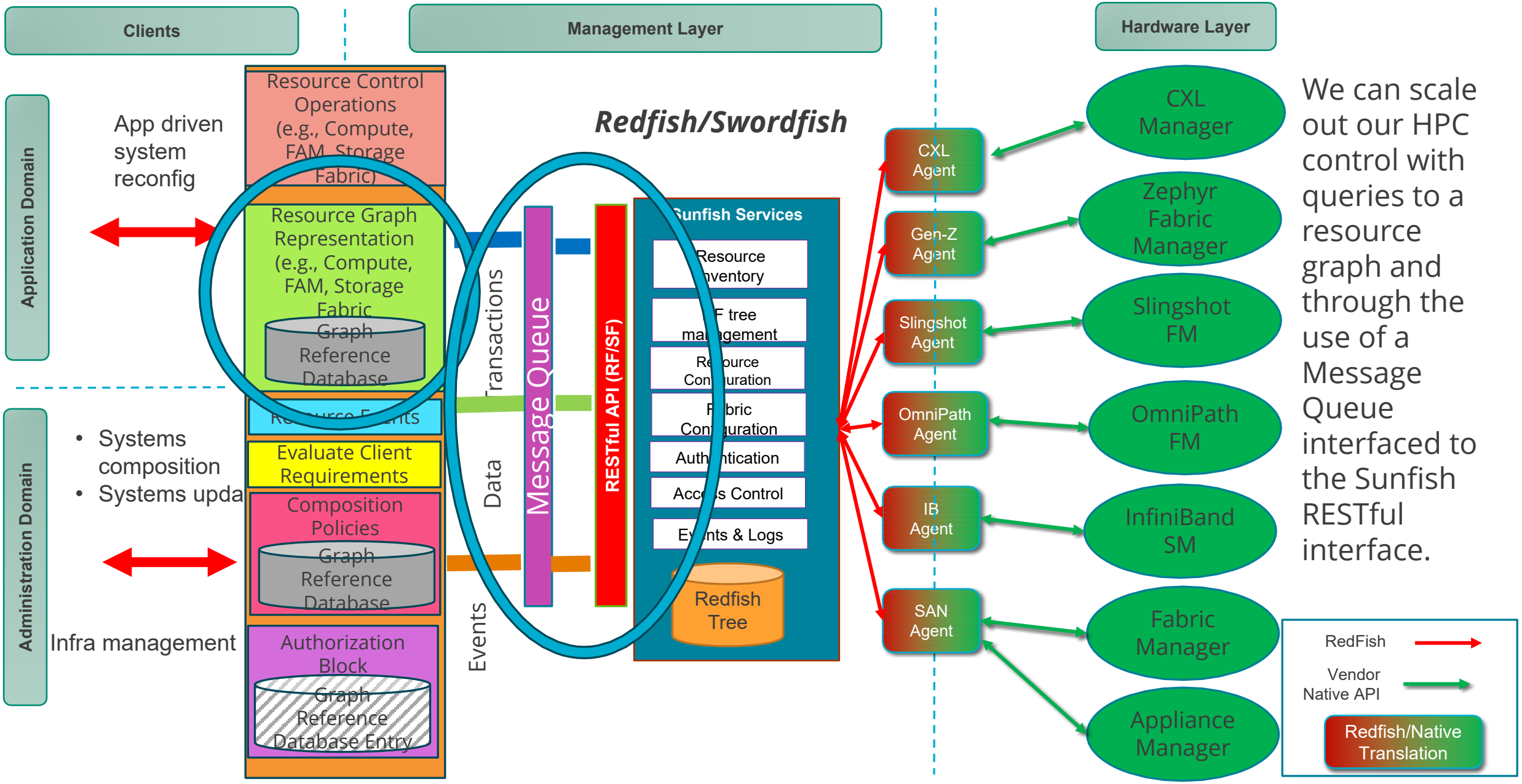
CDI Composition Interface

Composable Disaggregated Infrastructure Database and Tools

Hardware Interface

Hardware Resources

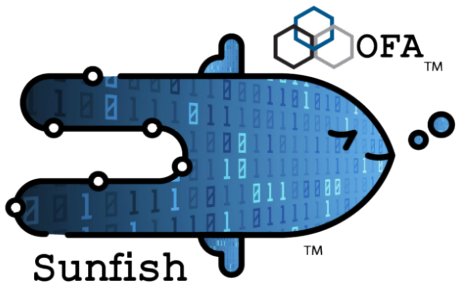
Introducing Sunfish



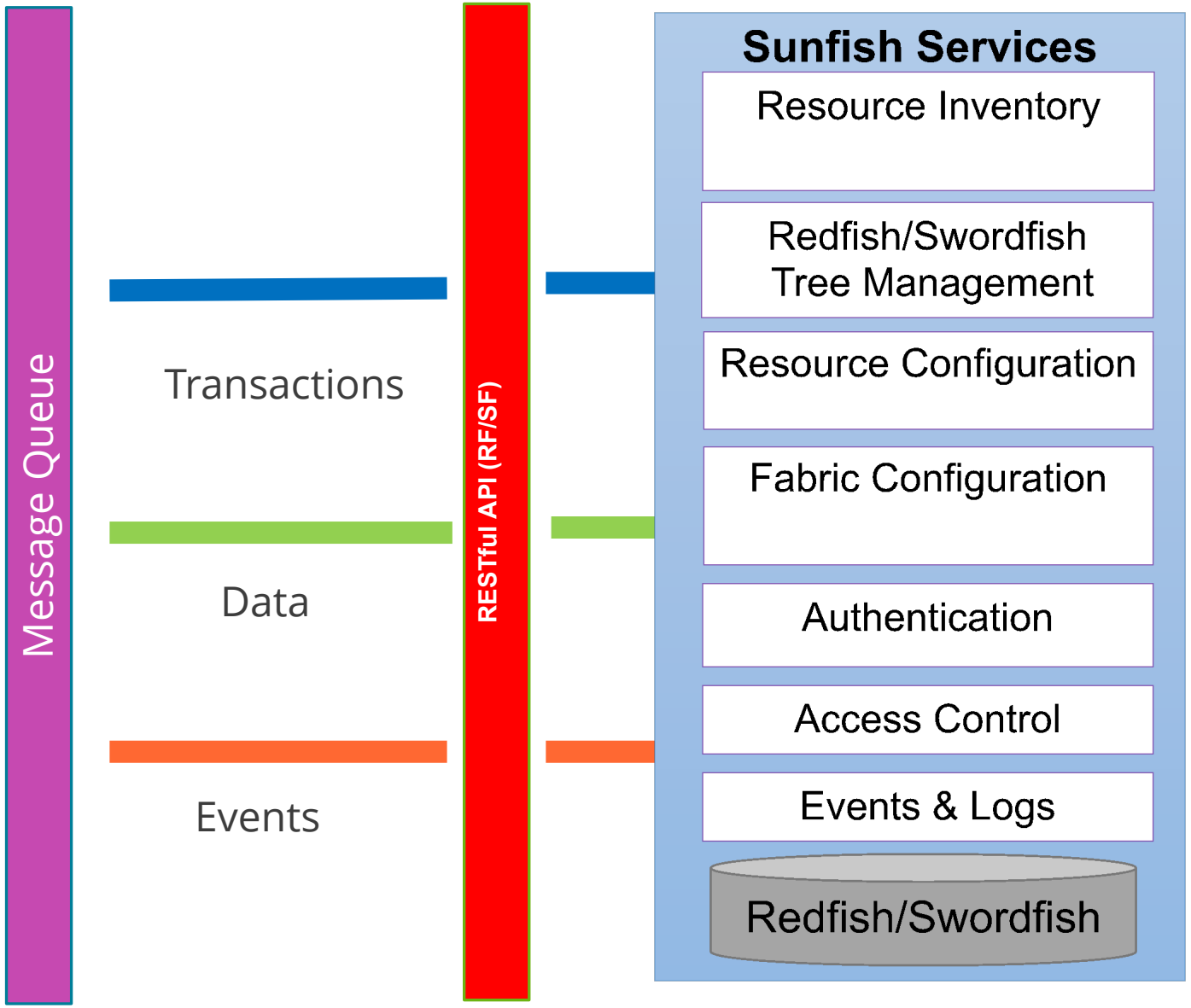
We can scale out our HPC control with queries to a resource graph and through the use of a Message Queue interfaced to the Sunfish RESTful interface.

Sunfish Core Services

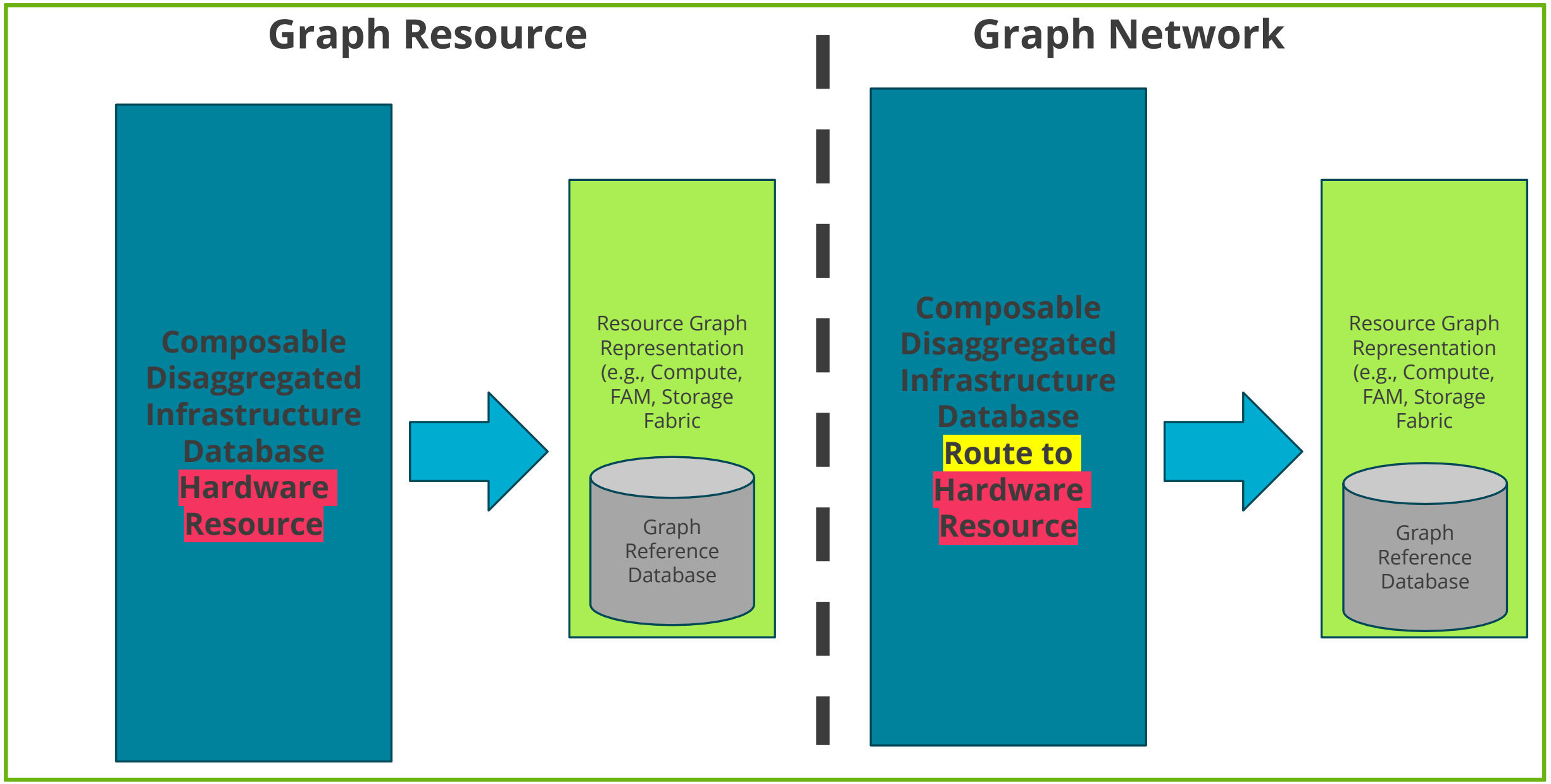
- Redfish/Swordfish Tree
- RESTful Interface
  - Supports message queues like RabbitMQ or Apache Kafka for scaling
- Built-In:
  - Authentication
  - Aggregation Support for Components
  - Event Communications and Subscriptions

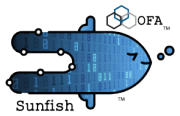


Sunfish Core Services



The Sunfish Composability Management Framework



The Sunfish  Composability Management Framework

Graph Resource

Composer Entry Name	Value Type	Description
ResourceName	String	The Sunfish Resource name
ComposerID	Integer	Installed Identification Number
ResourceType	String	Sunfish Resource Type
ResourceSubType	String	Resource Subtype
ResourceActive	Boolean	True or False
ResourceAllocated	Boolean	True or False
ResourceCharacteristic1	String	Sunfish Resource Characteristic
ResourceCharacteristic2	String	Sunfish Resource Characteristic
ResourceCharacteristic3	String	Sunfish Resource Characteristic
ResourceCharacteristic4	String	Sunfish Resource Characteristic
ResourceCharacteristic5	String	Sunfish Resource Characteristic
ResourceCharacteristic6	String	Sunfish Resource Characteristic
ResourceCharacteristic7	String	Sunfish Resource Characteristic
ResourceCharacteristic8	String	Sunfish Resource Characteristic
ResourceCharacteristic9	String	Sunfish Resource Characteristic
ResourceCharacteristic10	String	Sunfish Resource Characteristic
Message	String	Event Message
MessageID	Integer	Event Message ID
ProposedResolution	String	Error Event BMC Proposed Resolution Message
ResourceEndpointConnectionTypes	String and Comma Separated	Endpoint connection types available to the Resource
ResourceEndpointNames	String and Comma Separated	Endpoint connection names
ResourceEndpointConnectionBandwidths	Integers and Comma Separated	Bandwidth performance Values for the Connections
ResourceLocationPath	String	Reference Resource location path in Sunfish tree
AggregatedDevices	String and Comma Separated	Aggregated Devices
Tenancy	String	Tenancy
SecurityValue	String	Security Property Value
SecurityAssociation	String	Security Association

Graph Network

Composer Entry Name	Value Type	Description
ResourceName	String	The Sunfish Resource name
ComposerID	Integer	Installed Identification Number
ConnectionType	String	Sunfish Connection Type
ConnectionVersion	String	Sunfish Connection Version
LinkSpeed	Long Long	Link Bandwidth
Manufacturer	String	Manufacturer Name
FECN	Integer	Forward Explicit Congestion Notification
BECN	Integer	Backwards Explicit Congestion Notification
PerformanceIssues	Boolean	Link has problems
BytesTransmitted	LongLong	Counter
BytesReceived	LongLong	Counter
TransmitDiscards	Integer	Counter
TransmitPackets	LongLong	Counter
ReceivePackets	LongLong	Counter
LinkSpecificRecoveryFlags	Integer	How many times have we recovered from an error?
QueueBufferOverrun	Boolean	We ran over the buffer?
LinkCharacteristic1	String	Sunfish Resource Characteristic
LinkCharacteristic2	String	Sunfish Resource Characteristic
LinkCharacteristic3	String	Sunfish Resource Characteristic
LinkCharacteristic4	String	Sunfish Resource Characteristic
LinkCharacteristic5	String	Sunfish Resource Characteristic
LinkCharacteristic6	String	Sunfish Resource Characteristic
LinkCharacteristic7	String	Sunfish Resource Characteristic
LinkCharacteristic8	String	Sunfish Resource Characteristic
LinkCharacteristic9	String	Sunfish Resource Characteristic
LinkCharacteristic10	String	Sunfish Resource Characteristic
Message	String	Event Message
MessageID	Integer	Event Message ID
ProposedResolution	String	Error Event BMC Proposed Resolution Message
ResourceLocationPath	String	Reference Resource location path in Sunfish tree
Tenancy	String	Tenancy
SecurityValue	String	Security Property Value
SecurityAssociation	String	Security Association

The Sunfish Composability Management Framework

Gremlin Queries. Resource and Network Changes are mirrored to Sunfish

g.V().has('ComposerID',100)

==>v[8288]

g.V().has('ComposerID',100).values('ResourceName')

results in:

==>CXL Memory Pool 1

g.V().has('ComposerID',100).values('ResourceEndpointConnectionBandwidths')

results in:

==>resource endpoint connection values’

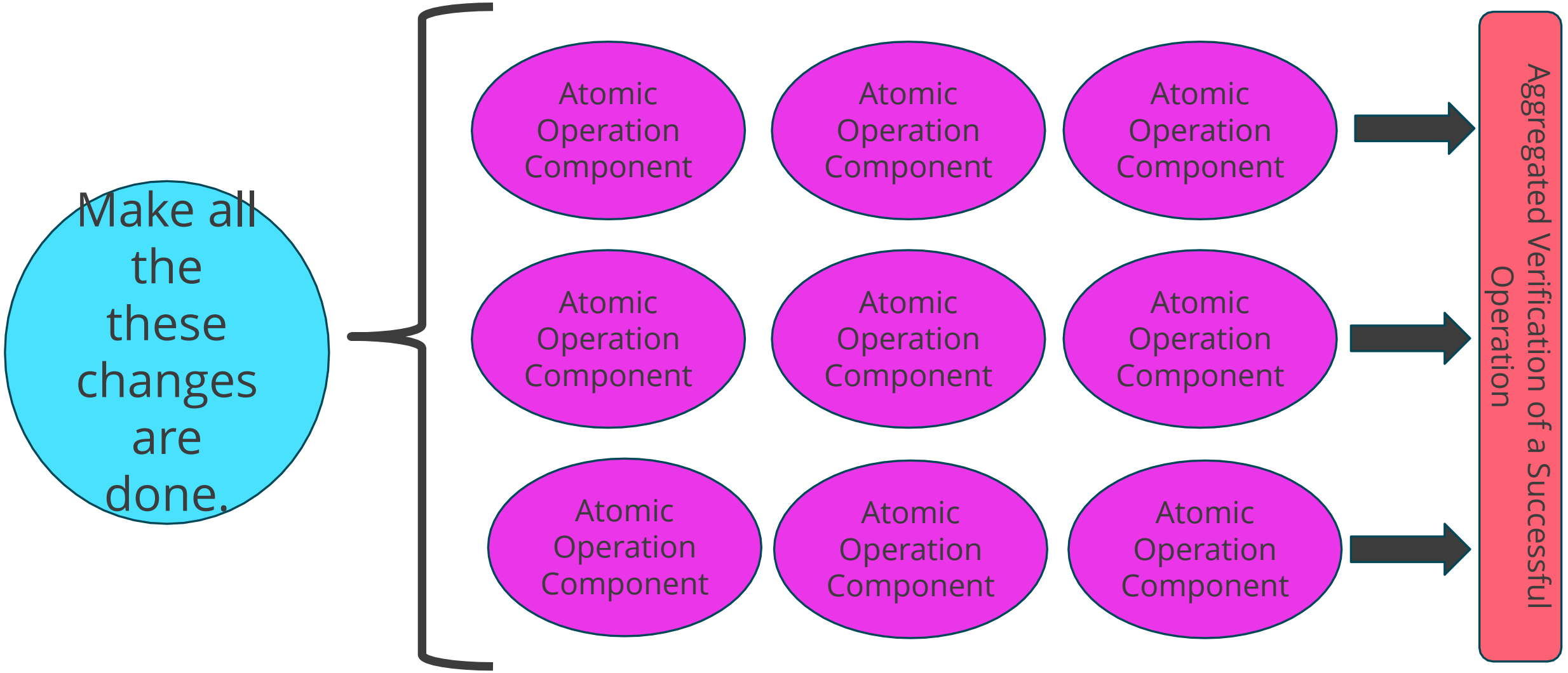
g.E().hasLabel('Nic 1 LID Nic 2 LID')

==>e[a9x-36w-i6t-3ao][4136-Nic 1 LID Nic 2 LID->4272]

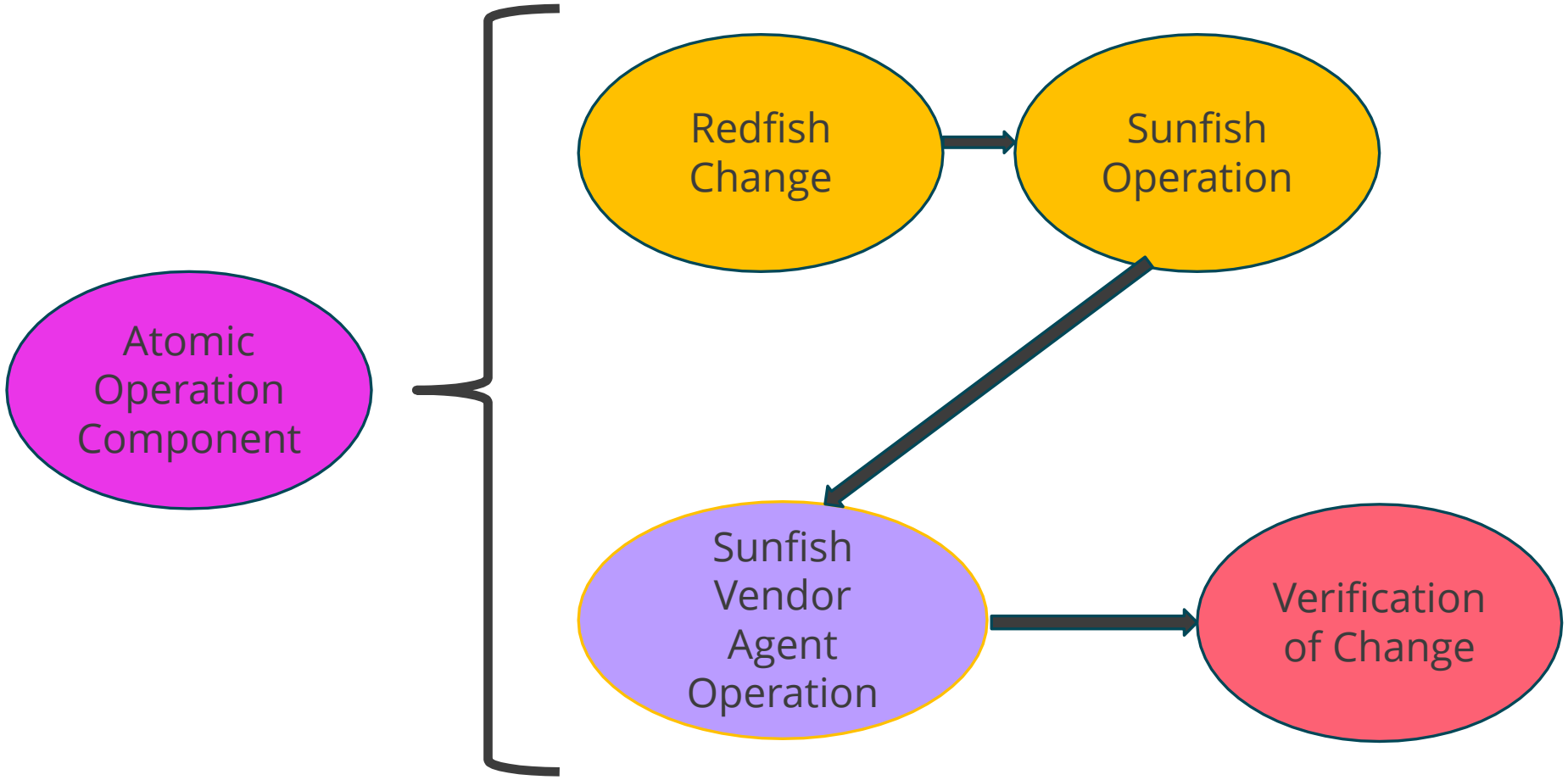
g.E().has('MessageID',260)

==>e[emd-36w-i6t-3ao][4136-Nic 1 LID Nic 2 LID->4272

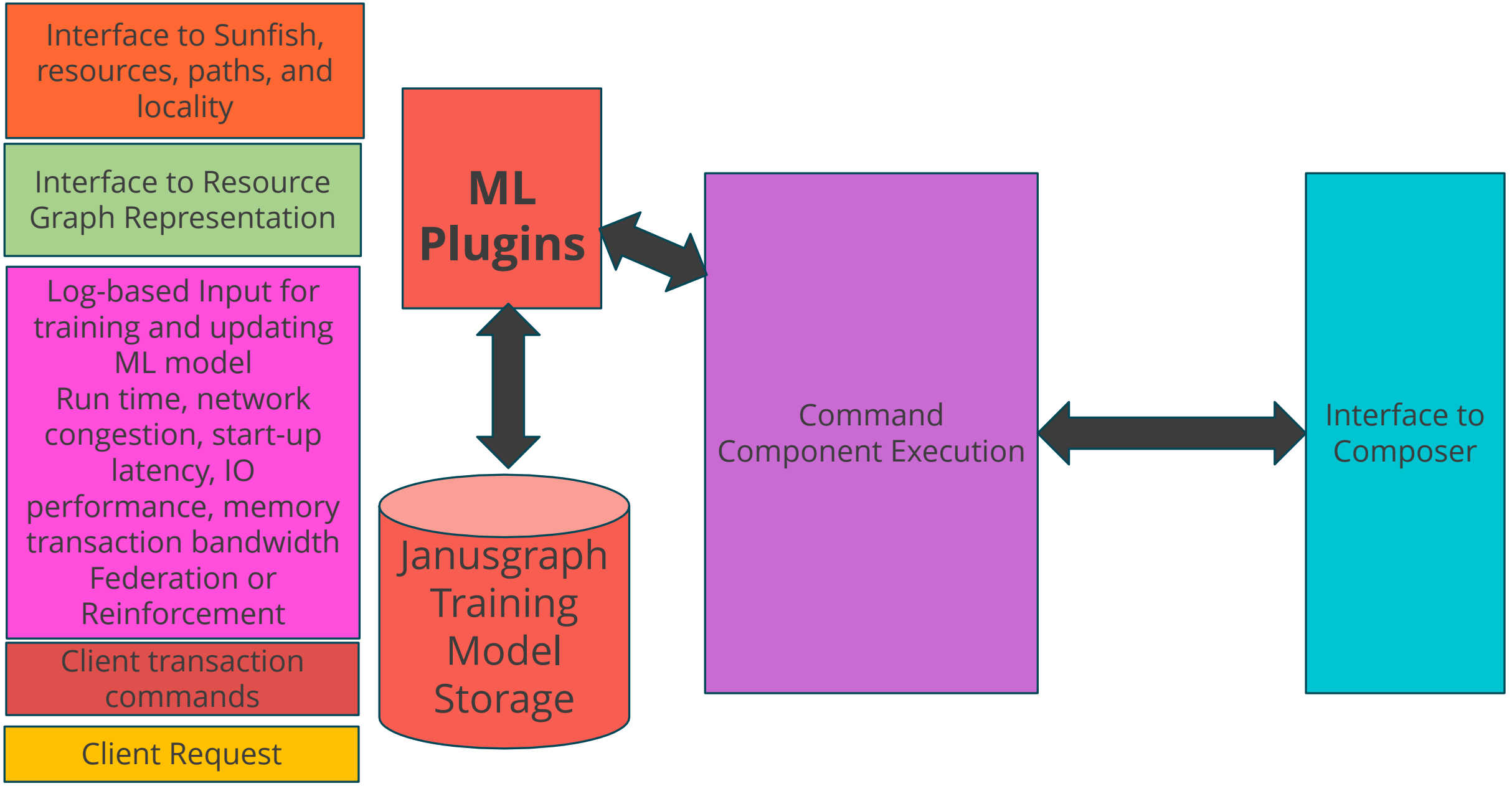
Each request is a single imperative operation
Do we have a partial success or failure?



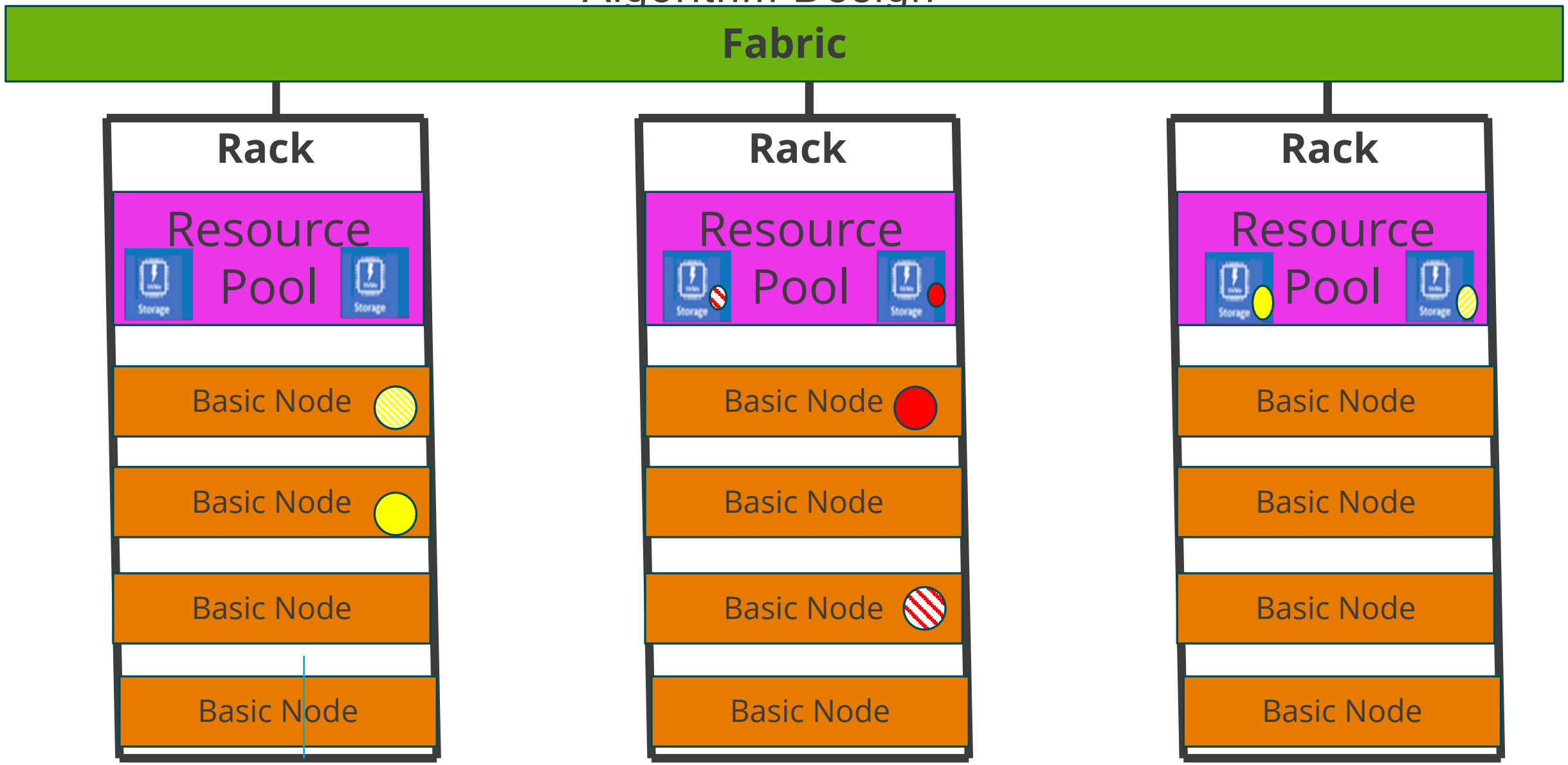
Each request is a single imperative operation
Do we have a partial success or failure?



The Sunfish Composability Management Framework



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



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

ML Plugins

Log-based Input for training and updating ML model
Run time, network congestion, start-up latency, IO performance, memory transaction bandwidth
Federation or Reinforcement

Problem: Need a good way to schedule jobs on the appropriate resources

Job scheduling

- Heuristics can be fair, but often aren't the most efficient
- Optimization algorithms must be highly tailored to specific machines

Resource scheduling

- The key to a composable system!

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

What is Reinforcement Learning?

A procedure that learns an optimal policy (behavior) through observations of interactions with its environment.

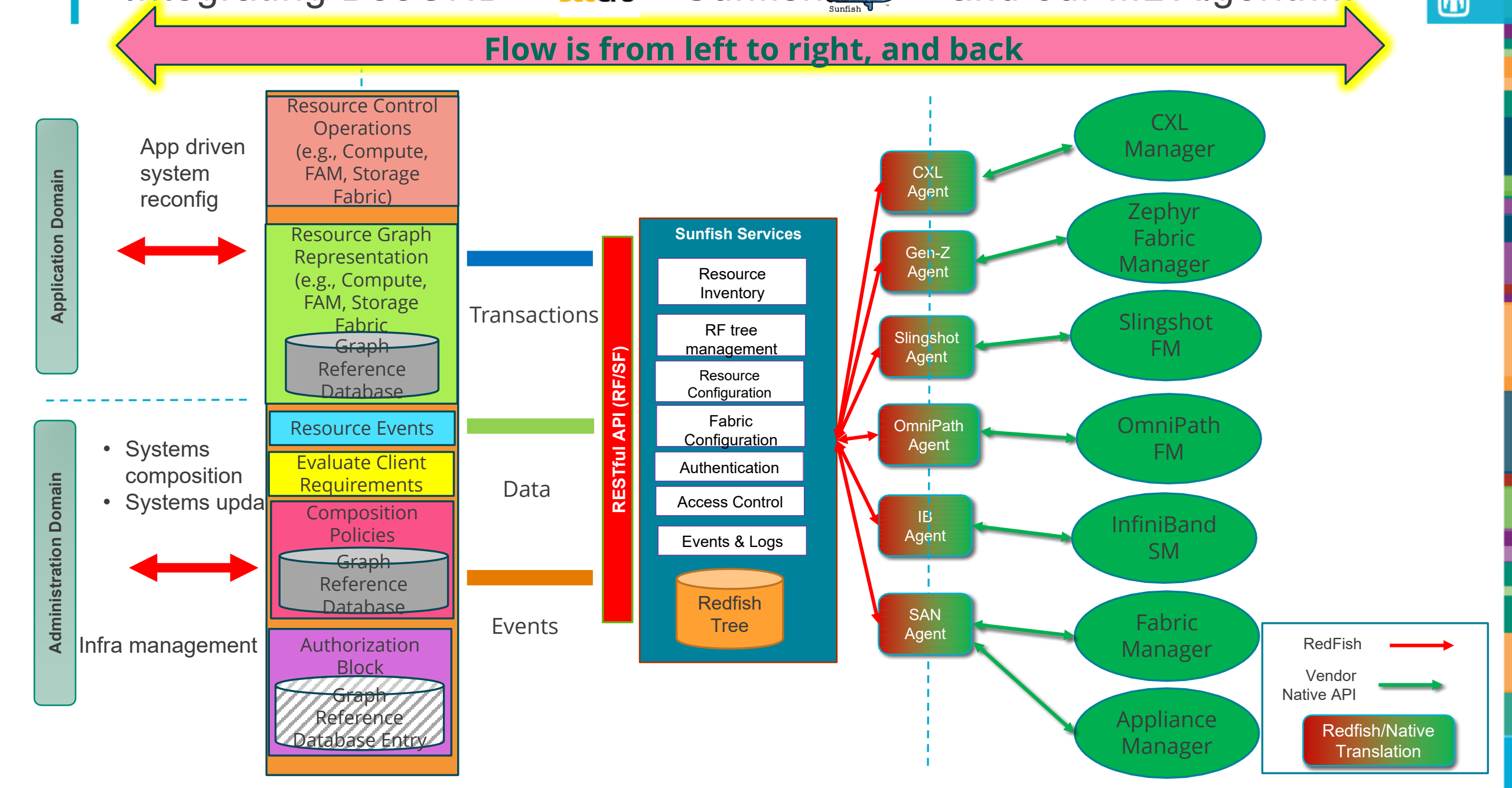
Why is this helpful?

- Allows us to set up a customized reward function, we can penalize long queue times
- Learns a better algorithm for scheduling over time, adapts to changes in traffic volume

Are there cons?

- Without careful attention, can be prone to job starvation
- Won’t backfill without explicit instruction to do so

Integrating BeeOND BeeGFS Sunfish and our ML Algorithm

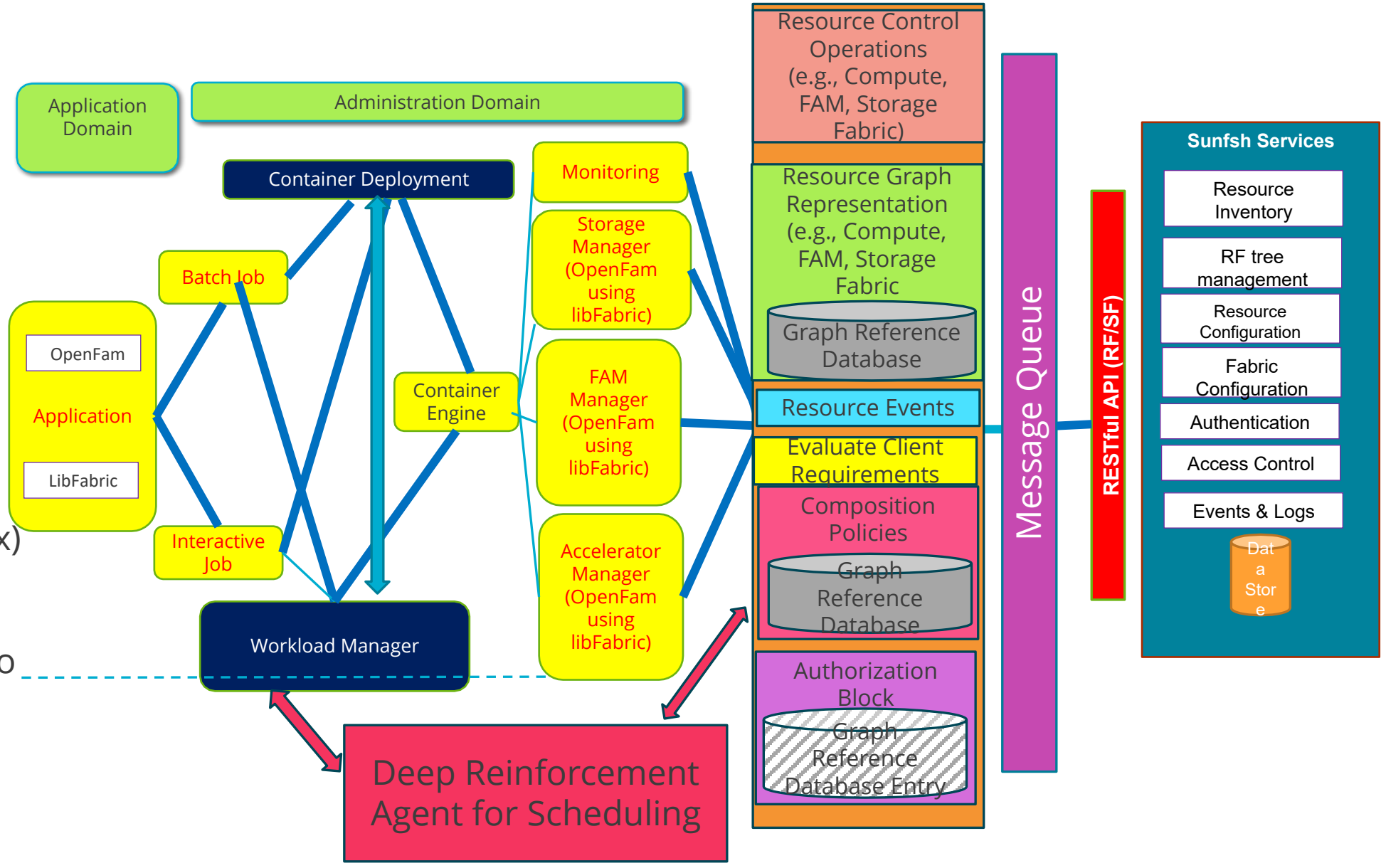


Integrating BeeOND, Sunfish, and our ML Algorithm
The Sunfish Composability Management Framework

Hardware execution is performed using Sunfish connected hardware Agents

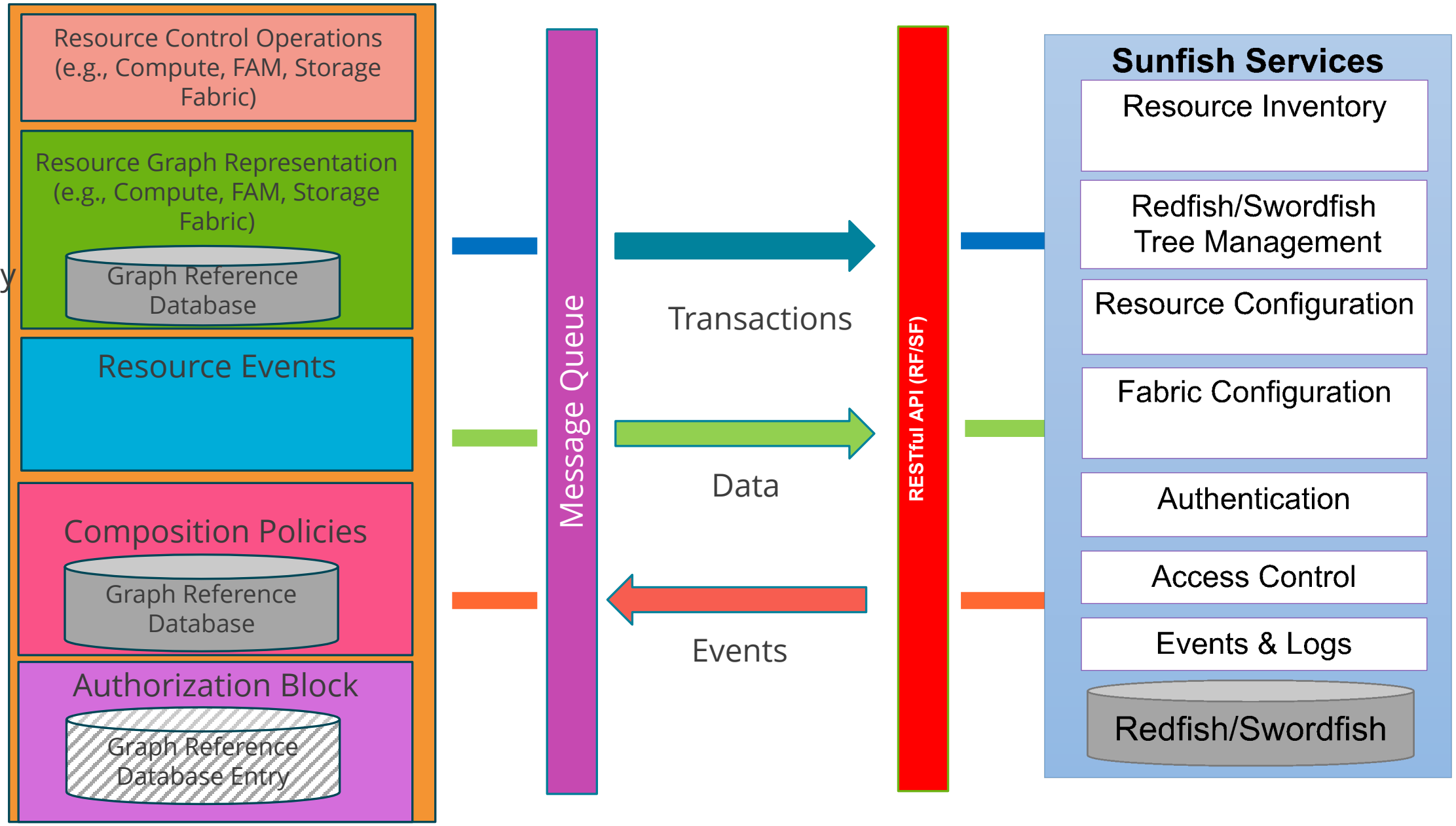
Management of the HPC System is performed by the Sunfish core services.

A Workload Manager (example Slurm or Flux) allocates nodes and requests hardware Resources as a client to the Composability Manager.



Integrating BeeOND, Sunfish, and our ML Algorithm

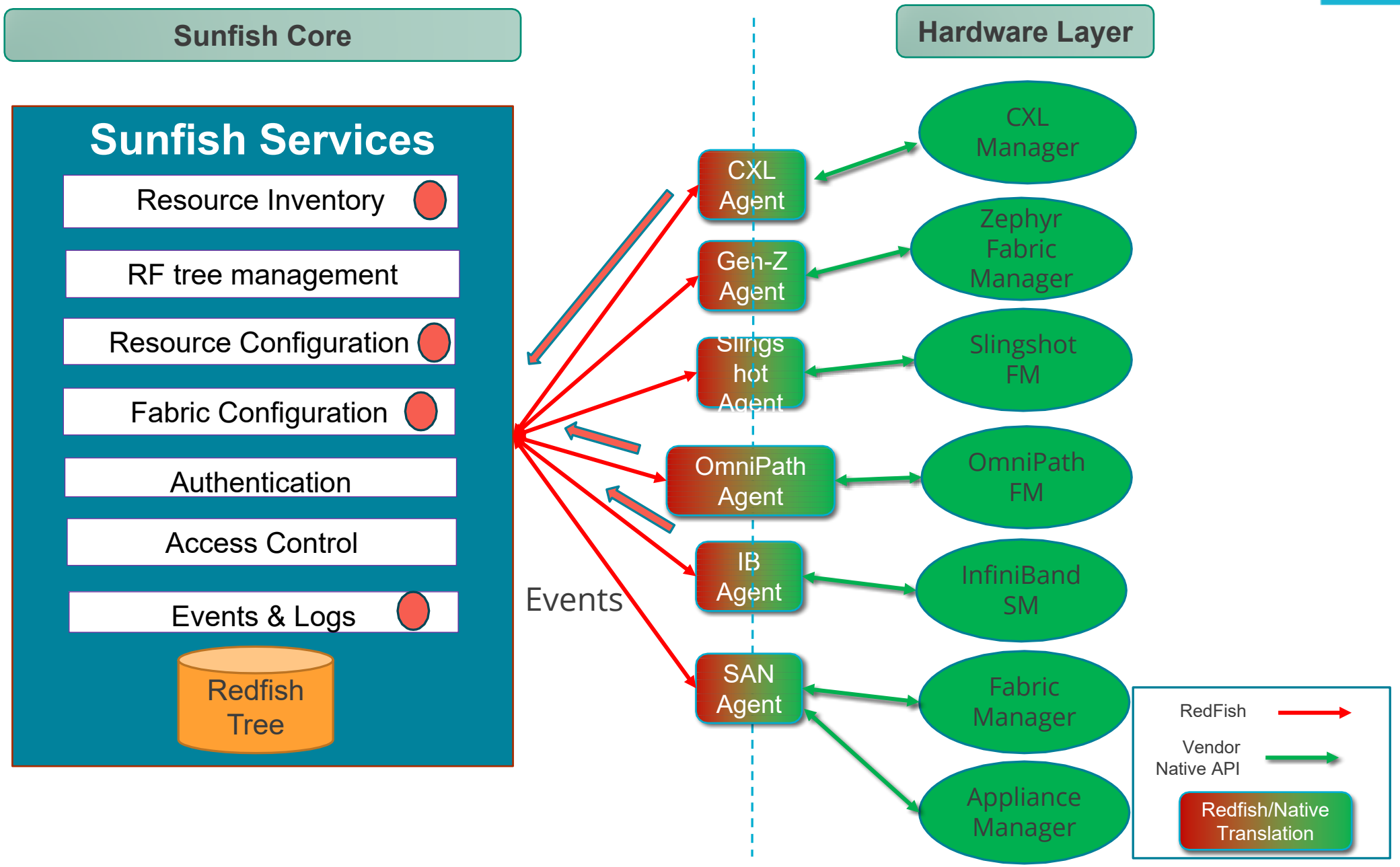
The Composability Manager requests the hardware Resources from Sunfish



Integrating BeeOND, Sunfish, and our ML Algorithm

Sunfish orders the hardware Agents to aggregate fabric routes and endpoints to fulfill a request for NVMeoF.

Events are generated and propagated back through the Composability Manager to the Workload Manager

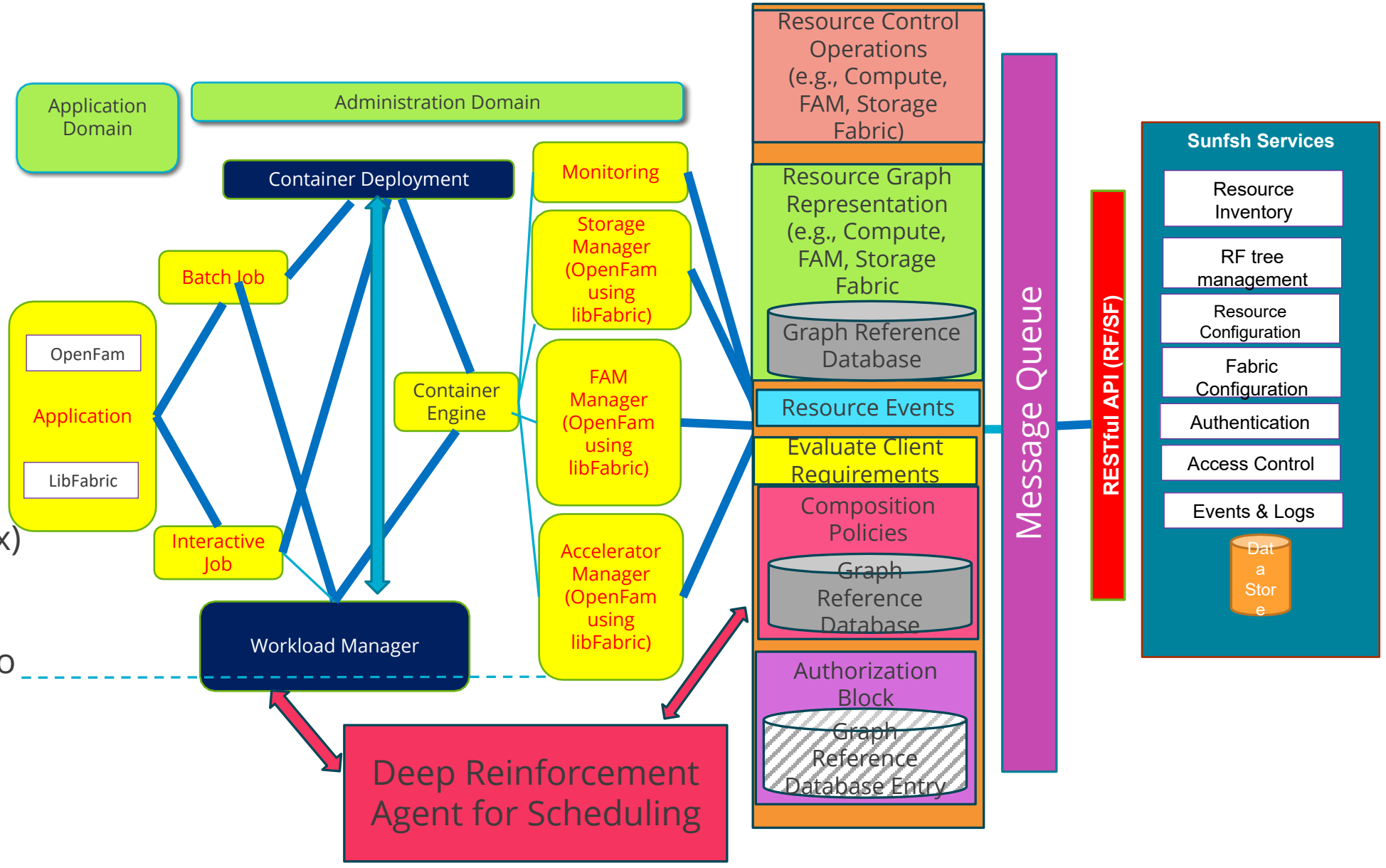


Integrating BeeOND, Sunfish, and our ML Algorithm
The Sunfish Composability Management Framework

Hardware execution is performed using Sunfish connected hardware Agents

Management of the HPC System is performed by the Sunfish core services.

A Workload Manager (example Slurm or Flux) allocates nodes and requests hardware Resources as a client to the Composability Manager.



Acknowledgements and Questions?

- Thinkparq
  - Troy Patterson, Philipp Falk
- OpenFabrics Alliance
  - Doug Ledford, Phil Cayton
- OpenFabrics Management Framework Working Group
  - Christian Pinto, Richelle Ahlvers, Russ Herrell, Michele Gazzetti, Jeff Hilland, John Mayfield, Jim Hull, Tracy Spitler, Chris Morrone, etc.
- Sandia Labs
  - Steve Monk, Matt Curry, Jeff Ogden, Joe Mervini, Doug Pase