Future HPC Data Architectures: Trying to Bring Definition to Blurring Lines

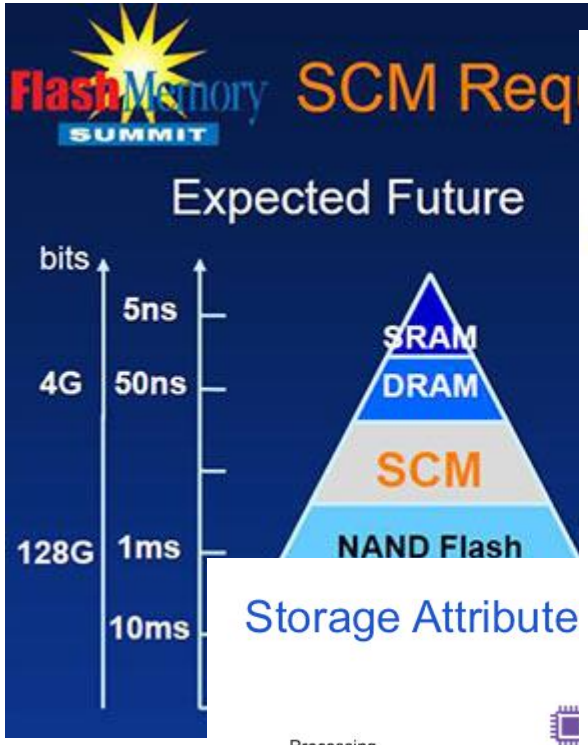
Dan Ernst, PhD
Advanced Technology
SSIO - 9/20/18

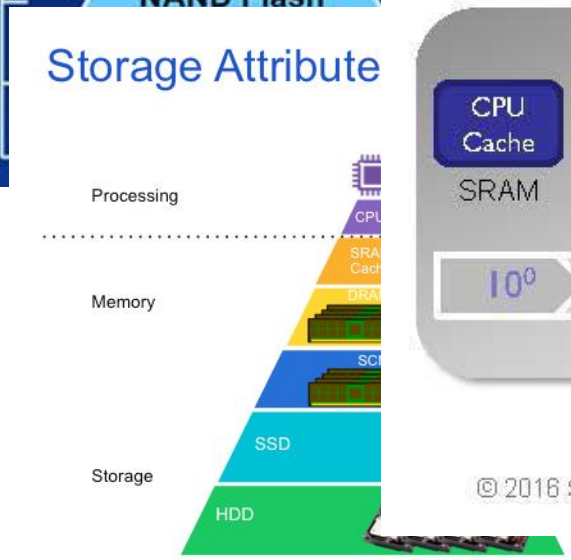
Some Starting Context

- I'm not a storage person
- I cover node architecture for Cray
- One particular focus has been memory
  - Both long-term and day-to-day
- Memory led me to some storage (media) stuff
- Memory and storage are the same thing?

Generic Hierarchy Problem Statement Slide

REIMAGINING THE DATA CENTER

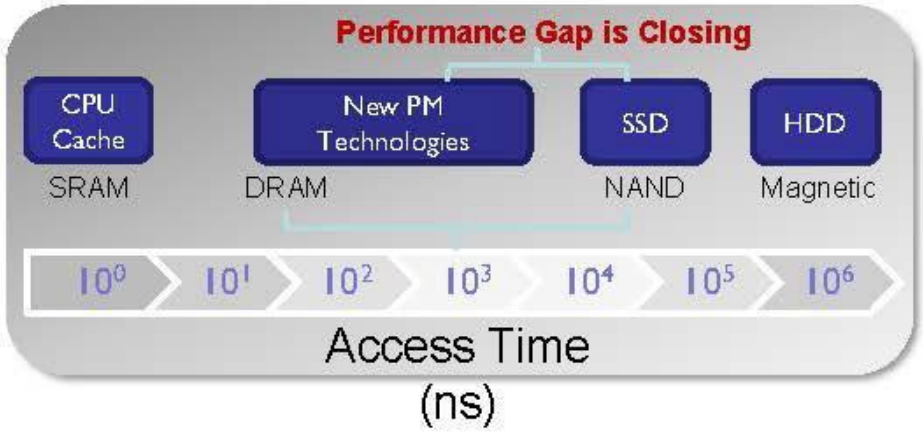


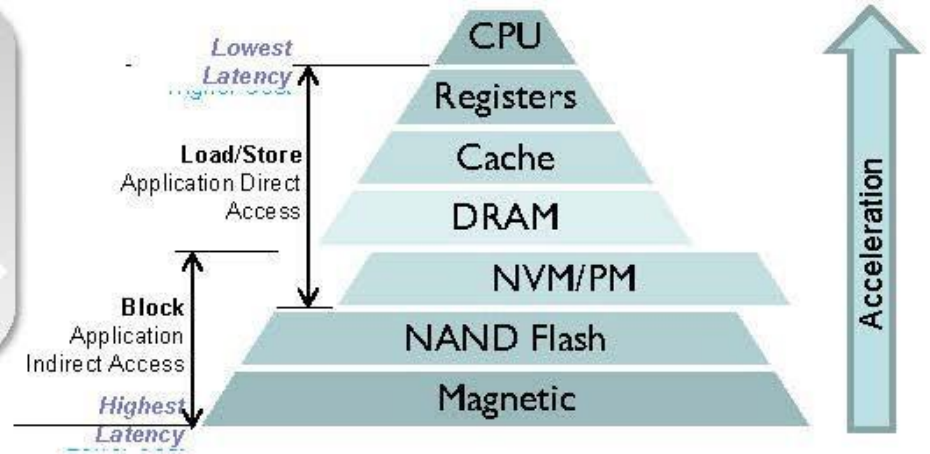


Memory – Storage Hierarchy

SNIA SSSI | SOLID STATE STORAGE

- Data-intensive applications need fast access to storage
- Persistent memory is the ultimate high-performance storage tier
- NVDIMMs have emerged as a practical next-step for boosting performance



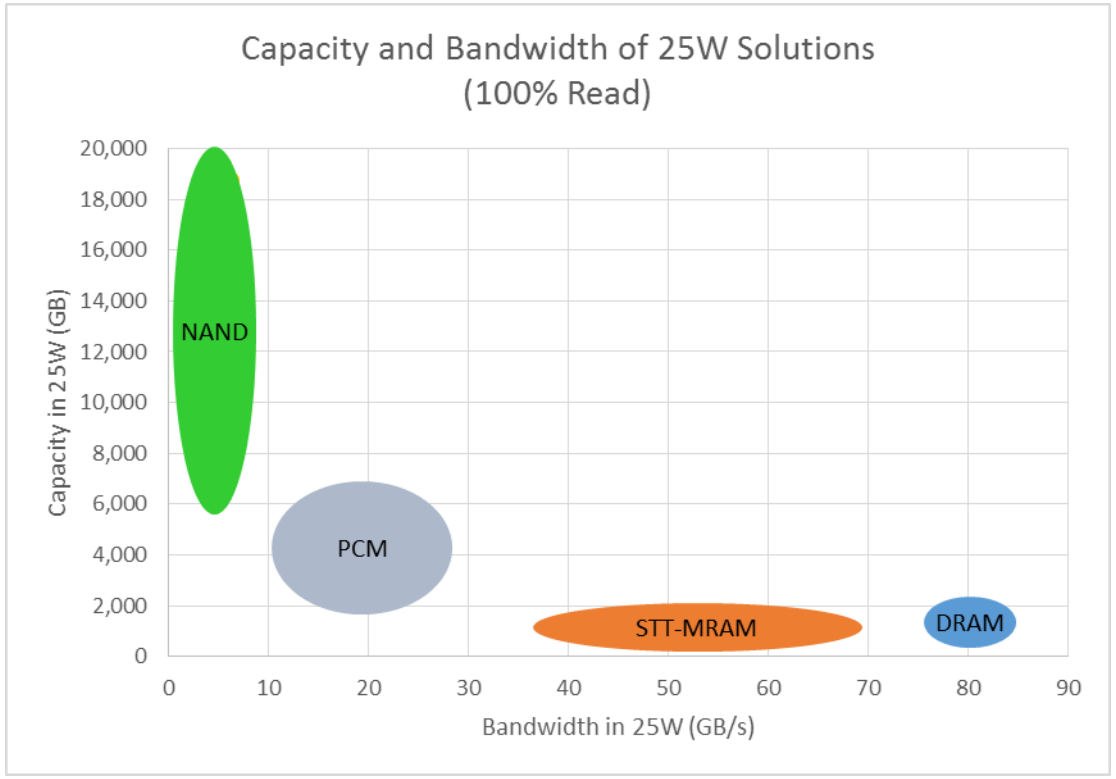


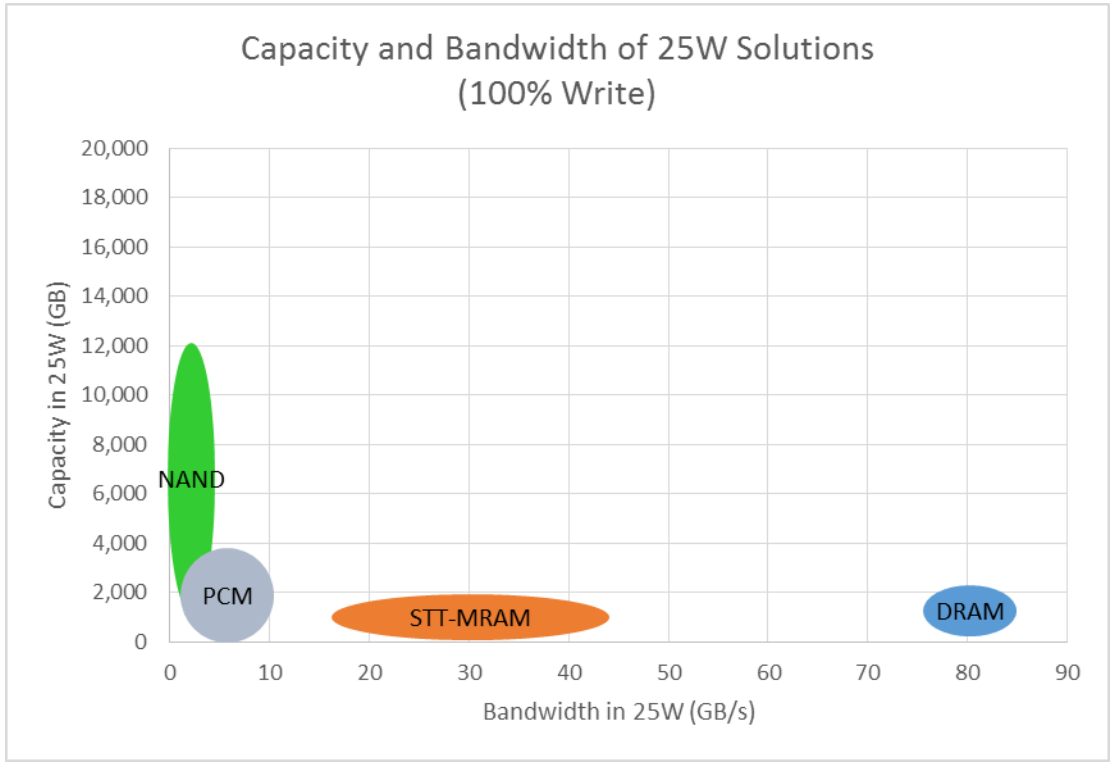
Source: HPE/SNIA 2015

(with optimized software)

Node-Side Memory: What Do We Know?

Memories Have Parameters

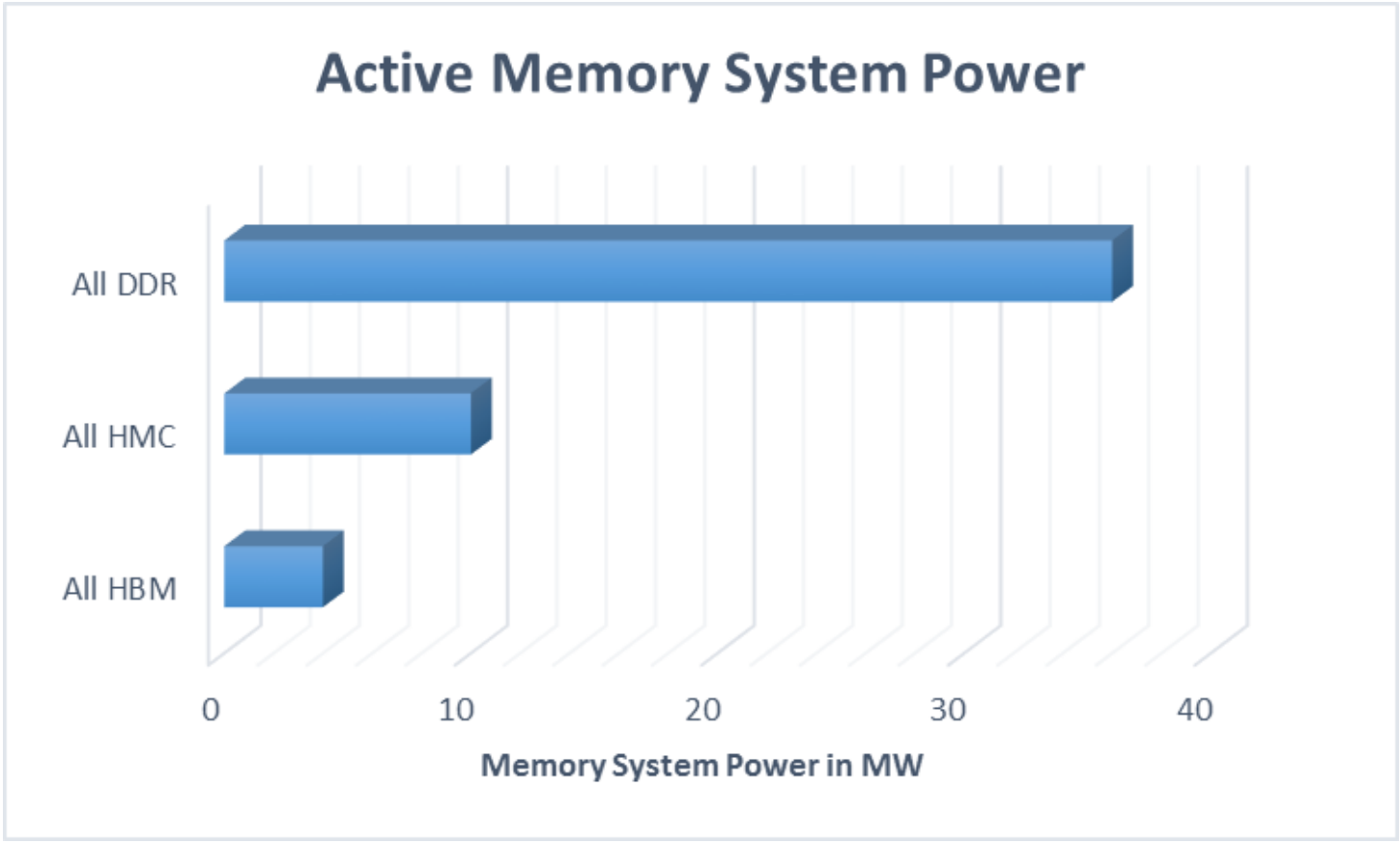




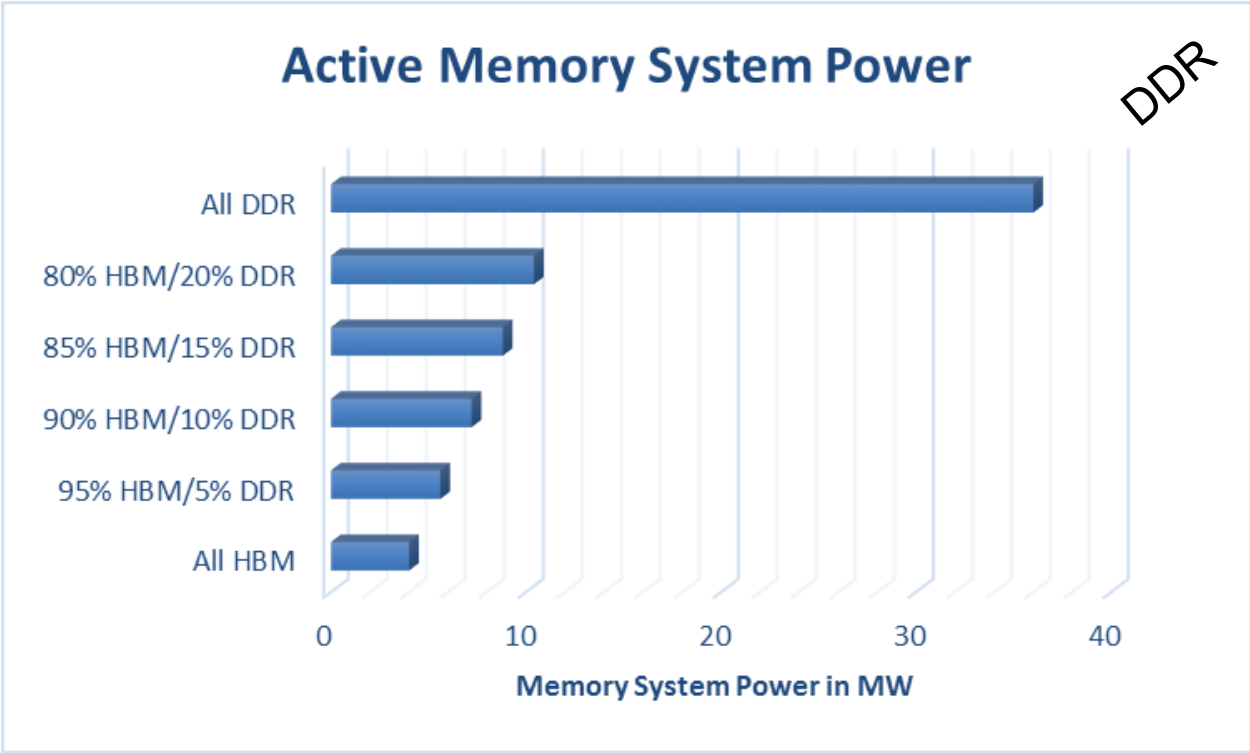
- **Insight: Most NVM media have near-zero idle power, but are very power-hungry when you actually use them**
  - Especially for writes
- **Insight #2: This isn't *really* a media (cell) technology chart**

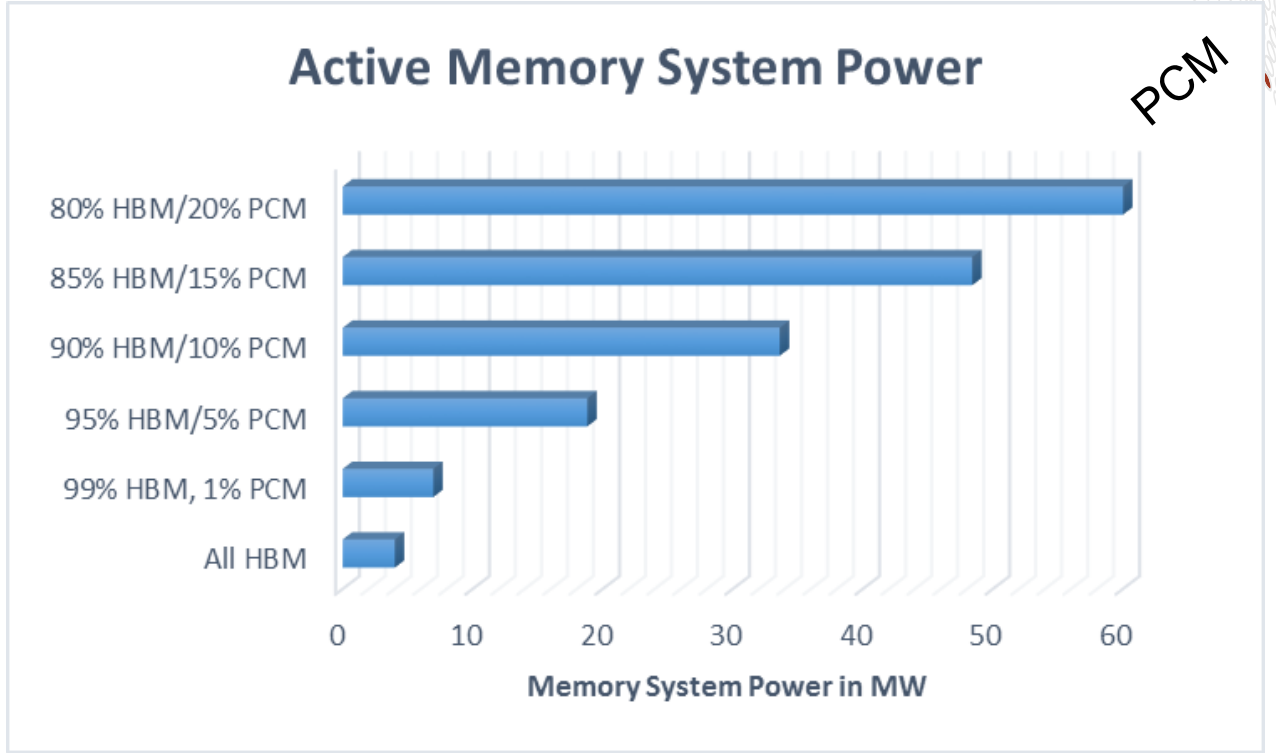
Memory System Design Space (System Level)

- 1 EF system with 0.2 Byte/s per Flop bandwidth



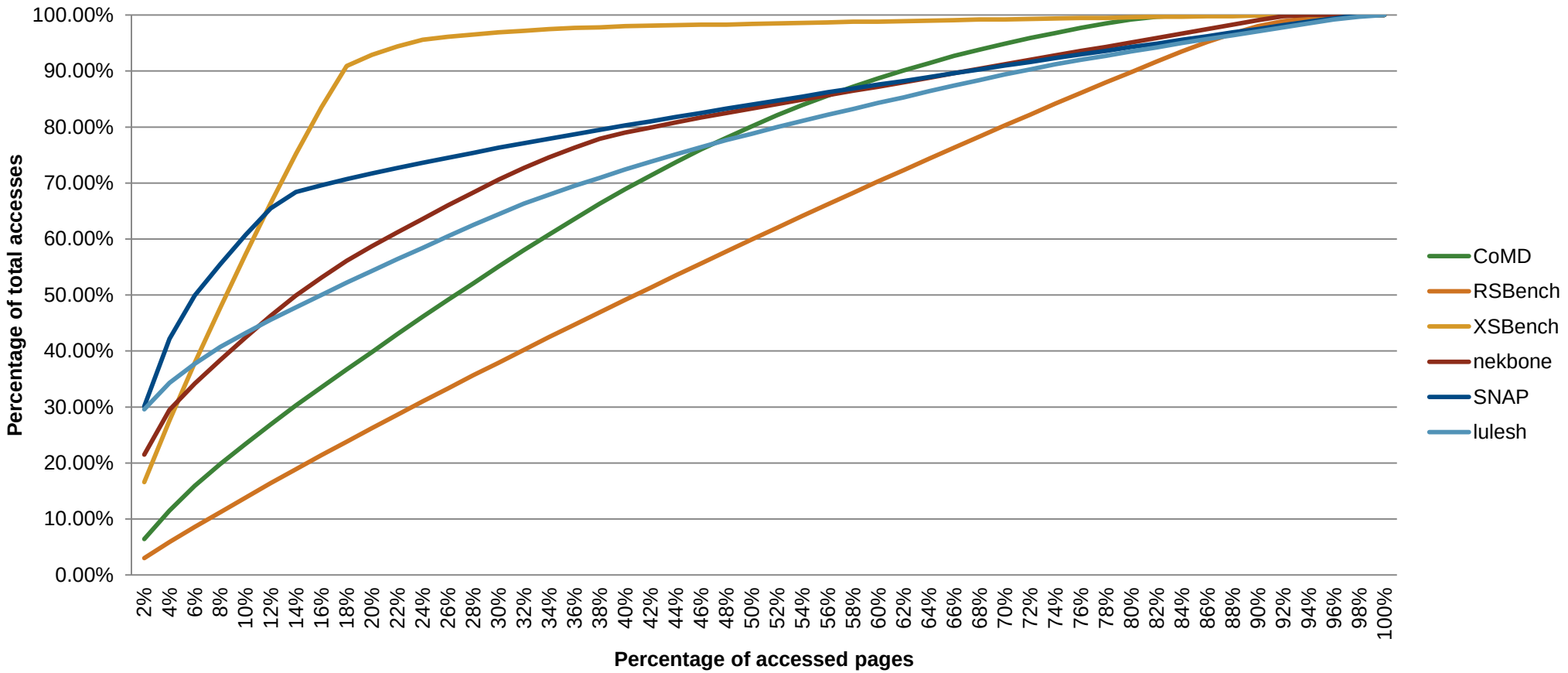
Bandwidth Allocation Boundaries





- **Insight: HBM:DDR:PCM bandwidths likely to have 100:10:1 ratio**
  - Likely better in the short term, but configurations will eventually be power constrained for Exascale

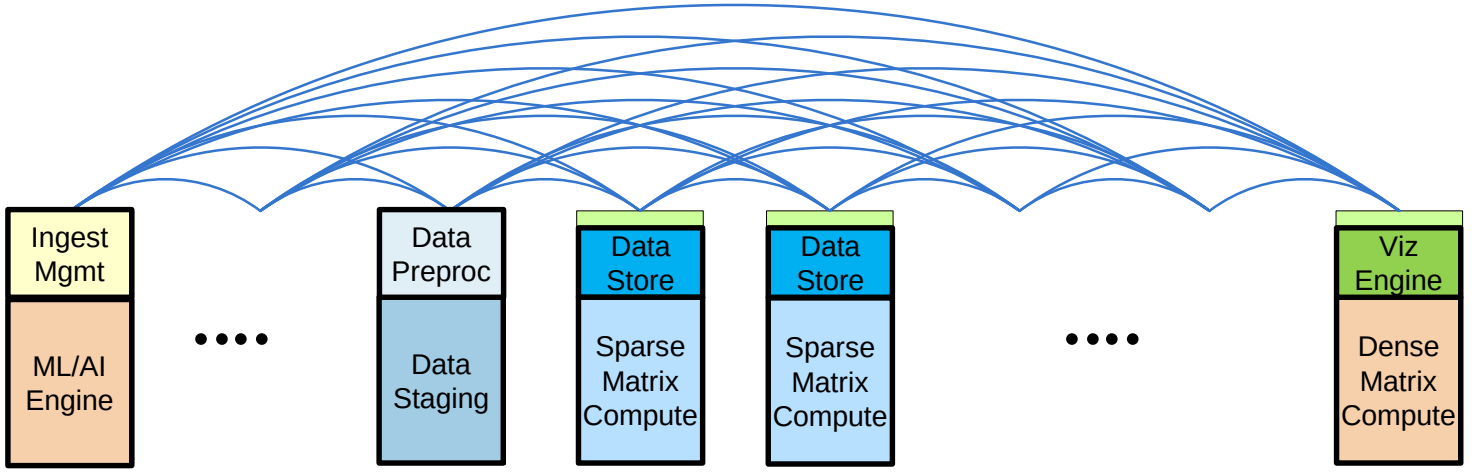
FF2 Study of Access Density



- **Insight: Without thoughtful staging/streaming, ratio > 10:1 will not perform**
- **Corollary: SCM is unlikely to be usable with existing memory use cases**

Unified Heterogeneous Systems

- In an era of specialization:
  a diverse user base
  → diverse applications
  → diverse requirements
- Also rising use of diverse *workflows*
- *Data interchange* becomes crucial component

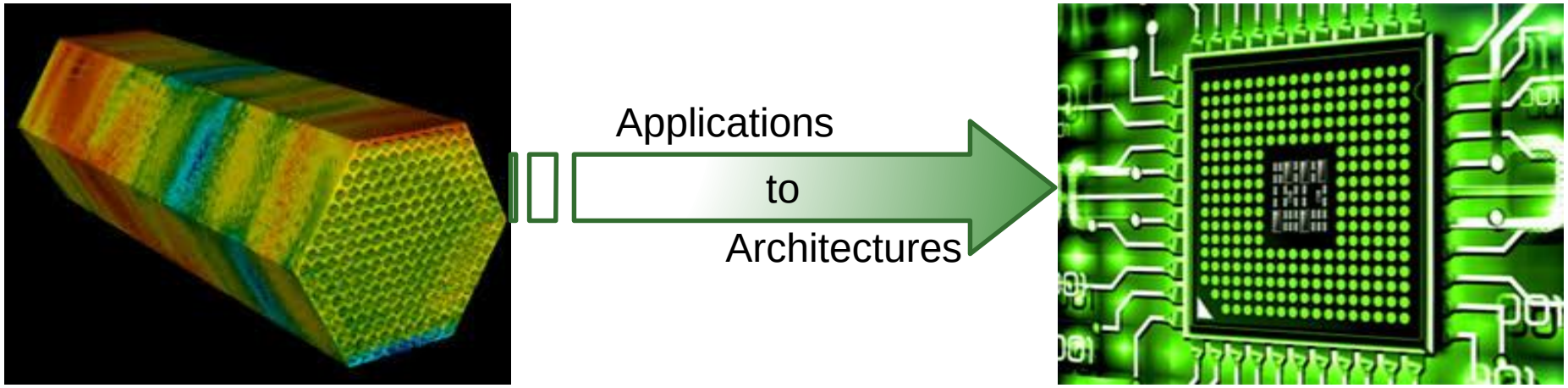


Data should be globally visible
→Not locked to a node
→Persists through jobs
→Some Guarantees

Analysis of Applications

To reach the goal of producing architectures well-suited to HPC applications…

… you must <u>understand the applications</u>

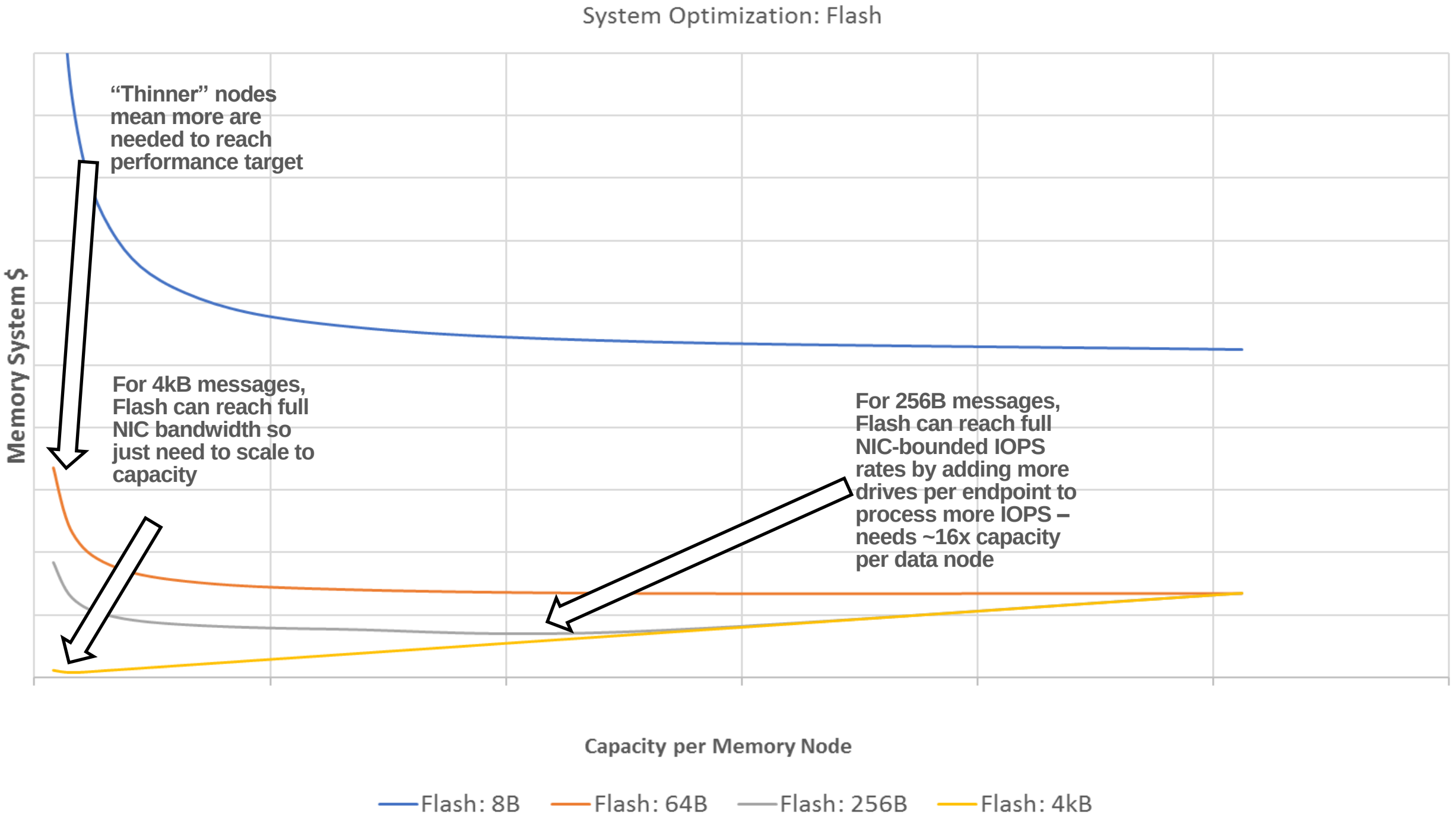


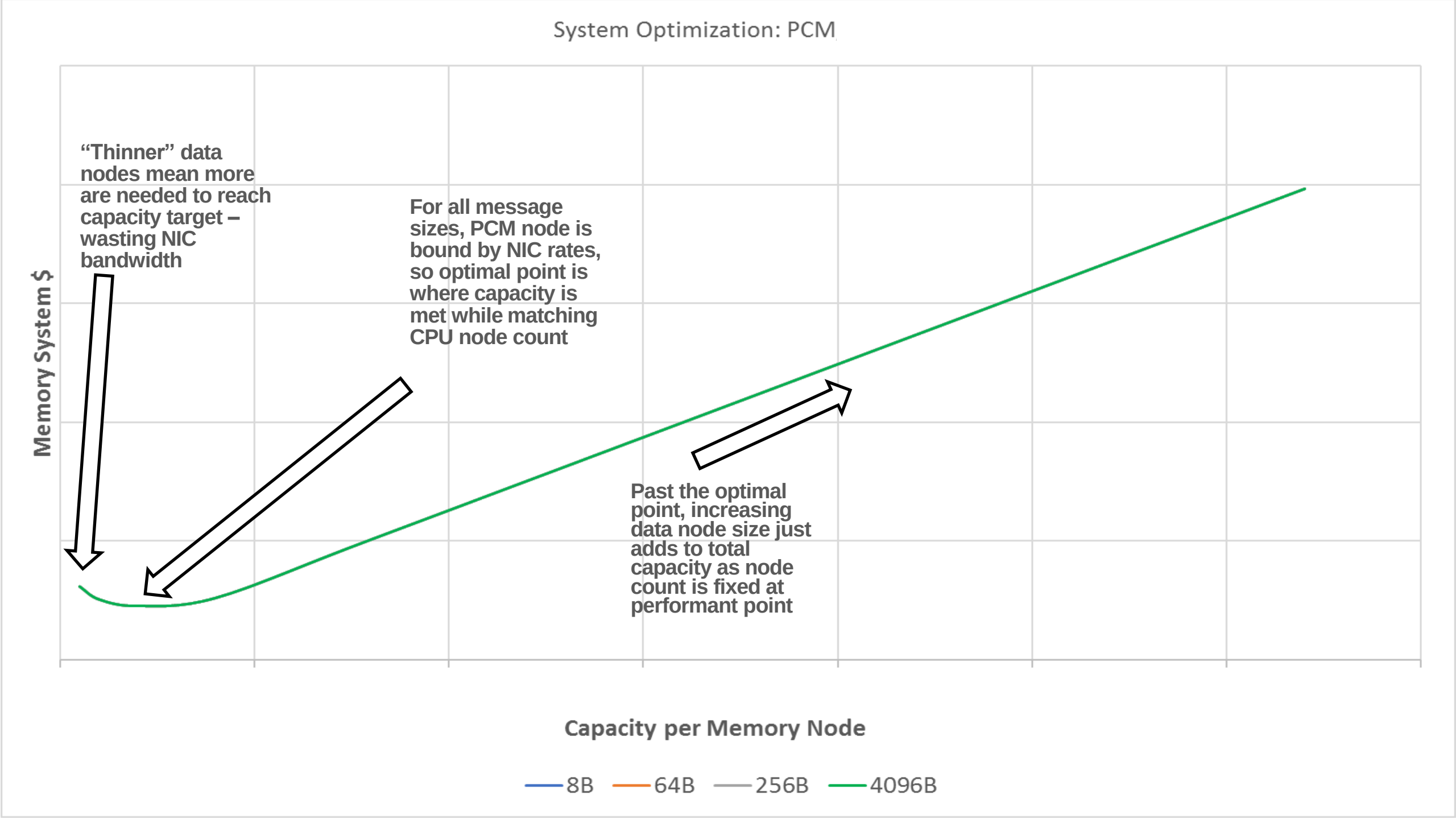
An Example Application Framework

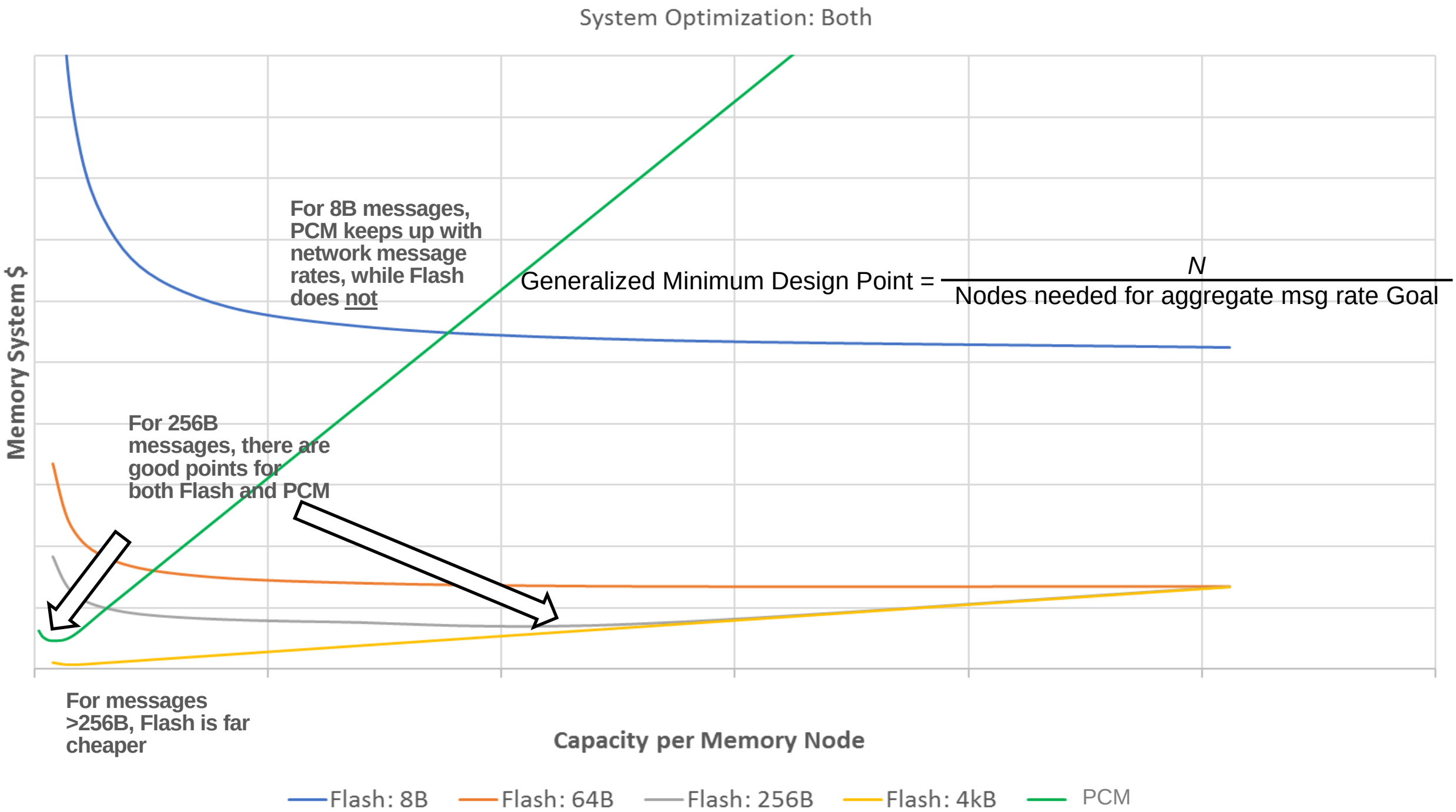
- **Assume *X* compute nodes**
- **Assume network (like Cray Aries) with good performance on small msgs**
- **Assume uniform random access to data**
  - But with varying object sizes

- **Goal:**
  - Capacity of *N* bytes of global Persistent data of whatever media type, AND…
  - Reach required total data bandwidth to match aggregate compute injection bandwidth
    - This number is adjusted based on supported network message rates

- **Let's explore persistent data configurations to meet this**
  - Calculate "reasonable" internal media bandwidths on "memory nodes" at minimum capacities
  - Scale capacity per endpoint to explore different balances


System Optimization: Flash
"Thinner" nodes mean more are needed to reach performance target
For 4kB messages, Flash can reach full NIC bandwidth so just need to scale to capacity
For 256B messages, Flash can reach full NIC-bounded IOPS rates by adding more drives per endpoint to process more IOPS – needs ~16x capacity per data node
Memory System $
Capacity per Memory Node
Flash: 8B
Flash: 64B
Flash: 256B
Flash: 4kB


System Optimization: PCM
"Thinner" data nodes mean more are needed to reach capacity target – wasting NIC bandwidth
For all message sizes, PCM node is bound by NIC rates, so optimal point is where capacity is met while matching CPU node count
Past the optimal point, increasing data node size just adds to total capacity as node count is fixed at performant point
Memory System $
Capacity per Memory Node
8B
64B
256B
4096B


System Optimization: Both
Memory System $
For 8B messages, PCM keeps up with network message rates, while Flash does not
Generalized Minimum Design Point = N / Nodes needed for aggregate msg rate Goal
For 256B messages, there are good points for both Flash and PCM
For messages >256B, Flash is far cheaper
Capacity per Memory Node
Flash: 8B
Flash: 64B
Flash: 256B
Flash: 4kB
PCM

The user interface to these should be the same!

"Put my data in the store"
"It should still be there when I spin up my next jobs"
"The software cost to accomplish that shouldn't destroy the utility"

Takeaways

1. **Memory and Storage are often drawn as triangles**
   - If your goal for adopting SCM is to fill in your triangle, you shouldn't be in charge of anything
2. **For existing scientific apps, direct access to on-node SCM is worth little at scale**
   - SCM seems best shared on the network (*at least logically*)
3. **The issue in point #2 is limited to existing simulation and does not mean it does not have a use case**
   - Lowest-hanging fruit is probably workflow-related
4. **Decoupling remote persistent memory from compute nodes has value**
   - Upward evolution of parallel FS, but without the baggage, please
5. **New memories can be arranged in a diverse set of configurations**
   - Implies that software interface architectures should be as media agnostic as possible
6. **None of those configurations provides a free lunch**
   - Specific application/workflow wins should be the target
   - Know your data patterns!

Thank You!

dje@cray.com