Architectural Principles for Networked Solid State Storage Access

SNIA Legal Notice

- The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

  NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

Who We Are

Doug Voigt
Chair, NVM Programming Model,
SNIA Technical Council
Distinguished Technologist, HPE

J Metz
SNIA Board of Directors
R&D Engineer
Cisco

Focus on Principles

- Technical and market dynamics are creating complexity
  - Permutations of technologies and positioning
  - Intricacies of new technology integration
- Foundational principles have not changed
  - Application views of memory and storage
  - The role of data access time (latency) in system architecture
- Principles transcend details
  - This is not a best practices presentation
  - This is presentation does not report benchmark results

Times are changing

- Storage access times are shrinking
  - Emerging persistent memory (PM) technologies
  - Faster than flash
- Interconnects are getting faster
  - Bandwidth
  - Latency
- Creating challenges for software
  - Software stacks are starting to dominate latency
  - Trigger for a fundamental architecture shift

Principles we will cover today

- Application view – I/O vs. Load/Store (Ld/St)
- Disruption caused by latency
- Persistence domains
- Latency impact of remote persistence

Application View

Input/Output (IO)

- Data is read or written using RAM buffers
- Software has control over how to wait
  - Poll
  - Context switch
- Status is explicitly checked by software

What is IO?

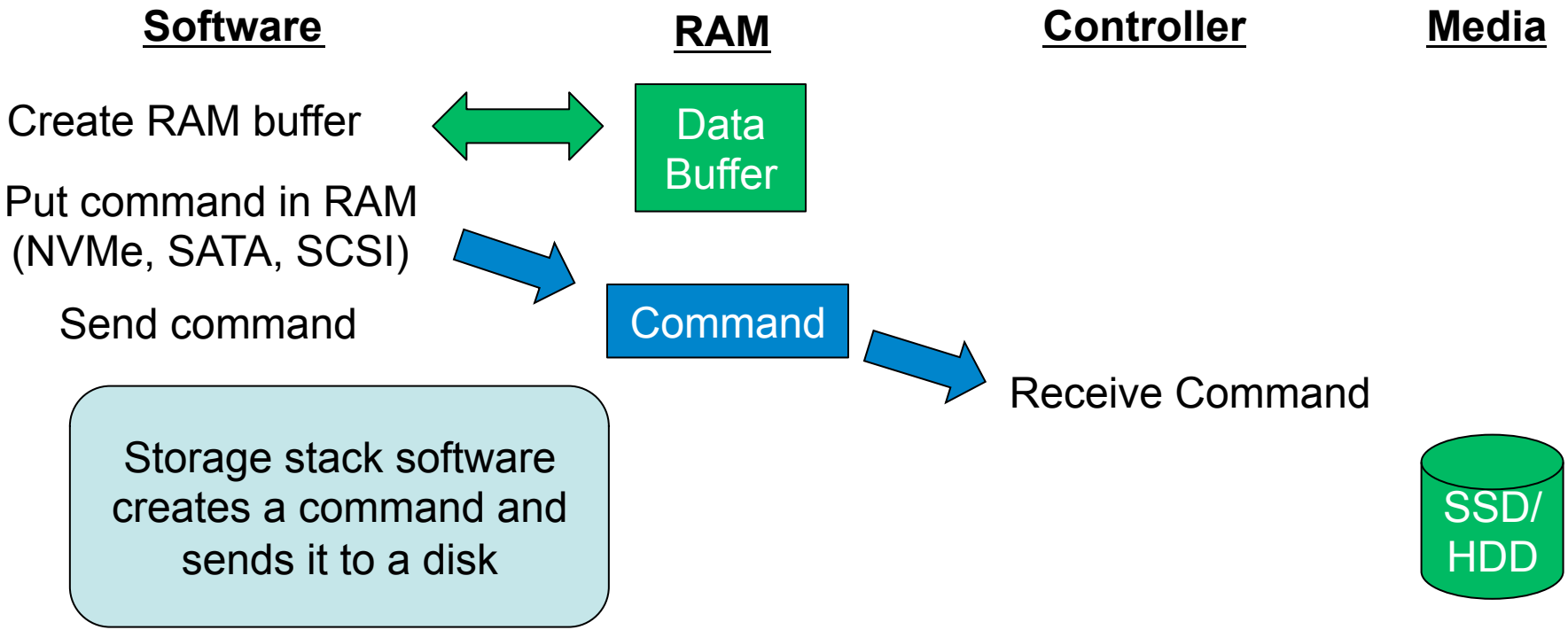
Software **RAM** **Controller** **Media**

Create RAM buffer

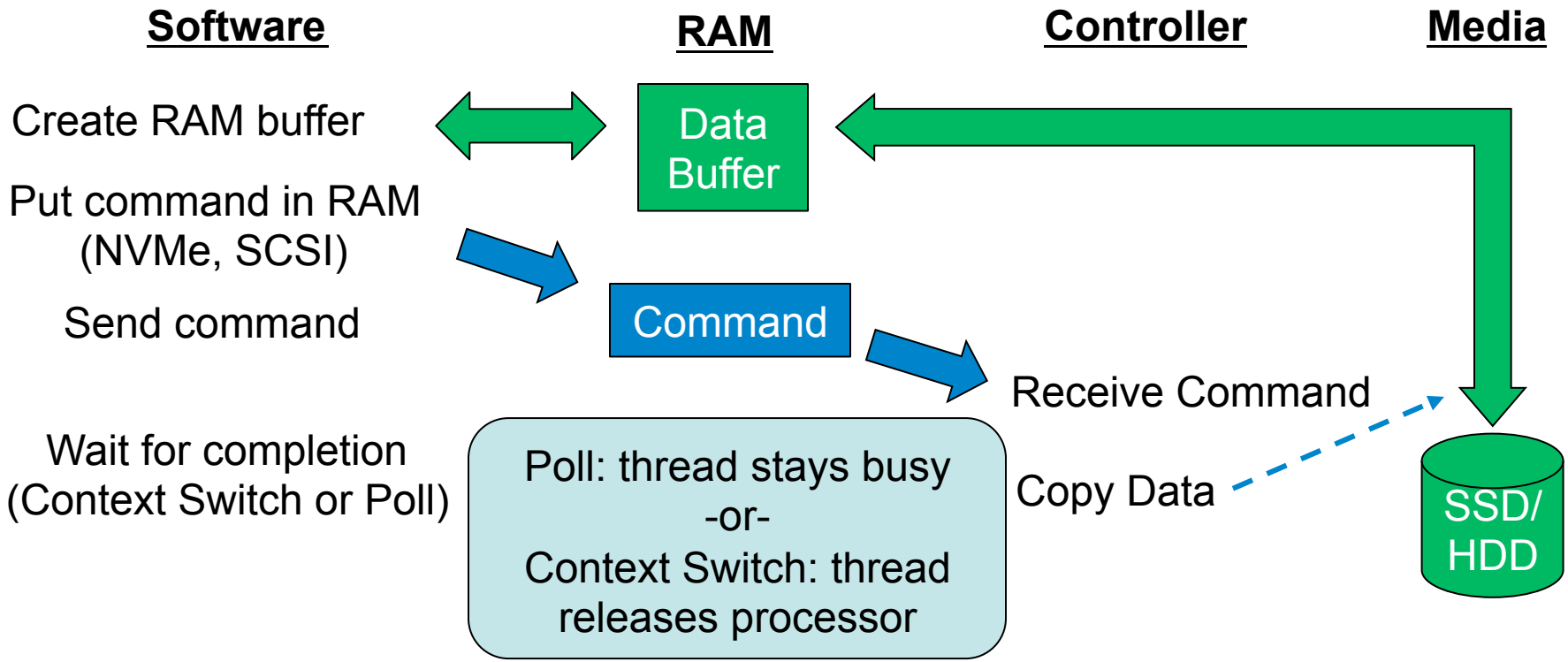
Data Buffer

IO copies data between RAM and Disk
(HDD: Hard Disk Drive, SSD: Solid State Disk)

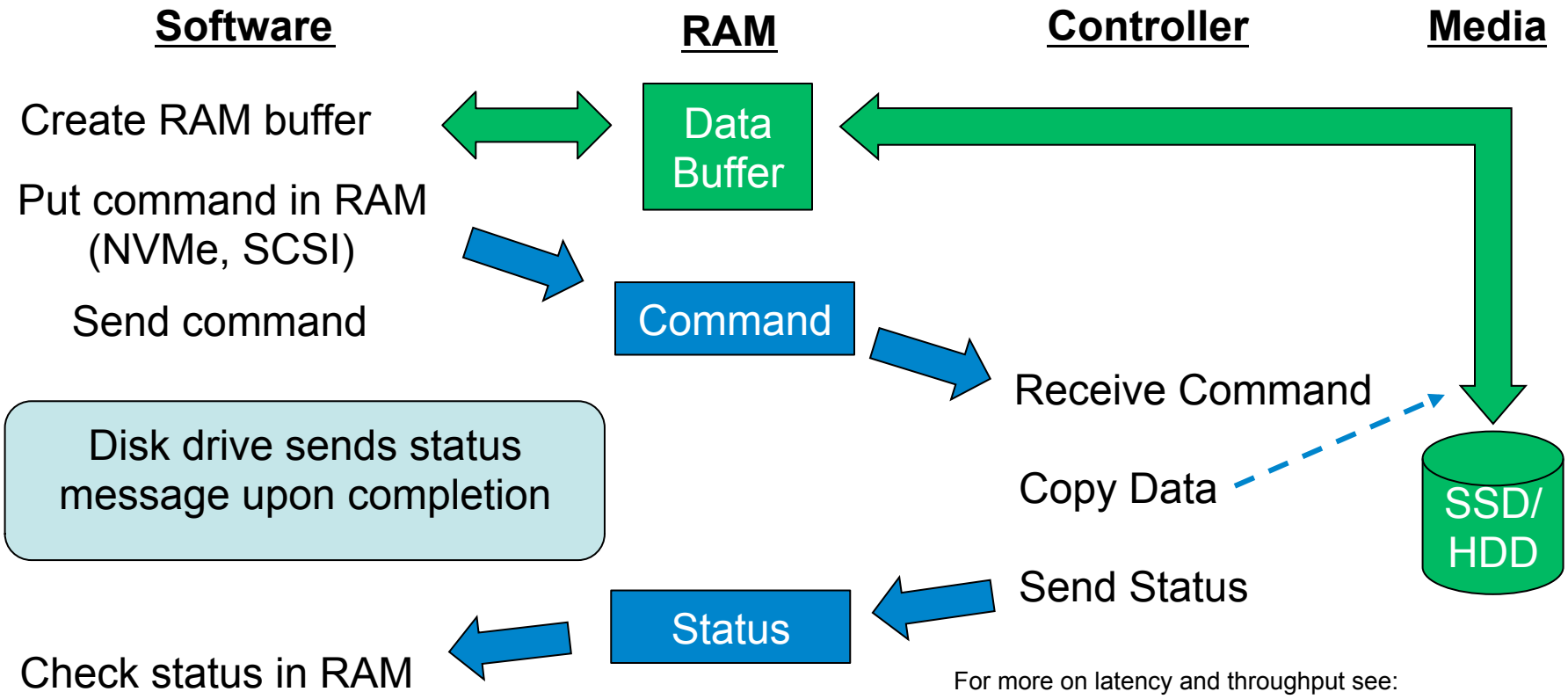
What is IO?



What is IO?

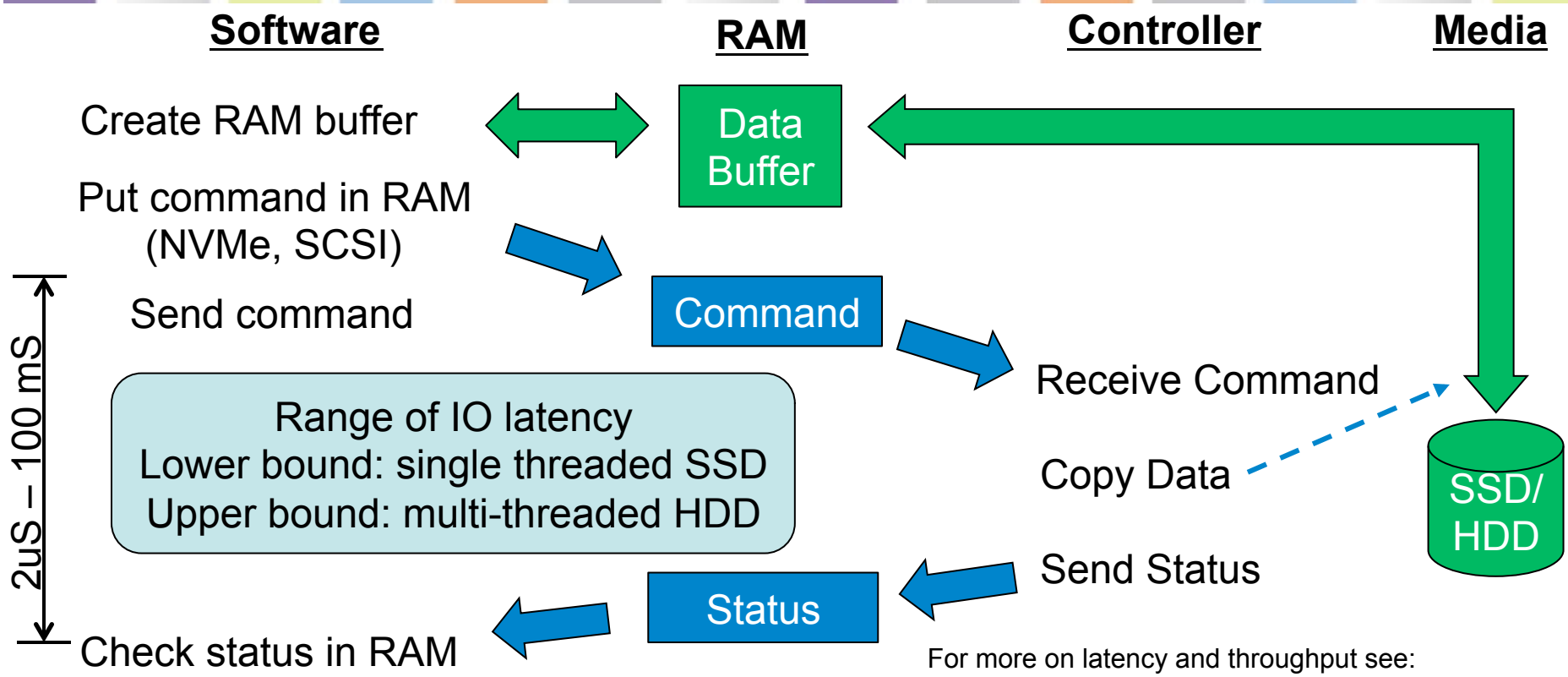


What is IO?



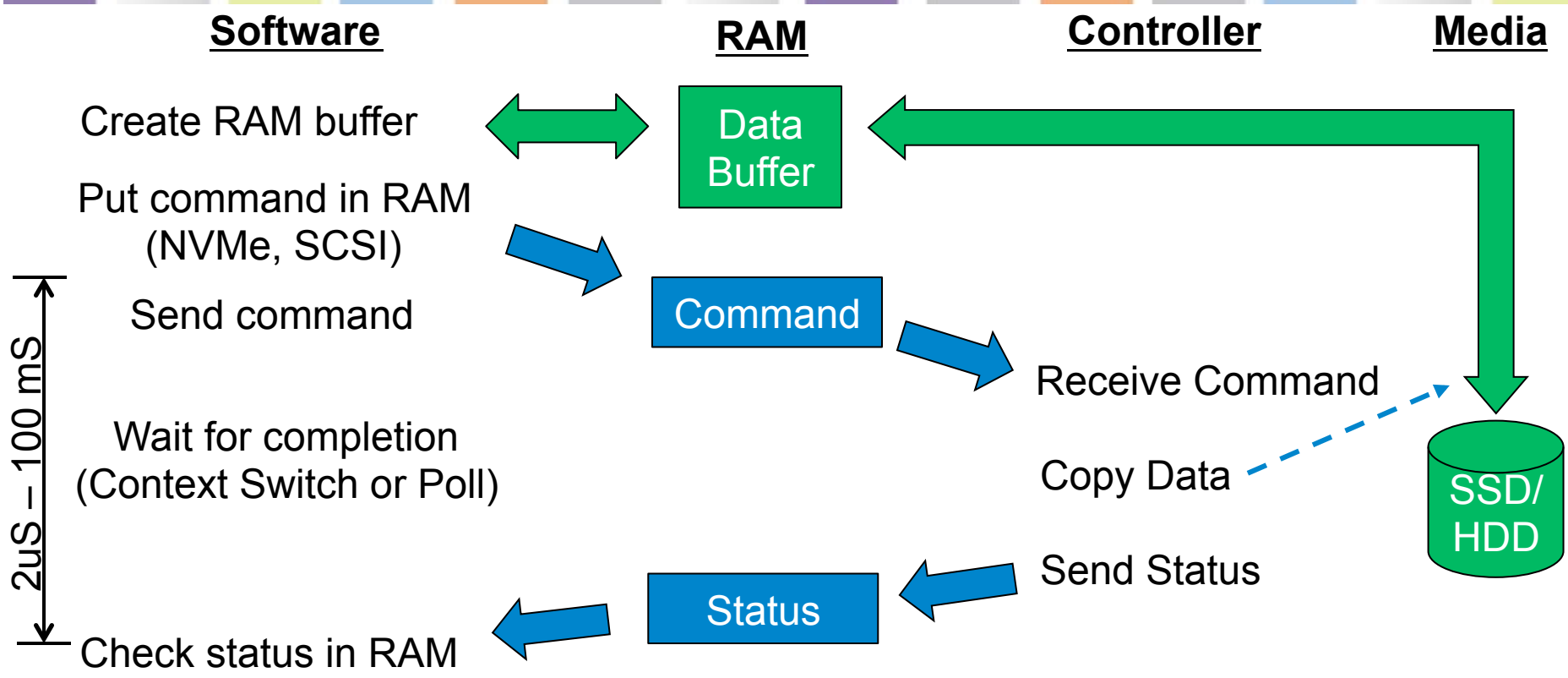
For more on latency and throughput see:
https://www.brighttalk.com/webcast/663/164323

What is IO?



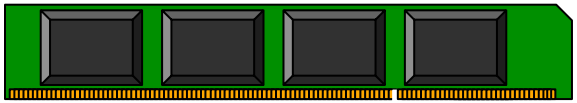
For more on latency and throughput see:
https://www.brighttalk.com/webcast/663/164323

What is IO?



Enter Persistent Memory

- Much faster than flash alone
- Available today in the form of an NVDIMM
- Can be used to store application data structures
  - Software uses Pmalloc (Persistent Memory Allocation) to allocate a region of persistent memory
  - Accessed by processor instructions like any other memory



Load/Store (Ld/St)

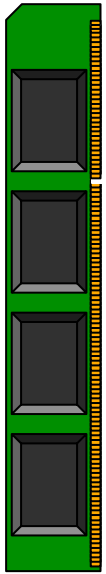
- Load and Store are processor instructions
  - Generated by compliers to access memory
  - Proxy for any processor instruction that accesses memory
- Data is loaded into or stored from processor registers
- Software may be forced wait for data during an instruction
  - The thread is busy
  - The processor won't let the thread do anything else
- No status checking – errors generate exceptions

What is Ld/St?

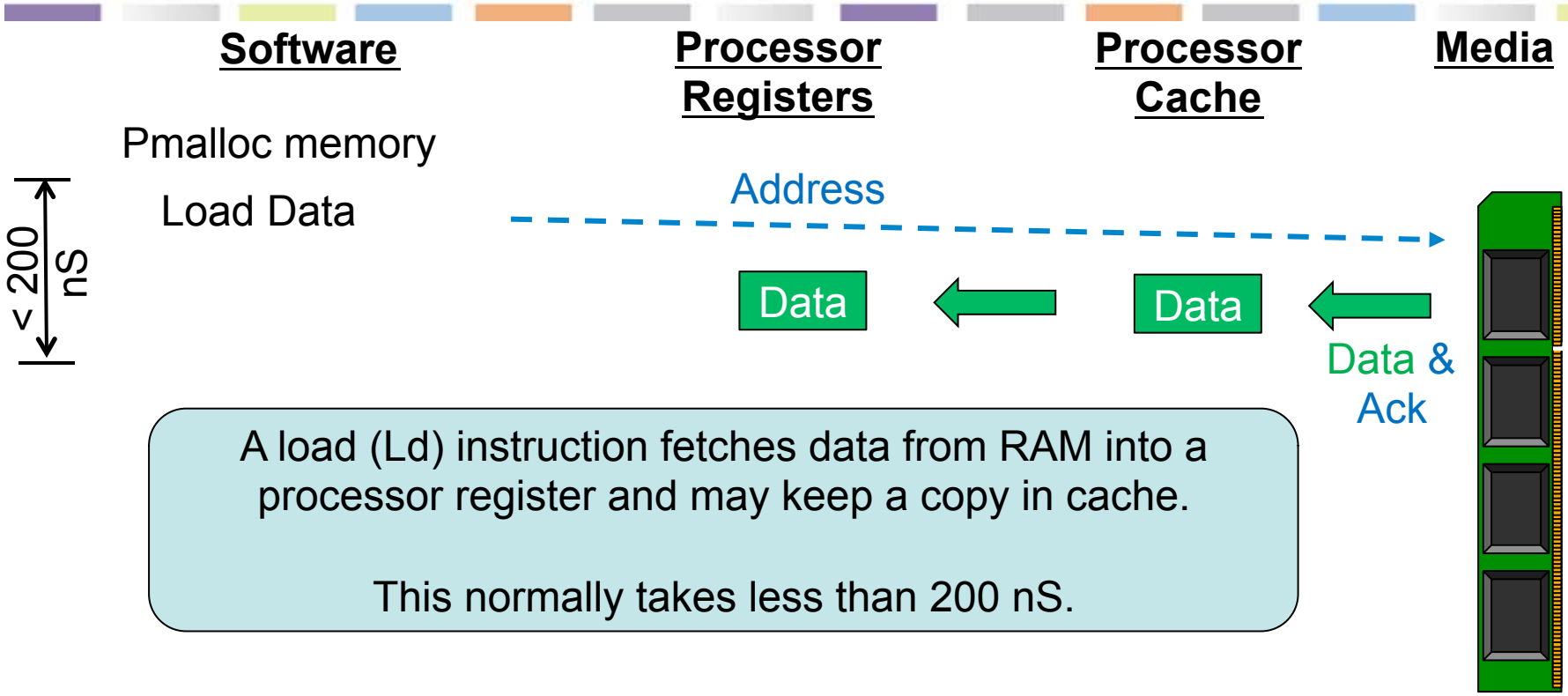
Software | **Processor Registers** | **Processor Cache** | **Media**

Pmalloc memory

Pmalloc returns persistent memory (PM) where software data structures can be permanently stored.



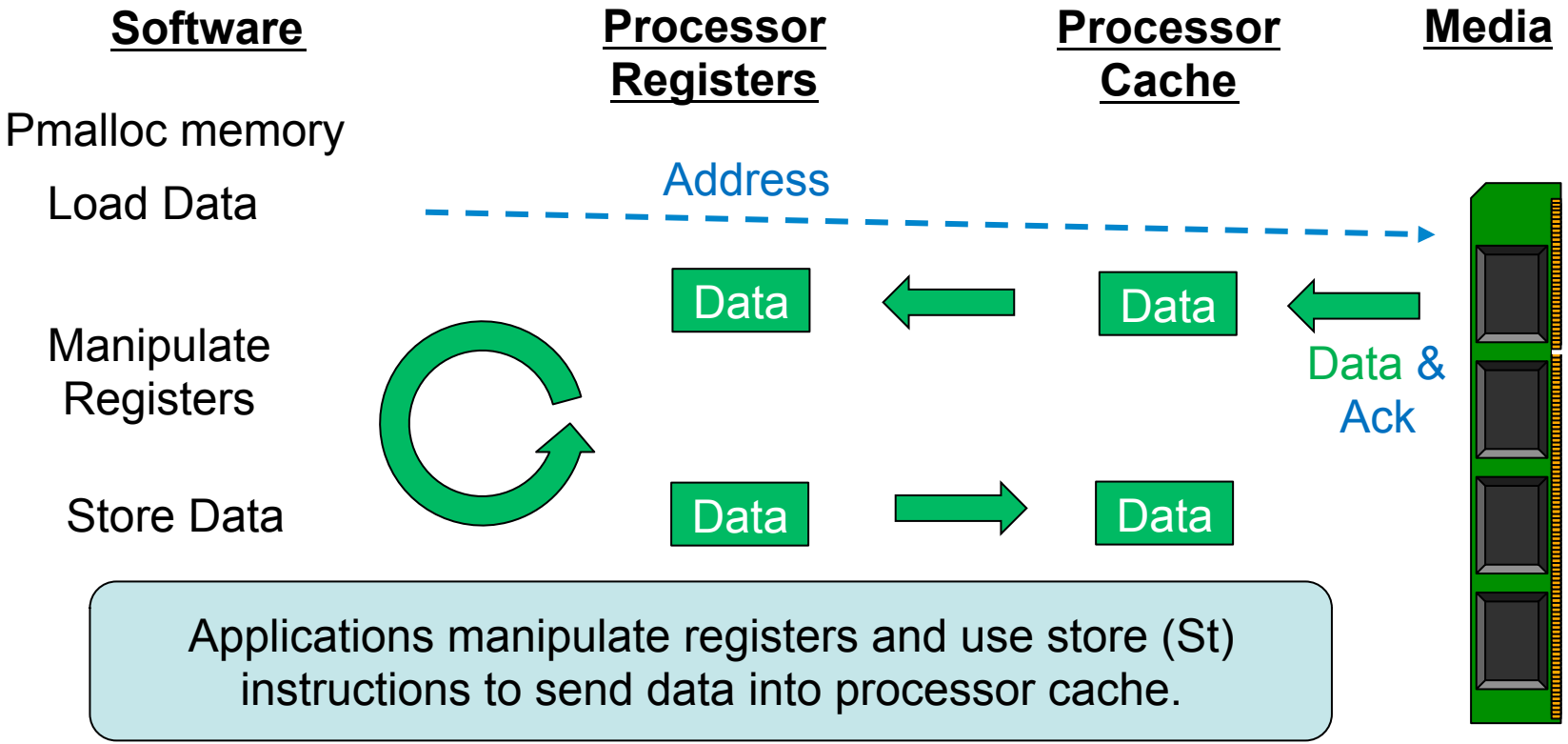
What is Ld/St?



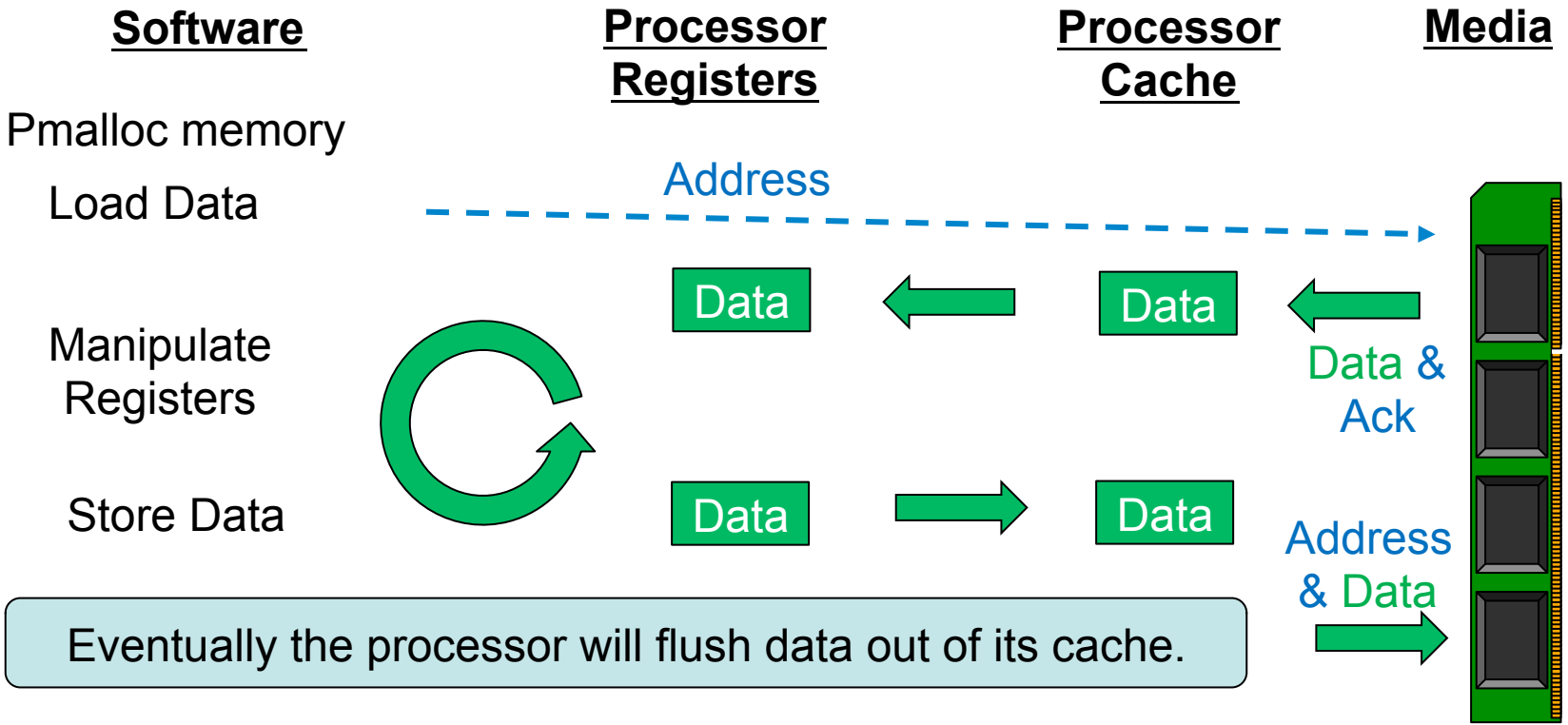
A load (Ld) instruction fetches data from RAM into a processor register and may keep a copy in cache.

This normally takes less than 200 nS.

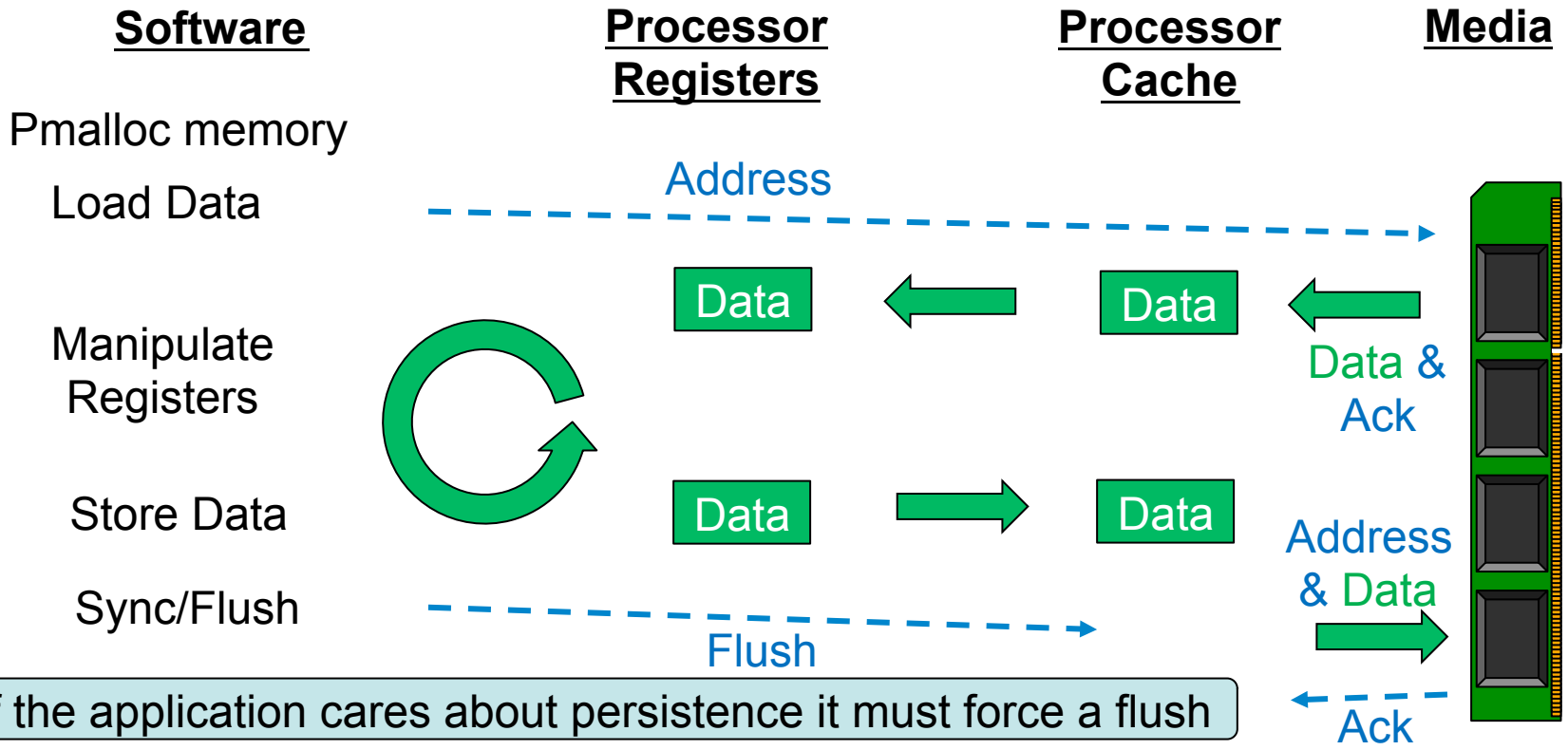
What is Ld/St?



What is Ld/St?

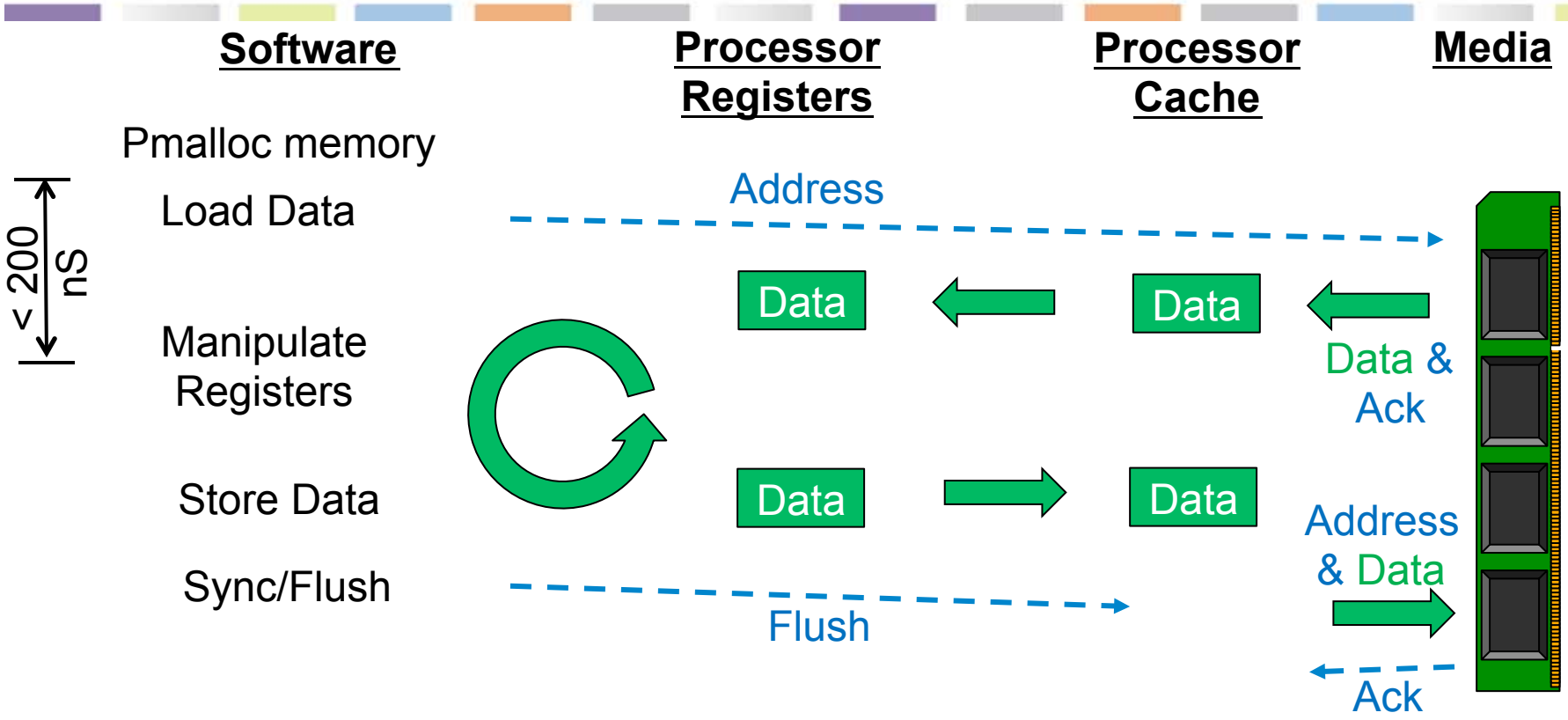


What is Ld/St?



If the application cares about persistence it must force a flush

What is Ld/St?



Compare and Contrast – Application View

- IO
  - Data is read or written using RAM buffers
  - Software has control over how to wait (context switch or poll)
  - Status is explicitly checked by software
- Ld/St
  - Data is loaded into or stored from processor registers
  - Software is forced by processor to wait for data during instruction
  - No status checking – errors generate exceptions

Ld/St and memory mapping

- "Memory mapped IO"
  - PCIe card registers are given memory addresses
  - Ld/St is used to access those registers
  - Designed for use to control IO (or other) card functions
- "Memory mapped files"
  - Data stored in files (in PM) is given memory addresses
  - Ld/St is used to manipulate user data
  - Must flush after St to assure persistence

Latency Is The Disrupter

Compare and Contrast – Technology View

Key Question: Is it OK to force the CPU memory access pipeline to wait for a given storage or memory access to complete before proceeding?

- May pause a thread in the middle of executing an instruction
- May block reads and writes to all cores

When does latency fit into a Ld/St Budget?

Key Question: Is it OK to force the CPU memory access pipeline to wait for a given storage or memory access to complete before proceeding?

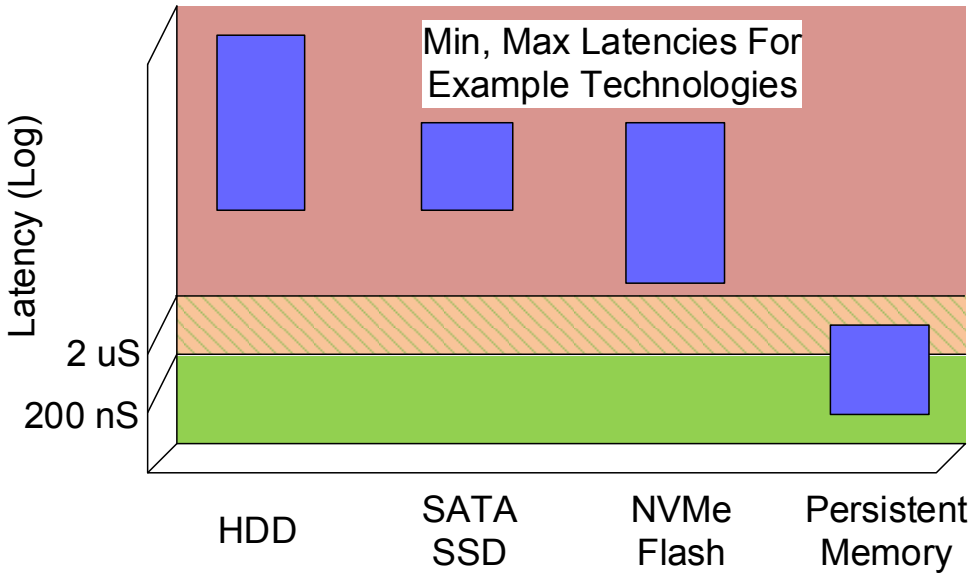
- May pause a thread in the middle of executing an instruction
- May block reads and writes to all cores

Technology	DDR Connect	Mem Mapped PCIe	PCIe I/O	Network (IB/Enet/FC)
RAM	Yes	Maybe	No	No
Battery+RAM	Yes	Maybe	No	No
PM	Yes	Maybe	No	No
Flash	Sometimes	Sometimes	No	No
Rotating	NA	NA	No	No

Sometimes??!!

- Flash write latency generally does not fit in Ld/St Budget
  - Write process takes much longer than a St
  - Long latency tail due to erase cycle, garbage collection
- Flash read latency interference from controller
  - Erase cycle/wear leveling forces block granularity indexing
  - Index traversal at SSD scale would stress Ld/St latency budget
- Flash with cache
  - Makes latency unpredictable
  - Is cache non-volatile? Hold that thought…

Latency Thresholds Cause Disruption



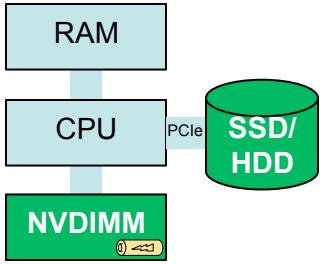
* Non Uniform Memory Access

Persistence Domains

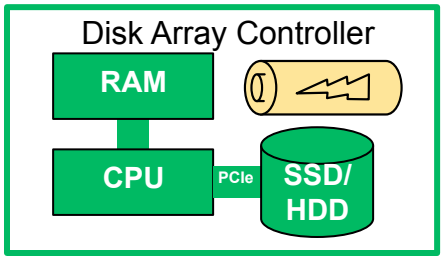
What is a persistence domain?

- A place that retains data during power loss
  - Because it is inherently non-volatile
  - Because it is flushed during or after power loss
- Examples:
  - A group of flash or PM devices
  - An NVDIMM
  - A group of RAM devices that are powered by a battery
  - (part of) A controller that flushes caches and RAM to an SSD
    - With battery power
    - With capacitance

Examples of Persistence Domains



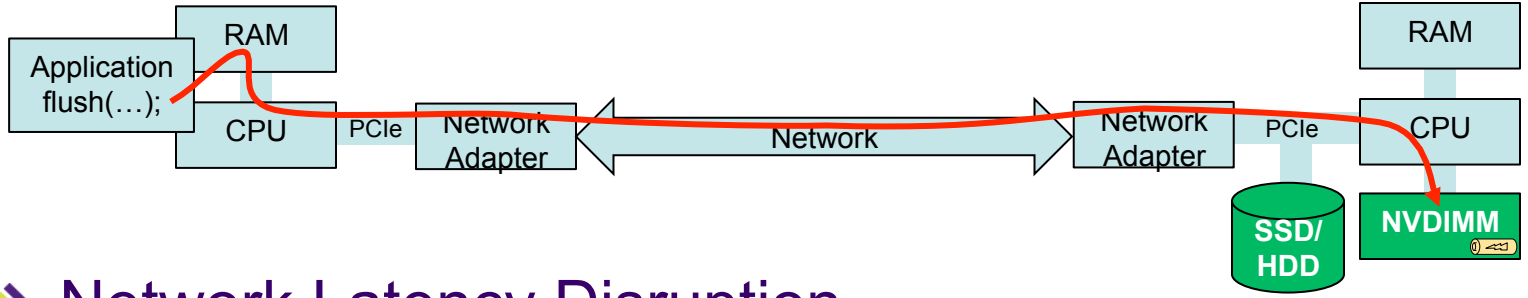
- Requires Flush-On-Demand
  - Explicit flush by application or library
  - Disrupts CPU Caches



- Enables Flush-On-Fail
  - "Everything" is saved after power loss
  - We're gonna need a bigger battery

Interconnect Latency

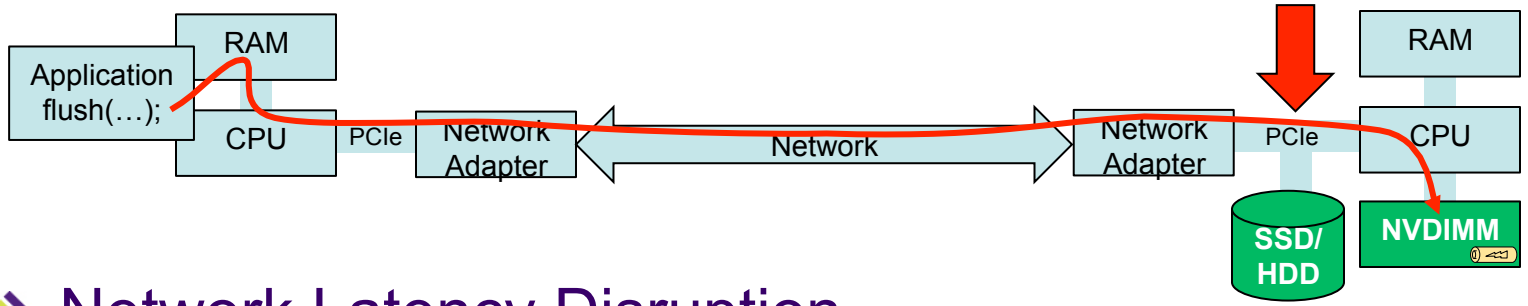
What about remote persistence domains?



- Network Latency Disruption
  - Most of today's networks do not approach Ld/St timescales.
  - The fastest port to port times for IB and projected Omnipath technologies might fit within a Ld/St budget
  - RDMA helps by eliminating round trips and software overhead
  - BUT…

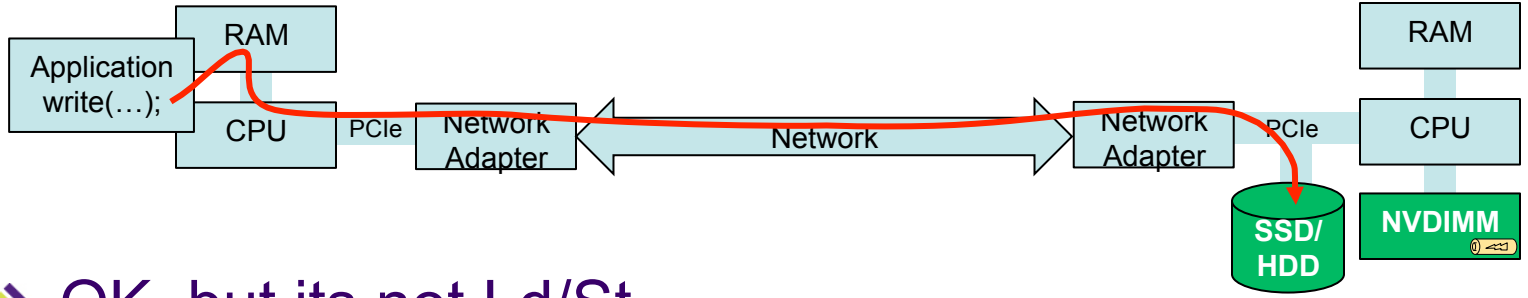
For more on RDMA see:
https://www.brighttalk.com/webcast/663/185909

What about remote persistence domains?



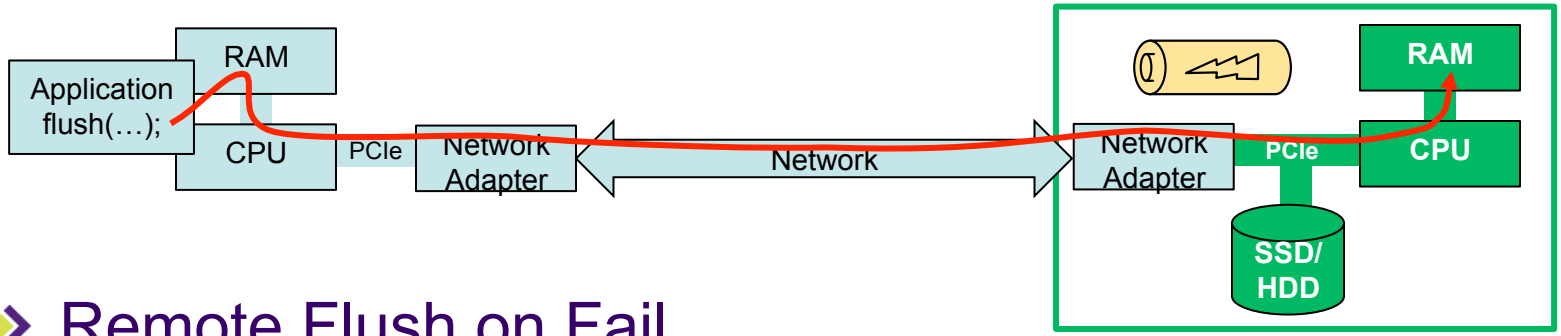
- Network Latency Disruption
  - There is no hardware flush action over PCI
  - AND there is no RDMA completion that indicates persistence
  - SO software is involved at the remote node

So How About This?



- OK, but its not Ld/St
  - Avoids remote processor
  - Minimal remote latency for IO

Or This?



- Remote Flush on Fail
  - This is how inter-controller links on disk arrays work.
  - But it is still very difficult to get sync anywhere near Ld/St latency
    - especially across blades or chassis
    - or with routing
  - AND remember, the remote write is on flush, not St.

Current Industry Challenge

Bring remote flush latency much closer to Ld/St time

- Effects all of the elements in the system
  - Processor
  - NIC
  - Persistence Domain
- Lots of activity is happening so stay tuned

NVM Programming Model TWG work

One important use case is described by SNIA's new "NVM PM Remote Access for High Availability" paper

- Taxonomy of local and remote use cases
- Application recoverability requirements
- Model and requirements for remote implementations of optimized flush as described in the "SNIA NVM Programming Model"

After This Webcast

SNIA Ethernet Storage Forum

- Please rate this Webcast and provide us with feedback
- This Webcast and a PDF of the slides will be posted to the SNIA Ethernet Storage Forum (ESF) website and available on-demand
- http://www.snia.org/forums/esf/knowledge/webcasts
- A full Q&A from this webcast, including answers to questions we couldn't get to today, will be posted to the SNIA-ESF blog
- http://sniaesfblog.org/
- Follow us on Twitter @ SNIAESF

Architectural Principles for Networked Solid State Storage Access