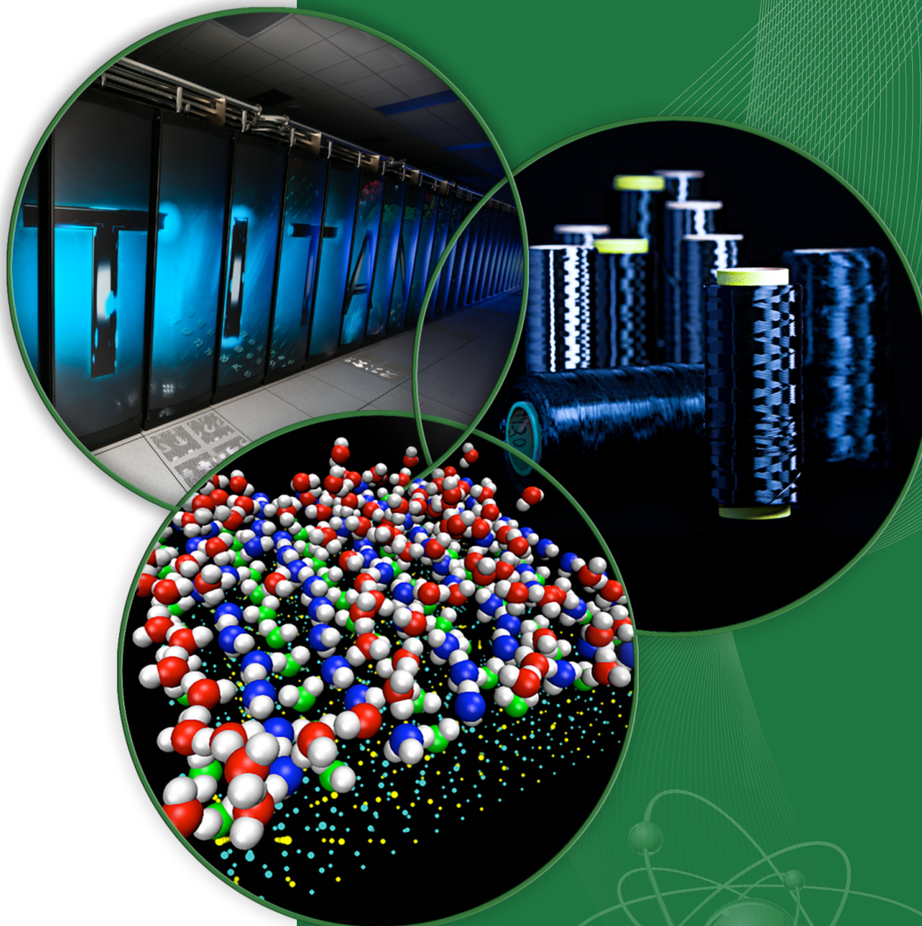
UNITY: Unified Memory and File Space



Terry Jones, ORNL

June 27, 2017

Terry Jones, ORNL PI

Michael Lang, LANL PI

Ada Gavrilovska, GaTech PI

Talk Outline

- Motivation
- UNITY's Architecture
- Early Results
- Conclusion

Timeline to a Predicament – APIs & Growing Memory Complexity

- Problem
  - The simple bifurcated memory hierarchy of the 1950s has evolved into a much more complex hierarchy while interfaces have remained relatively fixed.
  - At the same time, computer architectures have evolved from single nodes to large parallel systems.
- Solution
  - Update the interface to support a prescriptive (directive based) approach.
  - Manage dynamic placement & movement with a smart distributed runtime system
- Impact
  - Enable domain scientist to focus on their specialty rather than requiring them to become experts on memory architectures.
  - Enable target independent programmability & target independent performance.

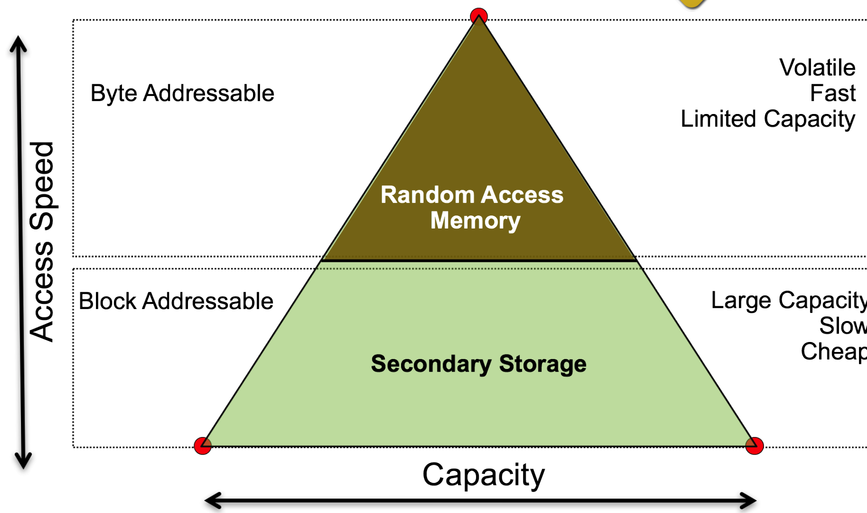
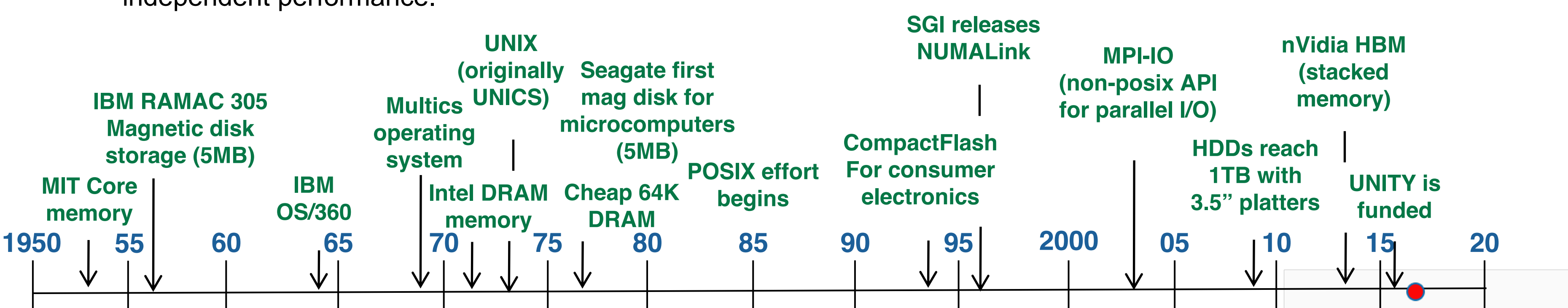
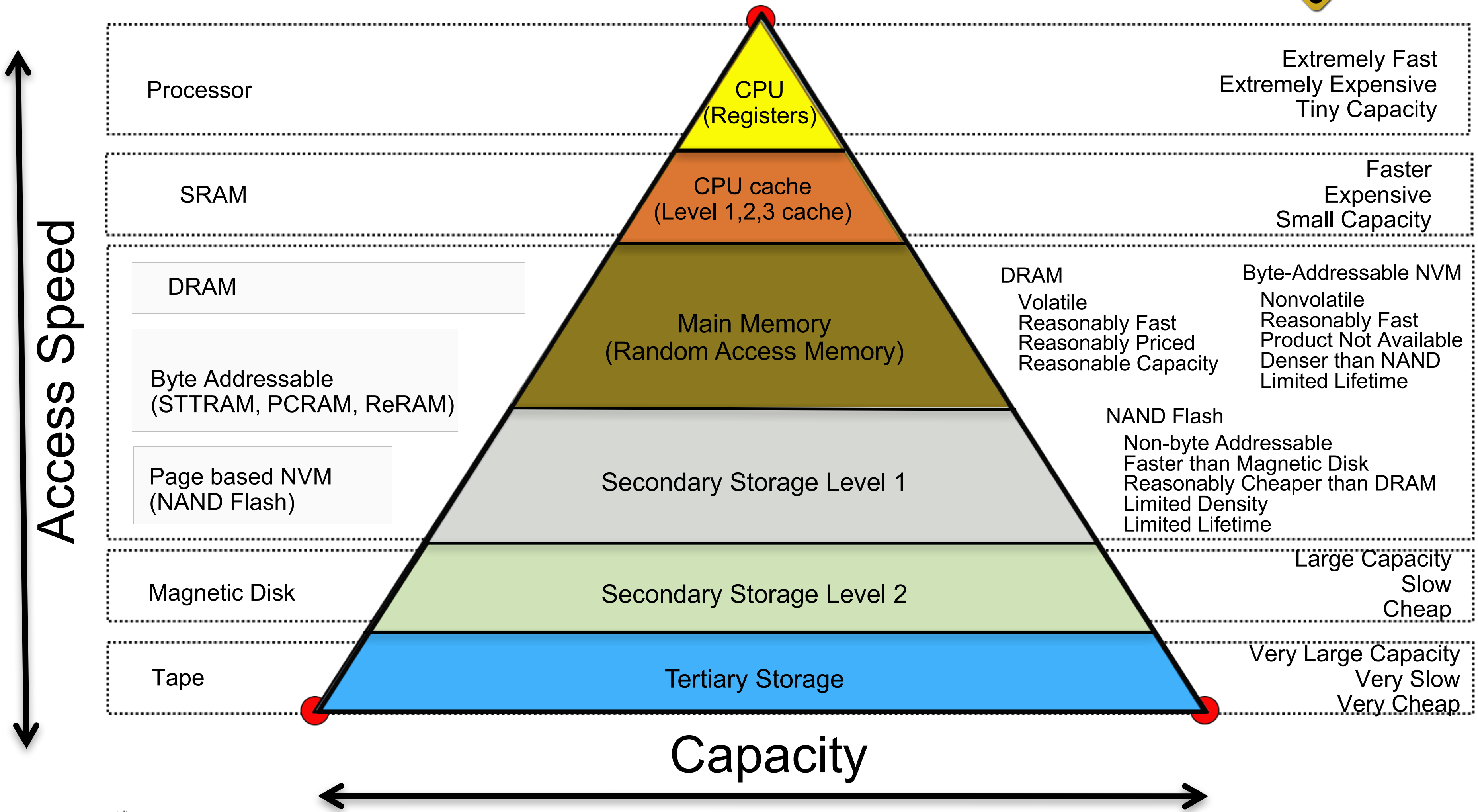


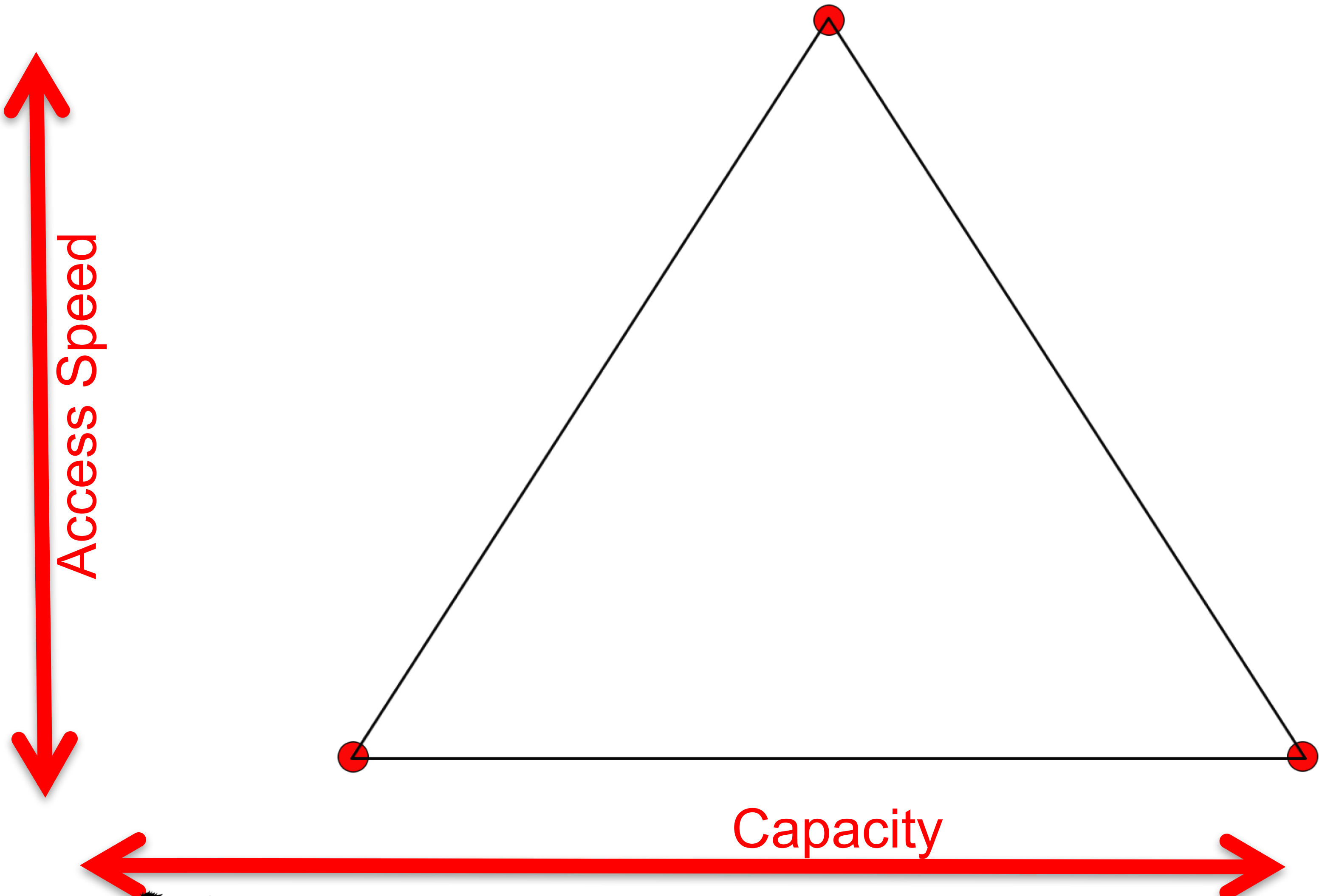
Fig 1: An early memory hierarchy.



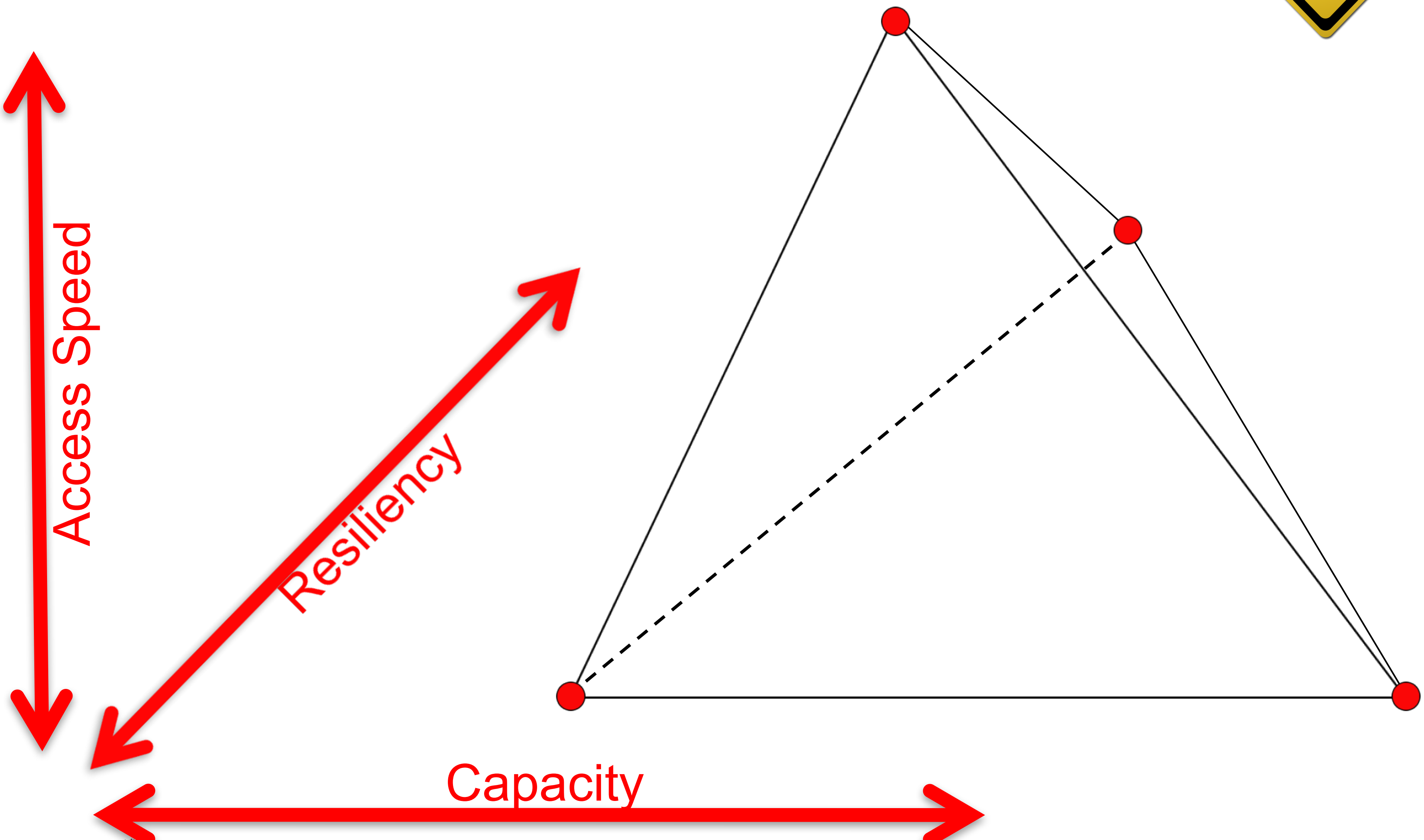
Exascale and beyond promises to continue the trend towards complexity



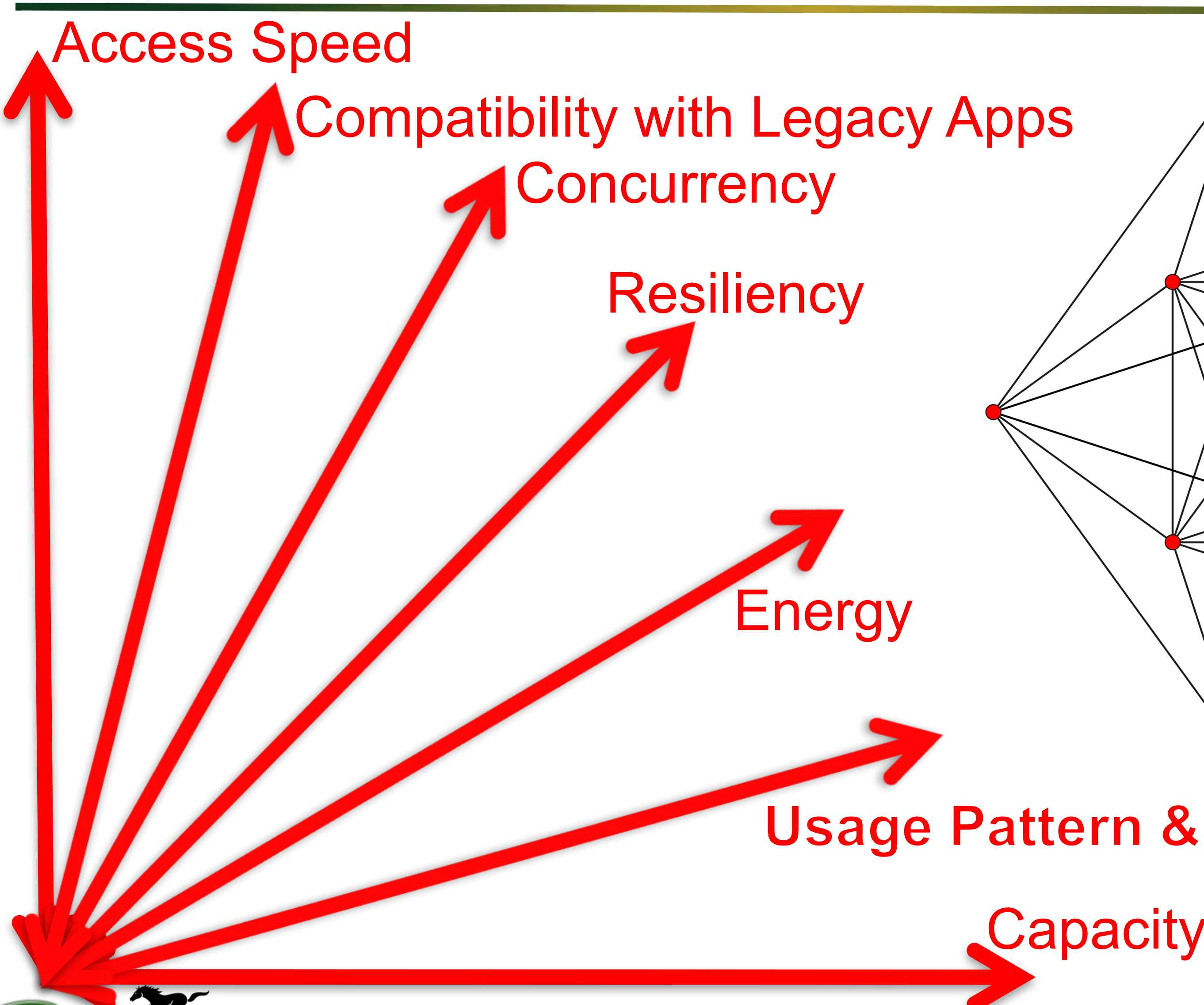
The Traditional Dimensions…

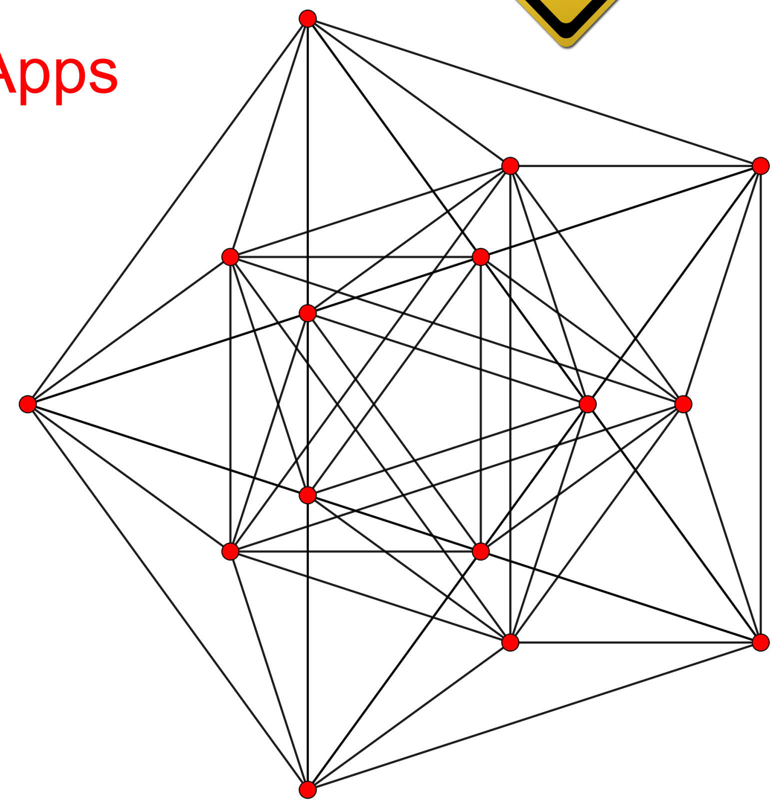


The Traditional Dimensions are Being Expanded…

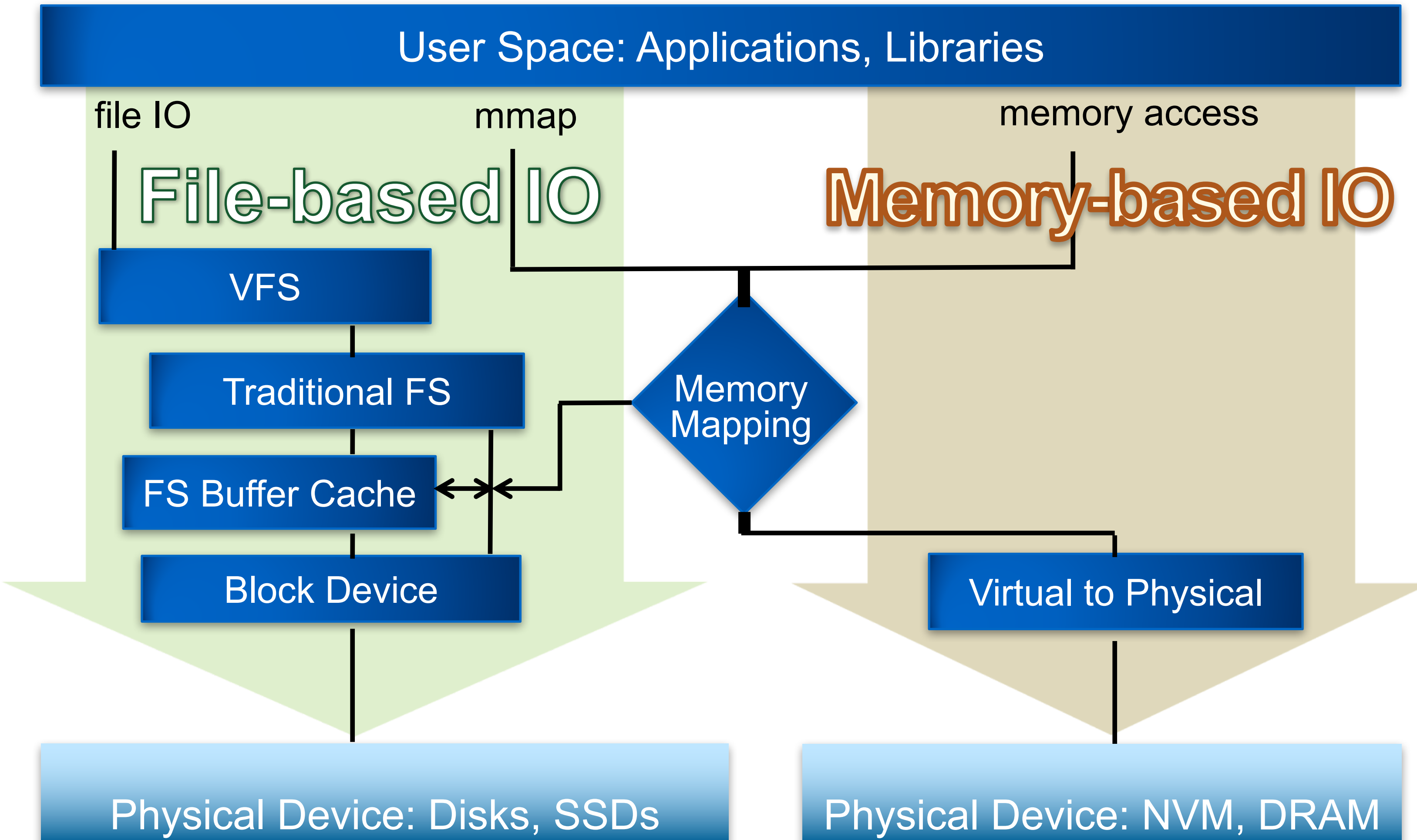


The Traditional Dimensions Are Being Expanded Into *future directions*



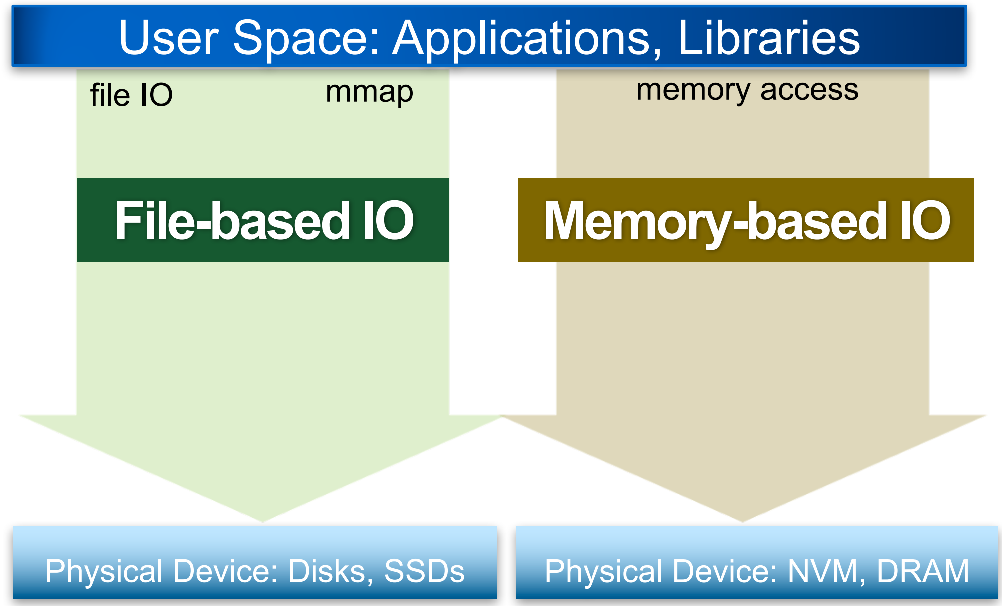


...But The Exposed Path to Our Memory Hierarchy Remains Bifurcated



Implications

- Complexities in managing power and resilience when actions can be taken independently down the two paths
- Results in application factoring in multiple data layouts for different architectural reasons



- Computers are good at handling the details dynamically
- Burst buffers, new memory layers, concurrency, power and resilience make data placement difficult for domain scientists.

How would A unified Path Work?

UNITY Architecture

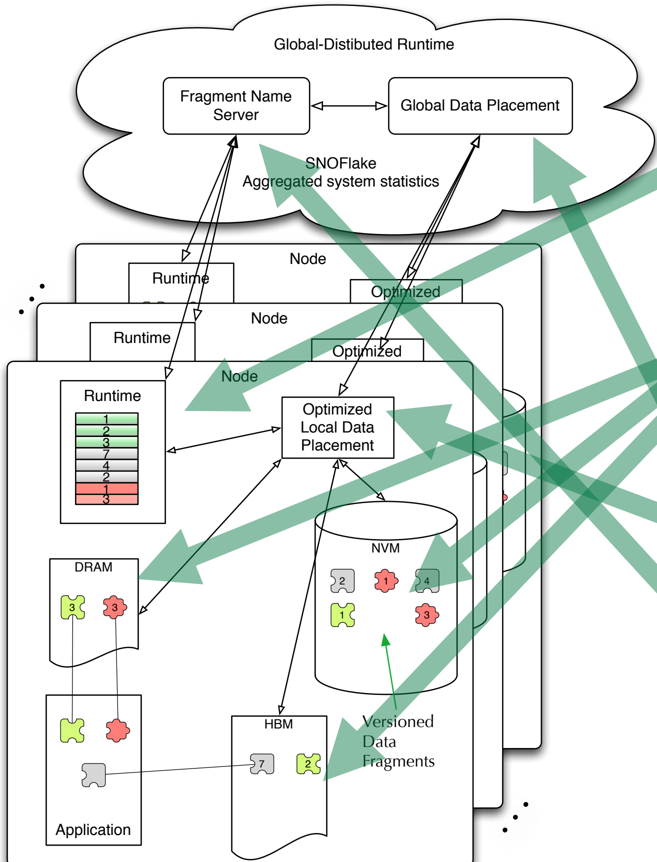


Fig 1: The UNITY architecture is designed for an environment that is (a) prescriptive; (b) distributed; (c) dynamic; (d) cooperative.

Local node runtime

- Persistent deamon to handle subsequent accesses
- Also performs post-job security cleanup

"Dynamic" components

- Active throughout life of application
- Able to adjust strategies
- Incorporates COW optimizations

Local & Global optimizers

- Directs data placement
- Global considers collective & machine status optimizations

Nameserver for metadata management

- Efficiently describes data mappings
- Keeps track of published objects
- Persistent daemon at well known address

UNITY Design Objectives

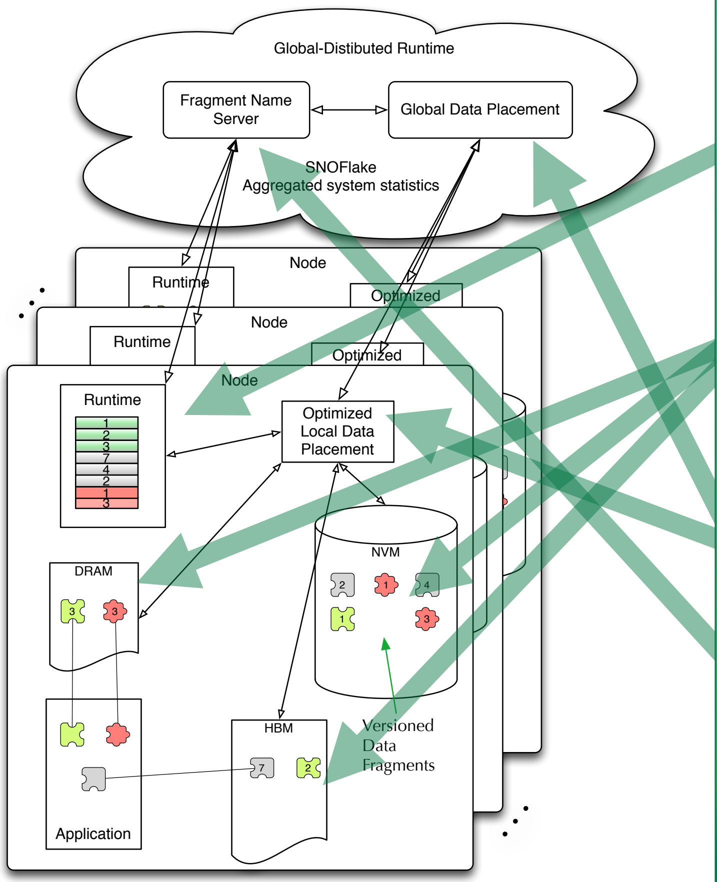
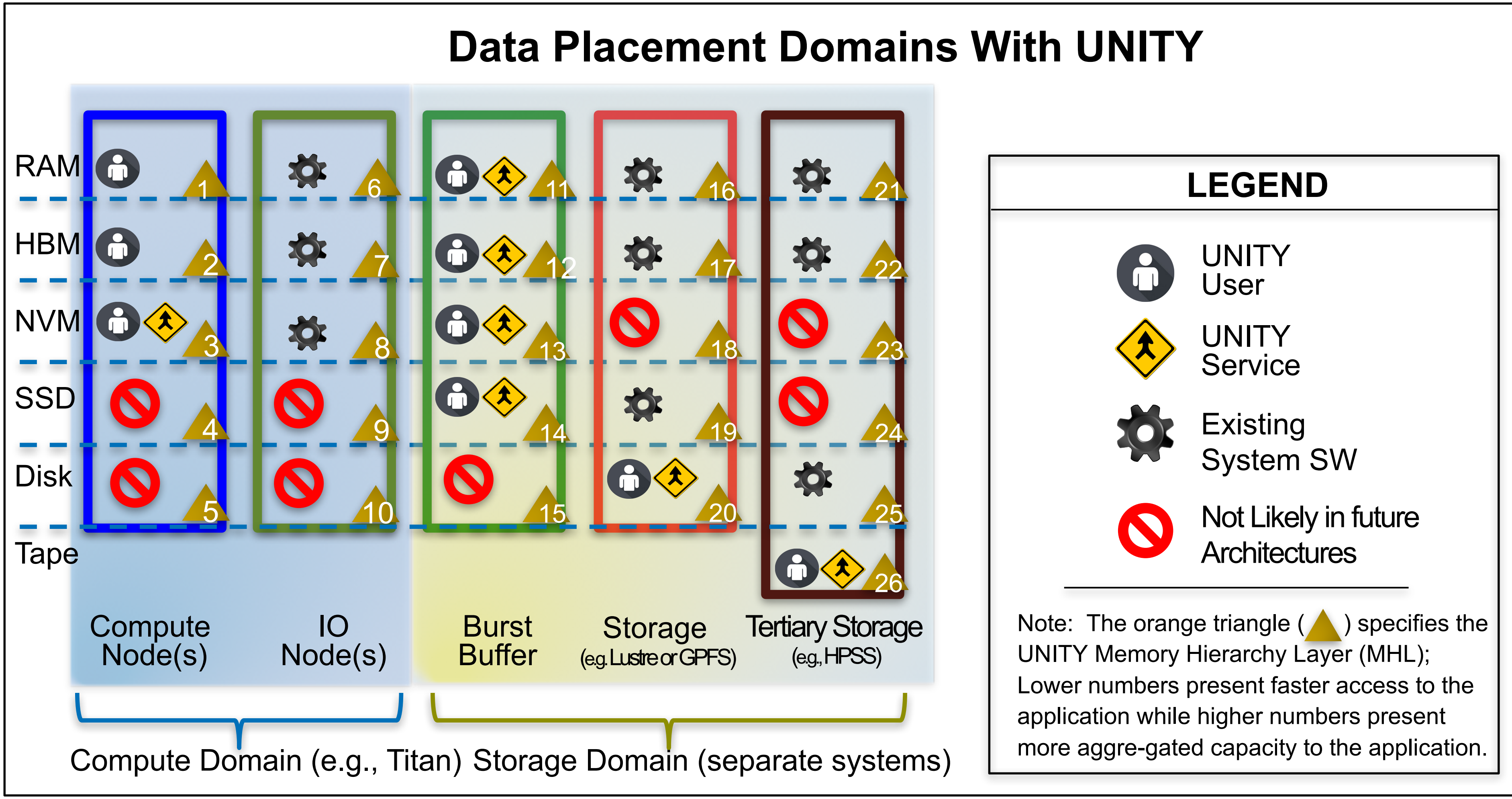


Fig 1: The UNITY architecture is designed for an environment that is (a) prescriptive; (b) distributed; (c) dynamic; (d) cooperative.

A unified data environment based on a smart runtime system:

1. frees applications from the complexity of directly placing and moving data within multi-tier storage hierarchies,
2. while still meeting application-prescribed requirements for data access performance, efficient data sharing, and data durability.

Automated movement with Unity



Providing a Prescriptive API

```
unity_create_object("a", objDescr);
workingset = unity_attach_fragment(workFrag, flags);
xghost = unity_attach_fragment(ghosttop, flags);
yghost = unity_attach_fragment(ghostleft, flags);

for (x=0; x<1000; x++) {
    if ( x>0 ) {
                reattach(xghost, x);
                reattach(yghost, x);
    }
    // do work
    unity_publish(workingset);
}
```

Fig 1: The UNITY API is designed to be extensible and flexible – here is an example with meshes & ghost cells.

Scientific Achievement

A new API enables domain scientists to describe how their data is to be used. This permits a smart runtime system to do the tedious work of managing data placement and movement.

Significance & Impact

Vendors are providing multiple APIs to deal with their novel abilities. Through our co-design oriented project, we provide a unifying way to achieve what the domain scientists want in a machine independent way.

Research Details

Currently we have a functional prototype that provides most of the functionality that will be visible to the application developers using the runtime. We have created a global name service that can be used to query datasets and where their data is located. We have created a runtime service that runs on each node and keeps track of the data available locally on the node. The runtime, in conjunction with the global name server create a distributed data store that can utilize both volatile and nonvolatile memory available on the supercomputer. We have designed and implemented hooks in the system, so intelligent data placement services can identify and update the data location and format in order to improve the overall performance. We have modified the SNAP proxy application to use the Unity's API. We can checkpoint and restart the application and can demonstrate the advantages of Unity by checkpointing a N-rank SNAP job and restarting it as a M-rank job. Datasets/checkpoints can be pulled from multiple hosts over TCP or Infiniband.

First – Do No Harm

Scientific Achievement

Demonstration of interface and architecture with SNAP.

Significance & Impact

Distributed systems permit needed functionality like persistent name spaces across run invocations and across an entire machine. However, they also require careful design to avoid costly overheads.

Research Details

A "worse-case" scenario for our design is an important test to determine if the idea is simply not feasible. We were able to validate that the overheads associated with UNITY are not prohibitive even without our "local placement" engine or "global placement" engine optimizations.

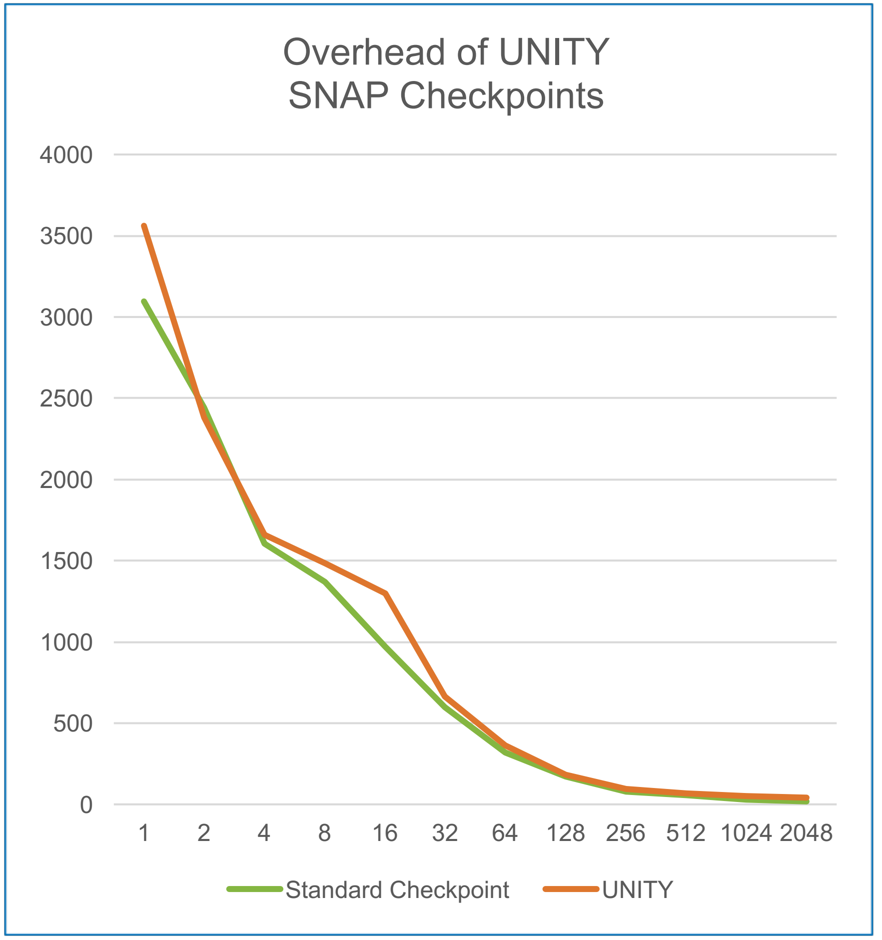


Figure 1: Reported times in seconds; results are averages of three runs.

Smarter about Energy Consumption

Scientific Achievement

UNITY improves energy consumption by intelligently incorporating an overview of data and thereby removing unnecessary overheads.

Significance & Impact

- In next generation machines, standard DRAM and stacked memory are expected to consume ~85% of system energy.
- Different policies result in dramatic differences in performance and energy usage, but existing interfaces provide little support for policy choice. We show that simply pre-copying data results in increased energy with mixed results

Research Details

- Minimized software stack overheads: shorter I/O path via API design provides improved performance. Eliminate NVRAM block device overhead due to file system and a lengthy I/O path.

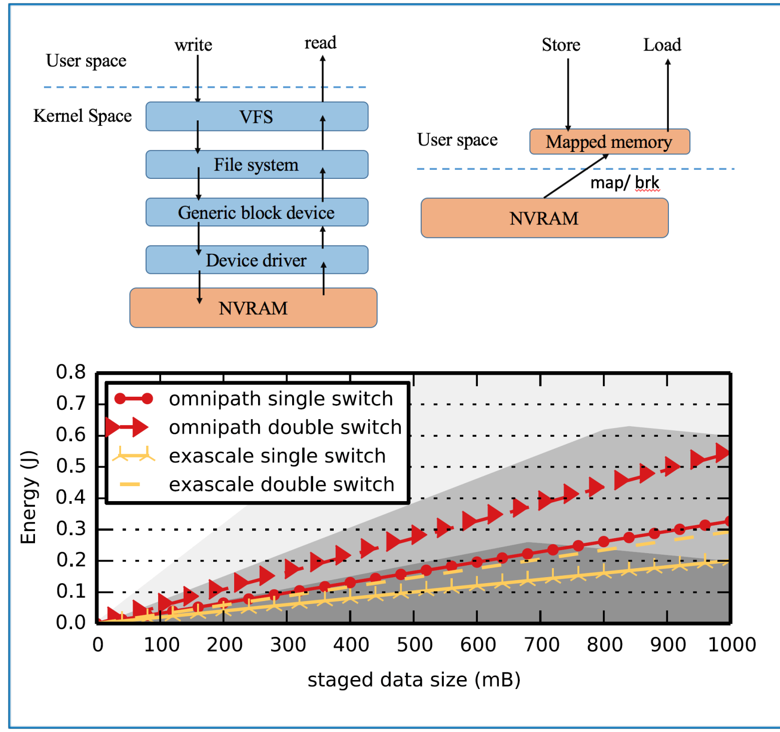


Figure 1: energy costs of checkpoint strategies

- UNITY PHX evaluations on real-world HPC applications and emulated NVRAM hardware shows up to around 12x speedups in checkpoint times over the naïve NVRAM data copy method. We believe UNITY PHX's performance gains will be even more significant on a larger scales where the C/R costs are even more greater.

Smarter About Placement Decisions

Scientific Achievement

UNITY provides a new advanced multi-level caching capability.

Significance & Impact

Caching automatically triggers costly data movement, but there are many applications that operate just as fast out of slower memory, such as those with 'stream-based' access patterns able to fully take advantage of built-in last level caches.

Research Details

Much memory is touched only once or rarely, but with caching, any such access results in data movement from slower to faster memory. This will quickly consume sparse 'fast' memory resources. Allocation and movement informed by application level hints is preferable. Our system uses a combination of user-directives and recent history to improve data placement.

> "There are only two hard things in Computer Science: cache invalidation and naming things."
>
> ~ Phil Karlton (Xerox PARC & Netscape)

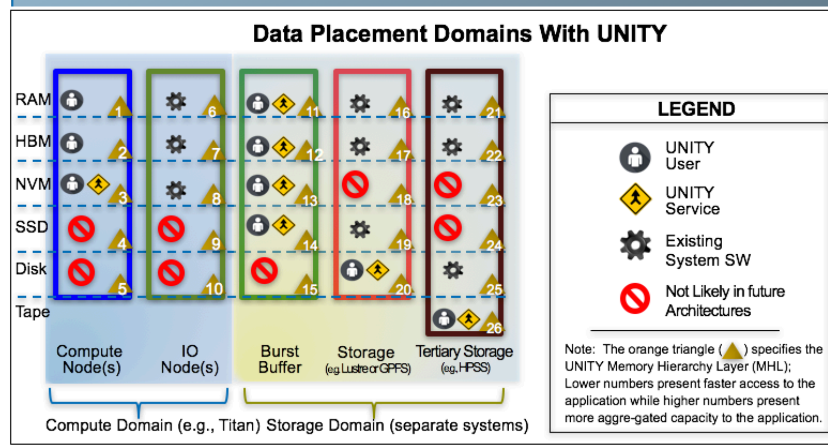


Figure 1: UNITY automatically migrates & and manages complex distributed data hierarchies.

Smarter About Resiliency and C/R

Scientific Achievement

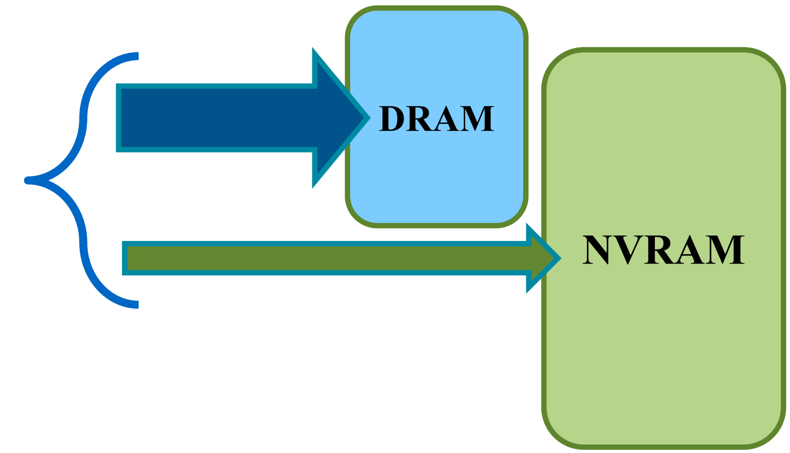
UNITY automatically supports aggregate-bandwidth checkpoints through concurrently accessing DRAM and NVRAM.

Significance & Impact

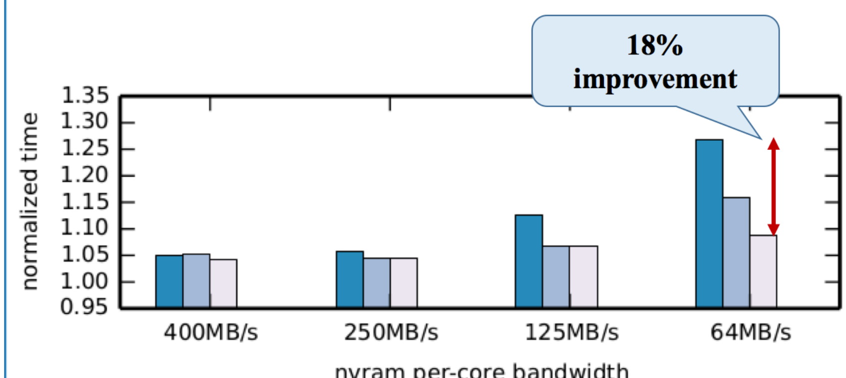
Checkpoint/Restart has become an important component to HPC resiliency. In the future, more I/O is expected (more cores per node implies more data).

Research Details

- NVRAM provides denser persistent memory, but has limited bandwidth. RAID like structures could possibly address bandwidth, but at the expense of energy.
- DRAM has superior bandwidth compared to NVRAMs (4x-8x).
- UNITY accelerates critical path data movement with bandwidth aggregation.



Improved bandwidth => improved performance
Improved performance => improved power usage



UNITY Summary

- OBJECTIVE: design and evaluate a new distributed storage paradigm that unifies the traditionally distinct application views of memory- and file-based data storage into a single scalable and resilient environment.
- DRIVERS:
  - Reducing complexity for Applications
  - Unprecedented concurrency (across nodes & within nodes)
  - Increasing complexity and tiers for memory & storage
  - Limited power budgets
  - Number of components raises concern over hardware faults
- WHAT DIFFERENCE WILL IT MAKE: effectively balance data consistency, data resilience, and power efficiency for a diverse set of science workloads,
  - First, provide explicit application-level semantics for data sharing and persistence among large collections of concurrent data users.
  - Second, intelligently manage data placement, movement, and durability within multi-tier memory and storage systems.

Acknowledgements

Funding for UNITY provided by DOE/ASCR under the SSIO program, program manager Lucy Nowell.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

The Unity Team is:

Terry Jones[1], Mike Lang[2], Ada Gavrilovska[3], Latchesar Ionkov[2], Douglass Otstott[2], Mike Brim[1], Geoffroy Vallee[1], Benjamin Mayer[1], Aaron Welch[1], Mason Watson[1], Greg Eisenhauer[3], Thaleia Doudali[3], Pradeep Fernando[3]

[1]Oak Ridge National Lab
Mailstop 5164
Oak Ridge, TN 37831

[2]Los Alamos National Lab
PO Box 1663
Los Alamos, NM

[3]Georgia Tech University
266 Ferst Drive
Atlanta, GA 30332