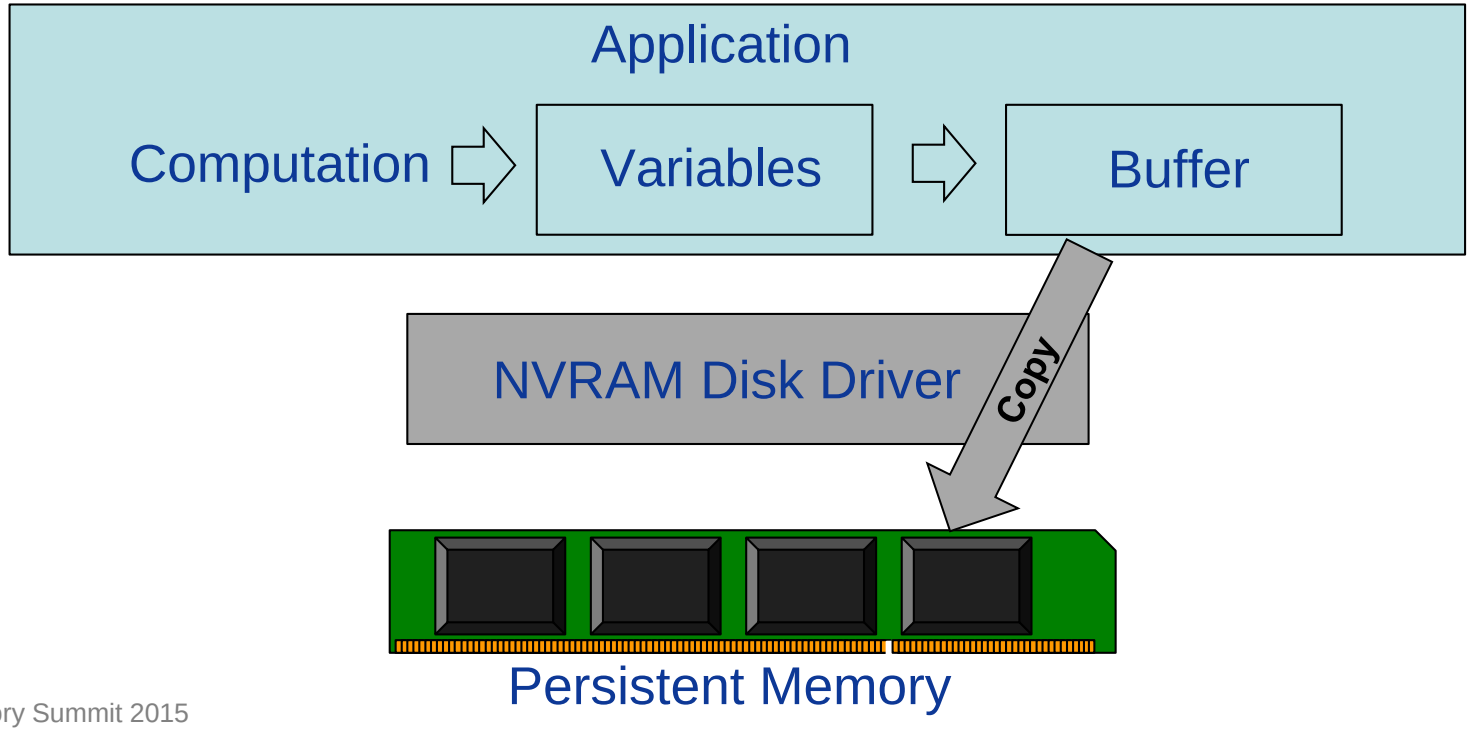
How Systems Benefit from Record Low Latencies

Doug Voigt,
Hewlett Packard (Enterprise)

Contents

- Application views of persistent memory
- Benefit of RAM disk access
- Benefit of Ld/St access
- Benefit of disaggregated persistent memory

Persistent memory as NVRAM Disk

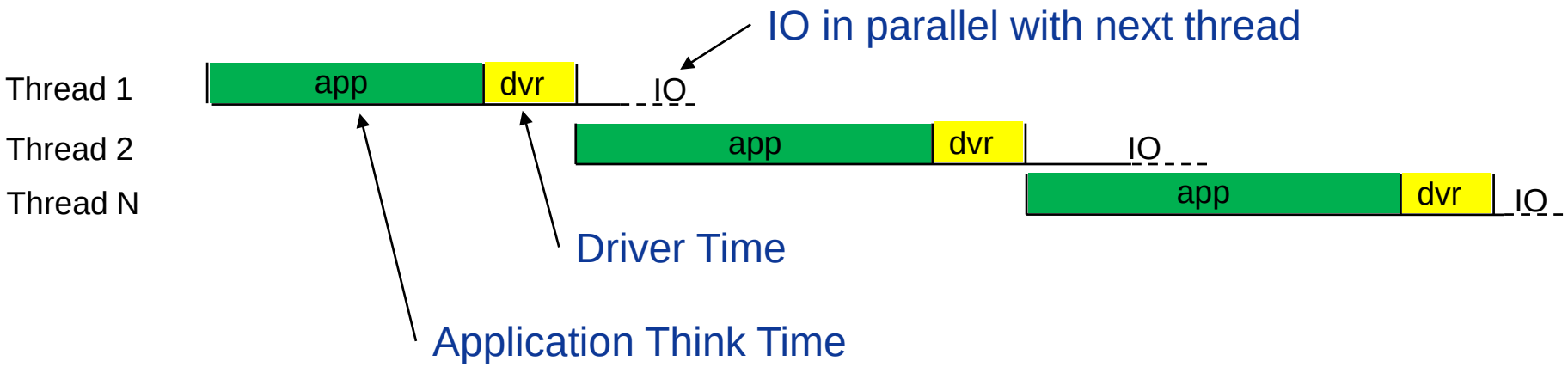


Ideal NVRAM Disk Access Efficiency

- Compare disk array access with PM access
- Input current disk array benchmark results
  - Throughput
  - Response time
  - Thread count
- Derive application think time per IO for various CPU utilizations
  - Assume constant driver overhead
  - Use above inputs to compute application time in each thread model case
  - Vary Demand (Gby/Hr), CPU Utilization
- Compare resulting CPU utilization deltas
  - CPU bound proxy
  - IO Bound proxy

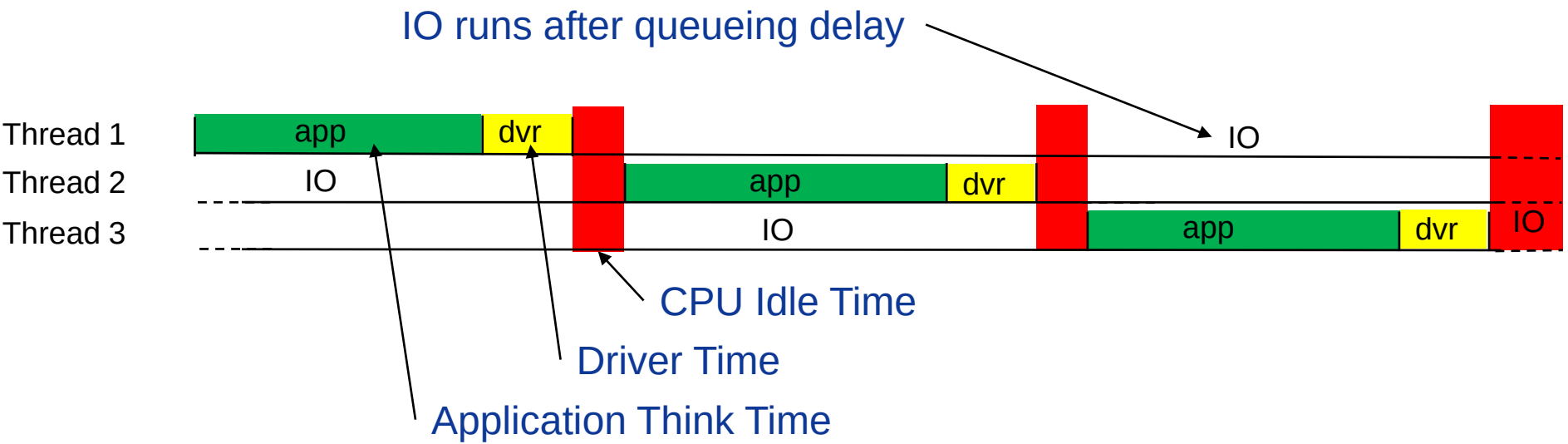
CPU Bound Use Case: Open Loop IO

CPU time = app + dvr, unlimited threads



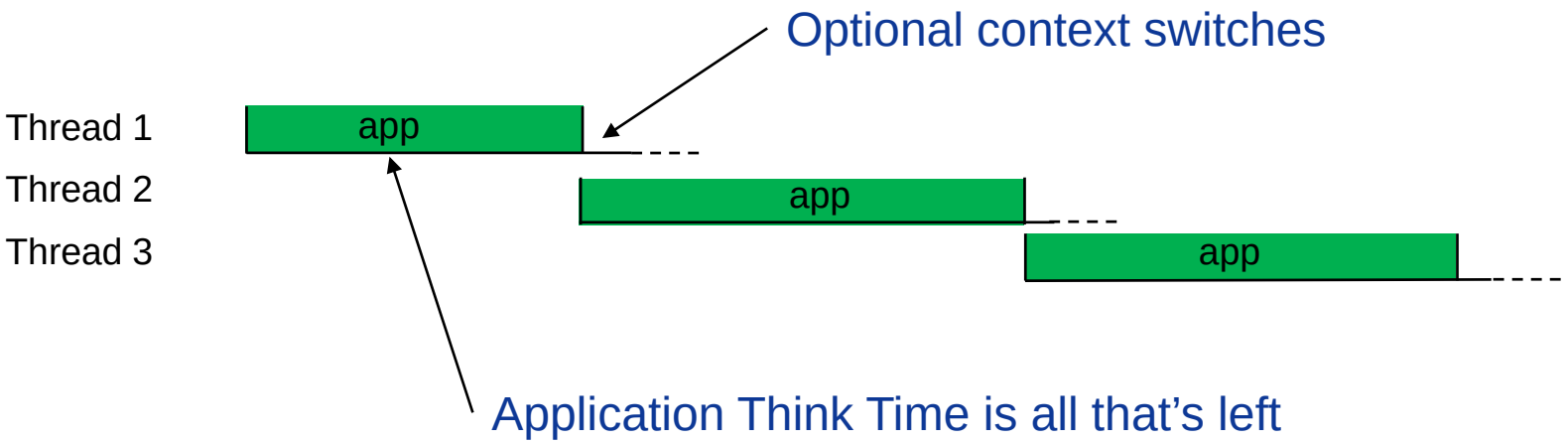
IO Bound Use Case: Closed Loop Threads

CPU time = app + dvr, limited threads



Persistent Memory Use Case: Always CPU/Memory Bound

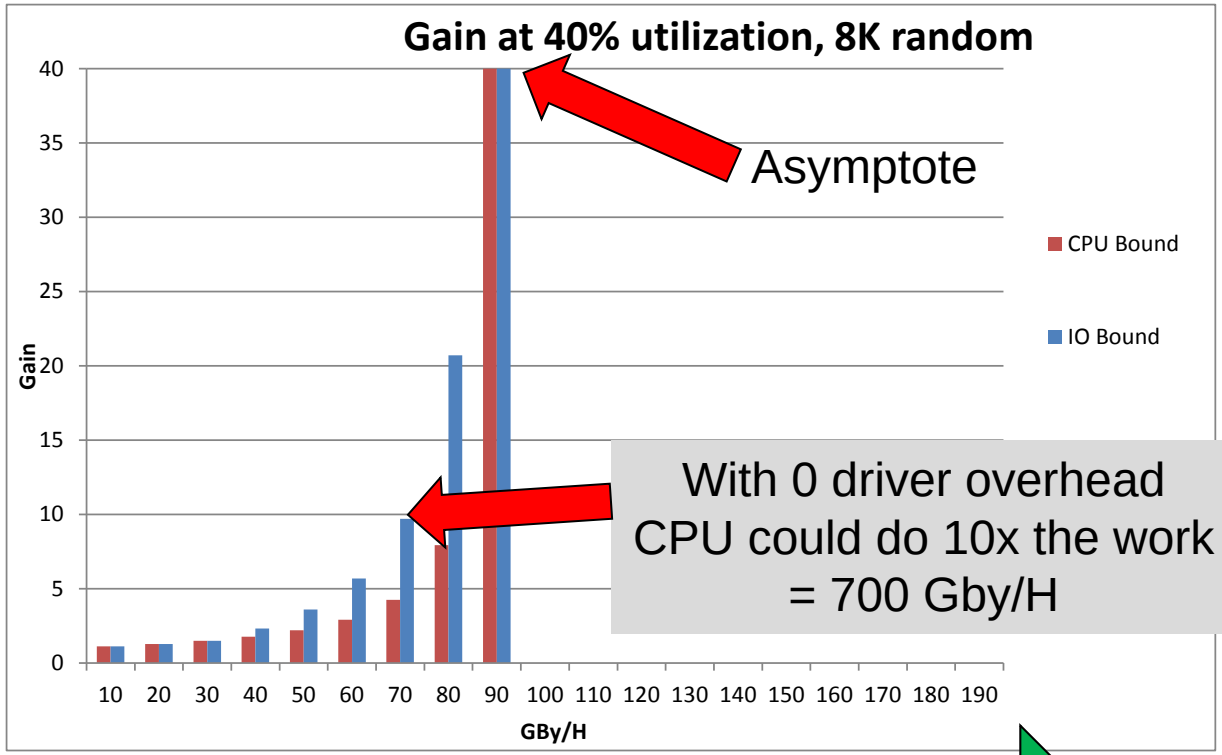
CPU time = app only, CPU bound



Application Think Time Paradox

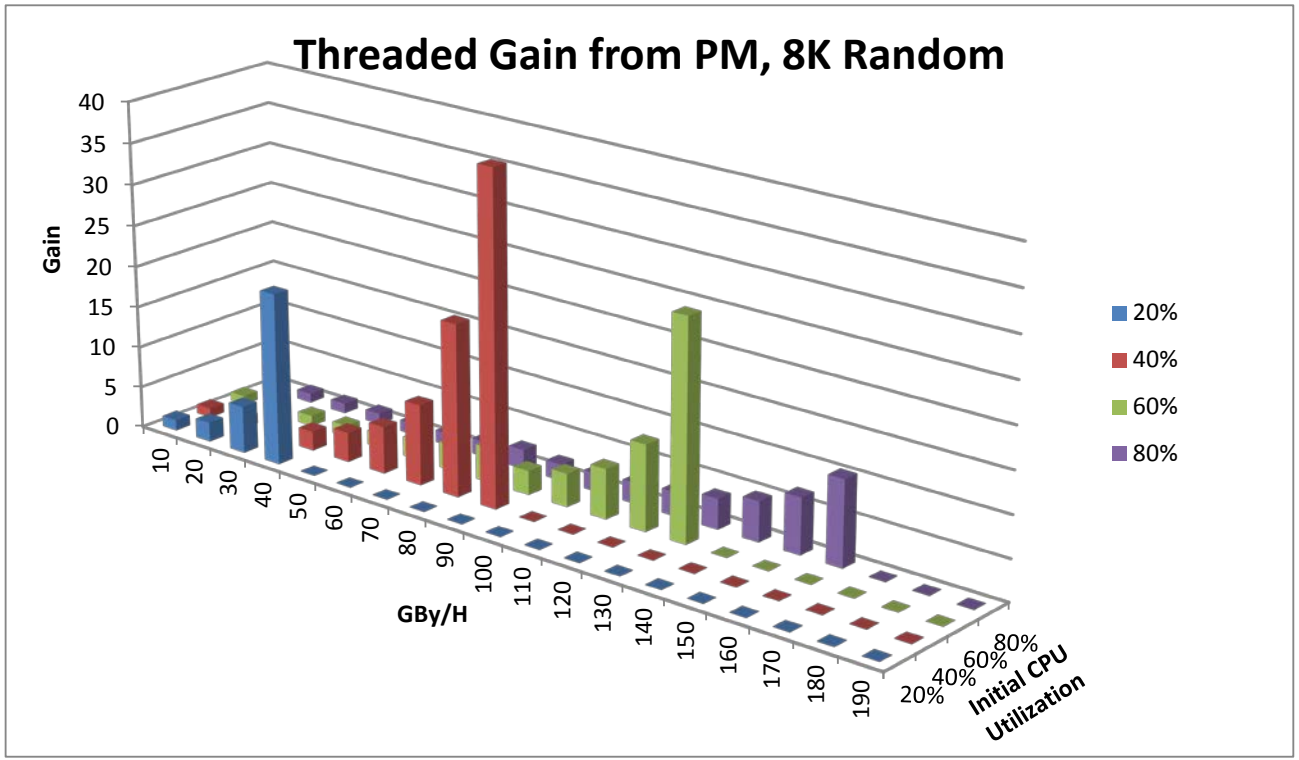
- Actual application think times are processor and application specific
- Use CPU utilization assumption instead – Creates a reading problem…
  - Suppose a CPU core is doing X operations per second
  - and each operation wastes Y microseconds (driver time).
  - If the current CPU utilization is Z%,
  - how much more work could be accomplished with Z% CPU utilization if Y were 0 (no driver overhead)
- Use real storage product latency vs. throughput measurements for comparison
  - As IOPs increase the CPU spends more and more total time in the driver
  - Amount of additional work (gain) has asymptote as driver overhead consumes entire core

Example gain curve



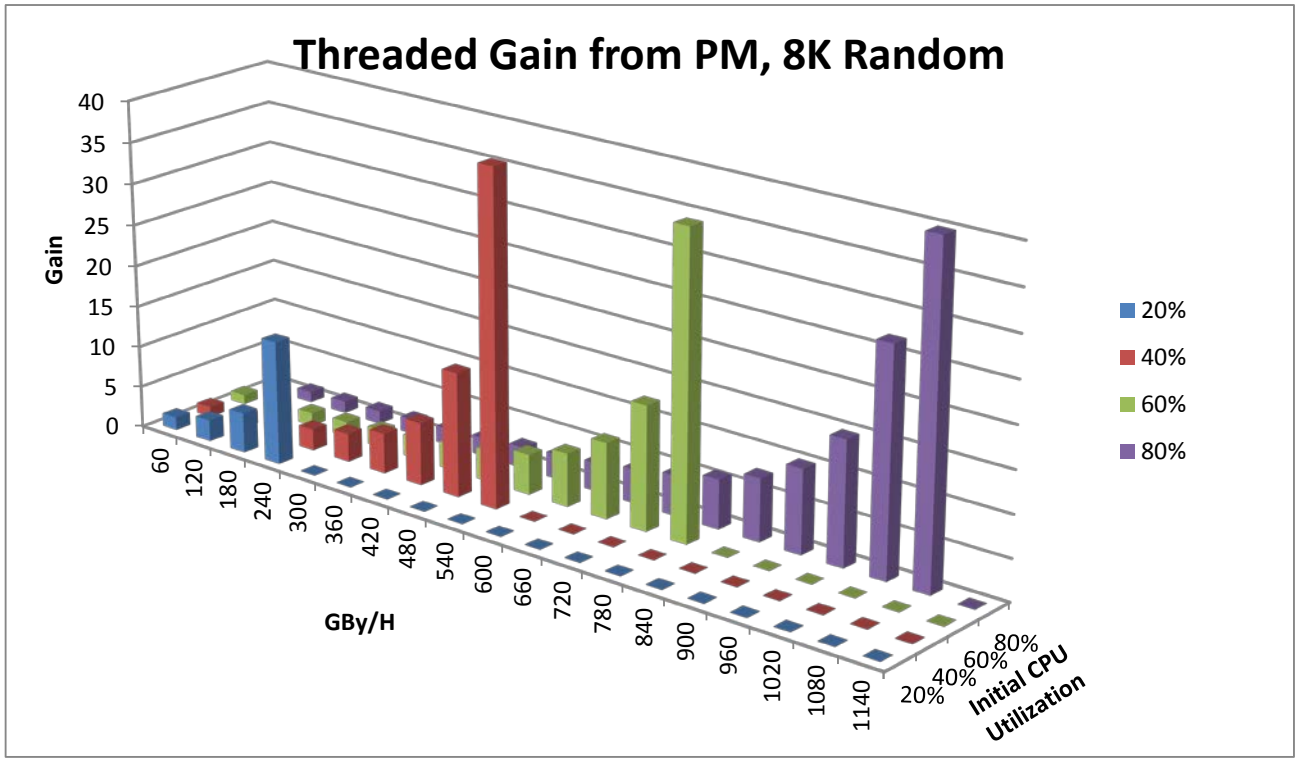
Primary Result for IO Bound Disk Array Workload, 120 uS Driver

Expect 5-10x gain when operating near, but not too near, saturation.



Primary Result for IO Bound PCIe Flash Workload, 20 uS Driver

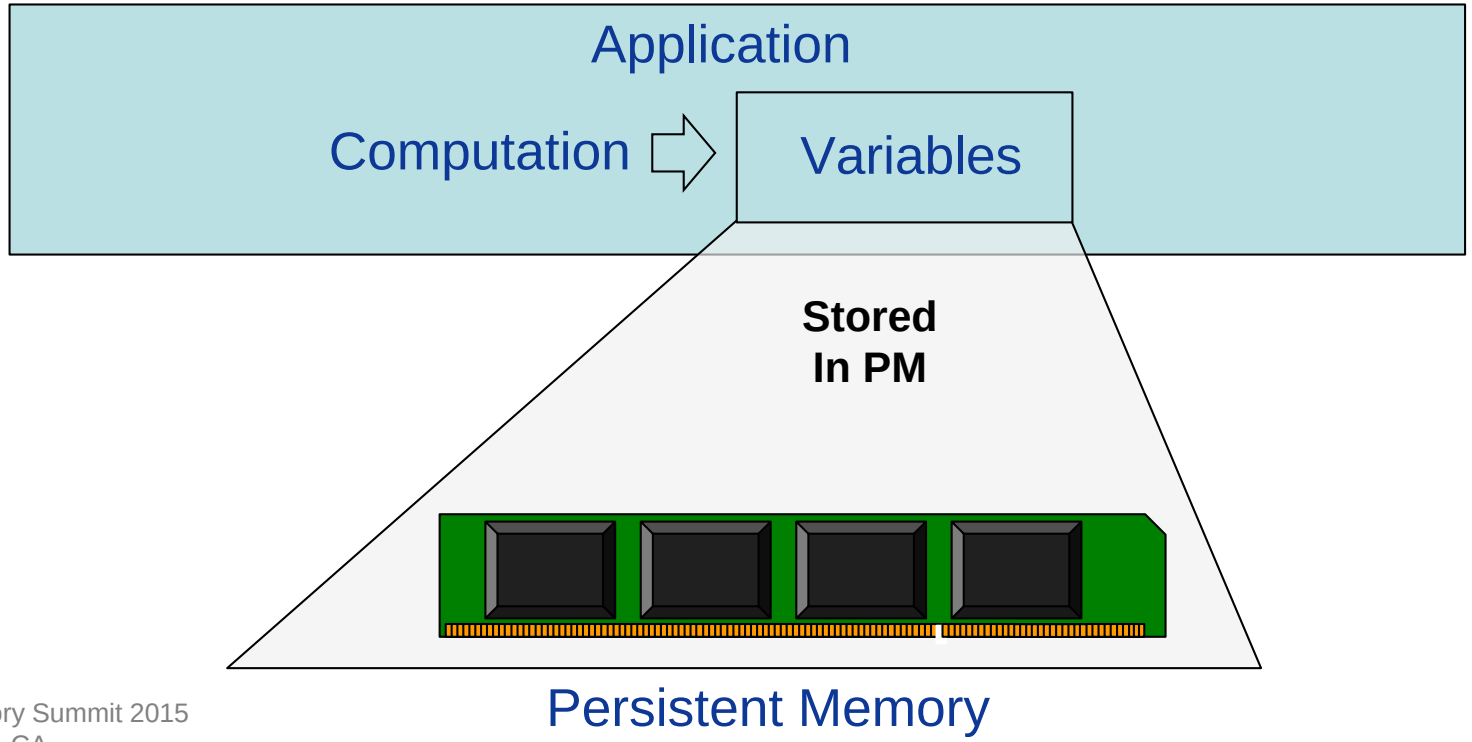
Very similar shape but on a different scale with elongated slope.



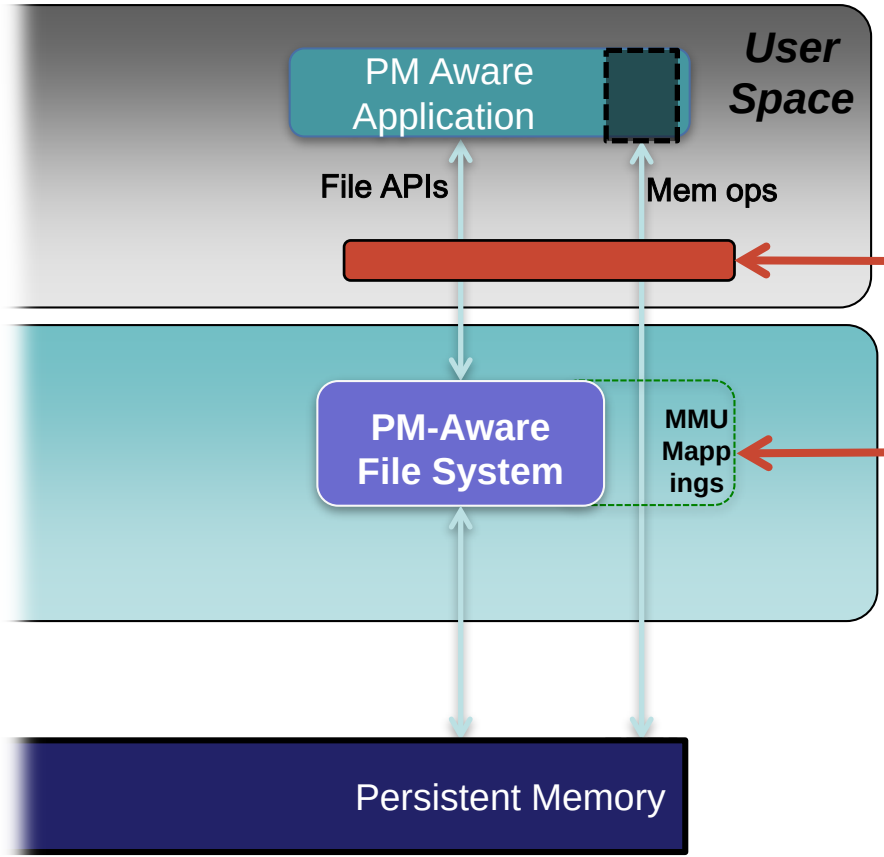
Conclusion for PM RAM Disk Which Applications Benefit:

- IO Bound
  - CPU bound workloads benefit less
  - PM turns previously IO bound workloads into CPU bound workloads
- Small Random Write
  - Sequential has lower overhead per Gby/H
  - Sequential lends itself to more efficient caching
  - Read intensive workloads are easily managed with caching
- Low Application Think Time
  - High application think times become CPU bound more quickly

Persistent memory as Ld/St



NVM Programming Model Example



PM data structure libraries

Middleware features e.g. RAID

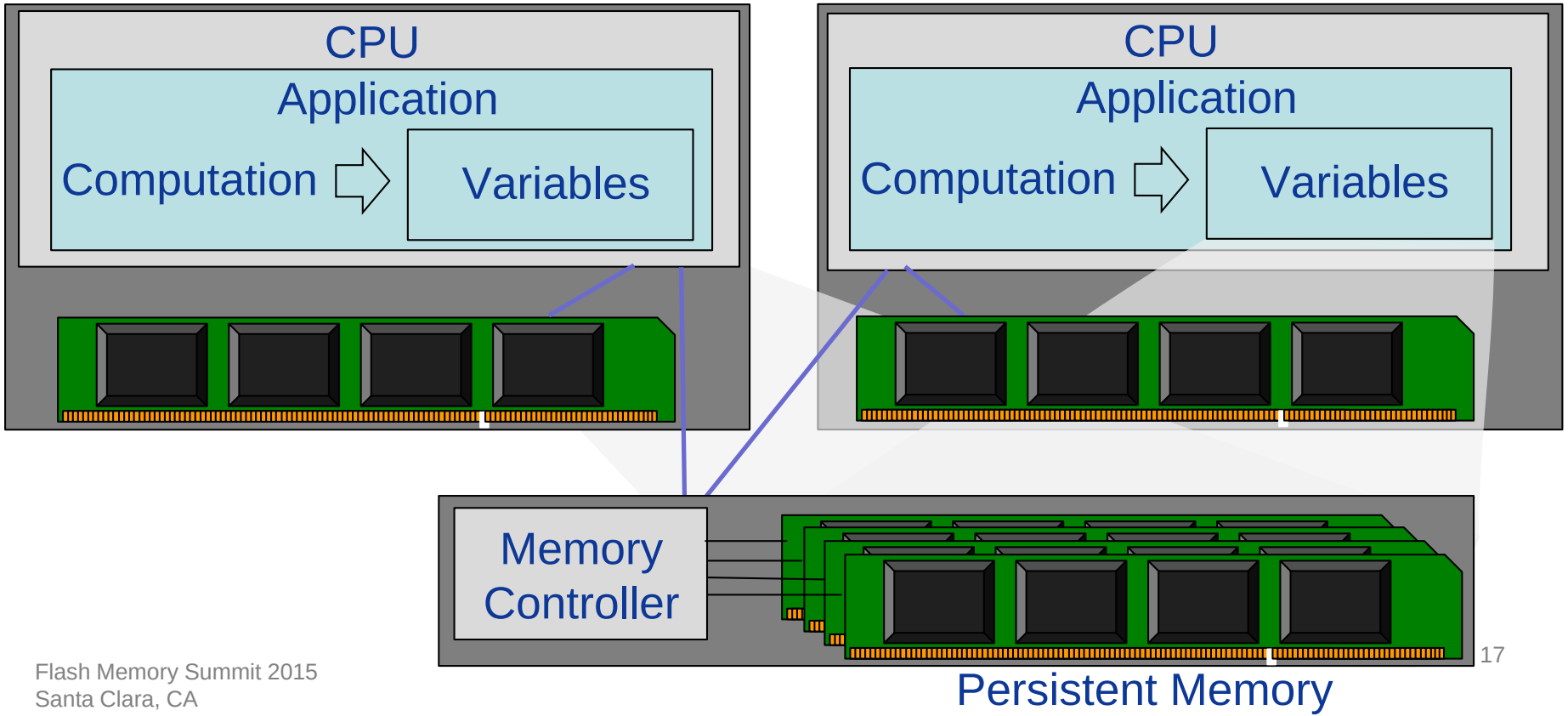
NVMP stack with libraries and HA

- PM data structure libraries make it easy to implement native PM data structures
  - Memory Mapped - allocate data structures from PM space within files
  - PM data structures can be manipulated by libraries using standard language features
  - Can be transactional – well defined commit points
- PM libraries and middleware hide complexity from applications such as
  - Sync – Flush CPU write pipeline to persistence domain
  - RAID (or erasure coding) – File system creates redundancy during sync
- However, Sync tends to interfere with optimal CPU performance
  - Invalidates cache lines, Causes remote access (for HA)
- How can sync disruption be mitigated?
  - Use Optimized Flush to delay and batch syncs into consistency points
  - Orchestrate recovery after failure to most recent consistency point
  - Similar to “Recover Point Objective” common in disk based disaster recovery systems

Conclusion for Ld/St
Which applications benefit:

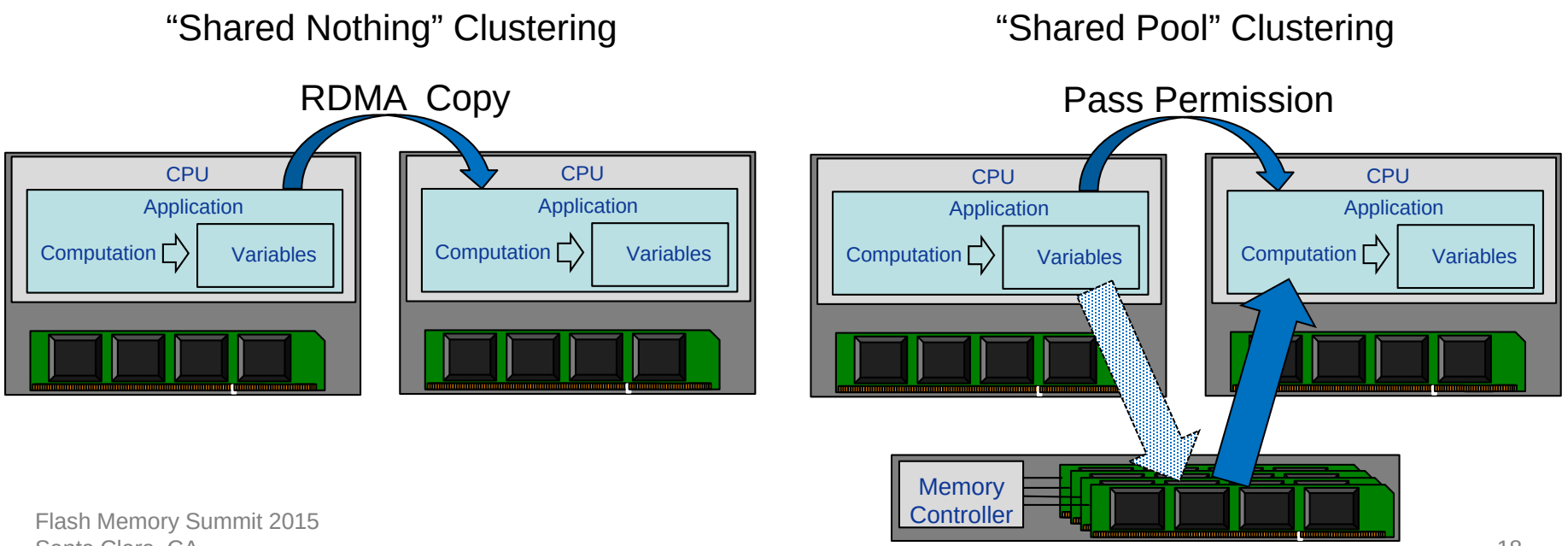
- Applications that use persistent memory data structures
- Applications that can group data structure manipulations into consistency points
  - Similar to transaction commits in classical databases
- Applications that can backtrack to the most recent commit point after failure
  - Exception handling replaces IO status
  - Avoid system restart on memory error
- Applications that use proven PM libraries to encapsulate complexity

Disaggregated Persistent Memory



Non-Cache-Coherent memory sharing

- Ld/St access per processor complex with cluster style logic across nodes
- Replace message passing with memory mapping of shared pool



Overall Conclusion
Which applications benefit:

- NVRAM Disk
  - IO Bound
  - Small Random Write
  - Low Application Think Time
- Persistent memory data structures
  - Use them
  - Group data structure manipulations into consistency points
  - Backtrack to the most recent commit point after failure
- Share cluster pooled memory instead of copying

Thank You