CS250B: Modern Computer Systems

Organizing Storage Devices

Sang-Woo Jun

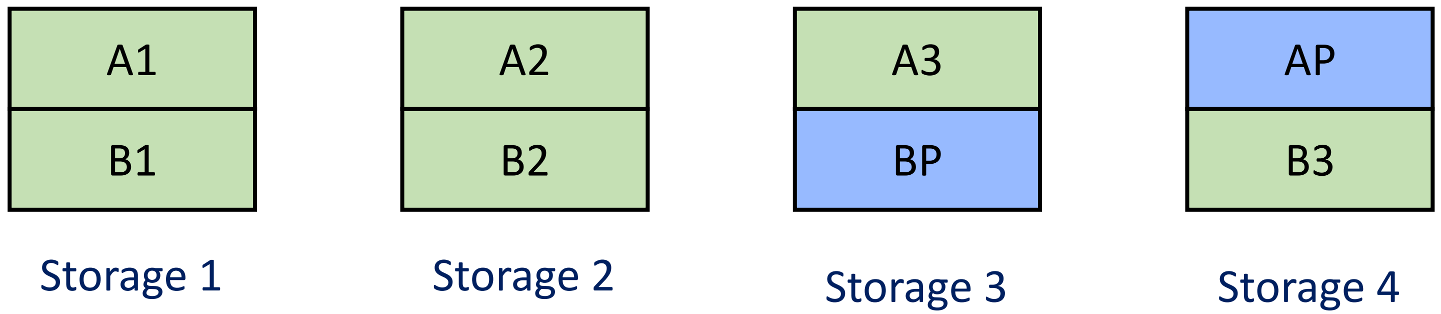
UCI

Redundant Array of Independent Disks (RAID)

- Technology of managing multiple storage devices
  - Typically in a single machine/array, due to limitations of fault-tolerance
- Multiple levels, depending on how to manage fault-tolerance
  - RAID 0 and RAID 5 most popular right now
- RAID 0: No fault tolerance, blocks striped across however many drives
  - Fastest performance
  - Drive failure results in data loss
  - Block size configurable
  - Similar in use cases to the Linux Logical Volume manager (LVM)

Fault-Tolerance in RAID 5

- RAID 5 stripes blocks across available storage, but also stores a parity block
  - Parity block calculated using xor (A1^A2^A3=AP)
  - One disk failure can be recovered by re-calculating parity
    - A1 = AP^A2^A3, etc
  - Two disk failure cannot be recovered
  - Slower writes, decreased effective capacity



Degraded Mode in RAID 5

- In case of a disk failure it enters the “degraded mode”
  - Accesses from failed disk is served by reading all others and xor’ing them (slower performance)
- The failed disk must be replaced, and then “rebuilt”
  - All other storages are read start-to-finish, and parity calculated to recover the original data
  - With many disks, it takes long to read everything – “Declustering” to create multiple parity domains
  - Sometimes a “hot spare” disk is added to be idle, and quickly replace a failed device

Storage in the Network

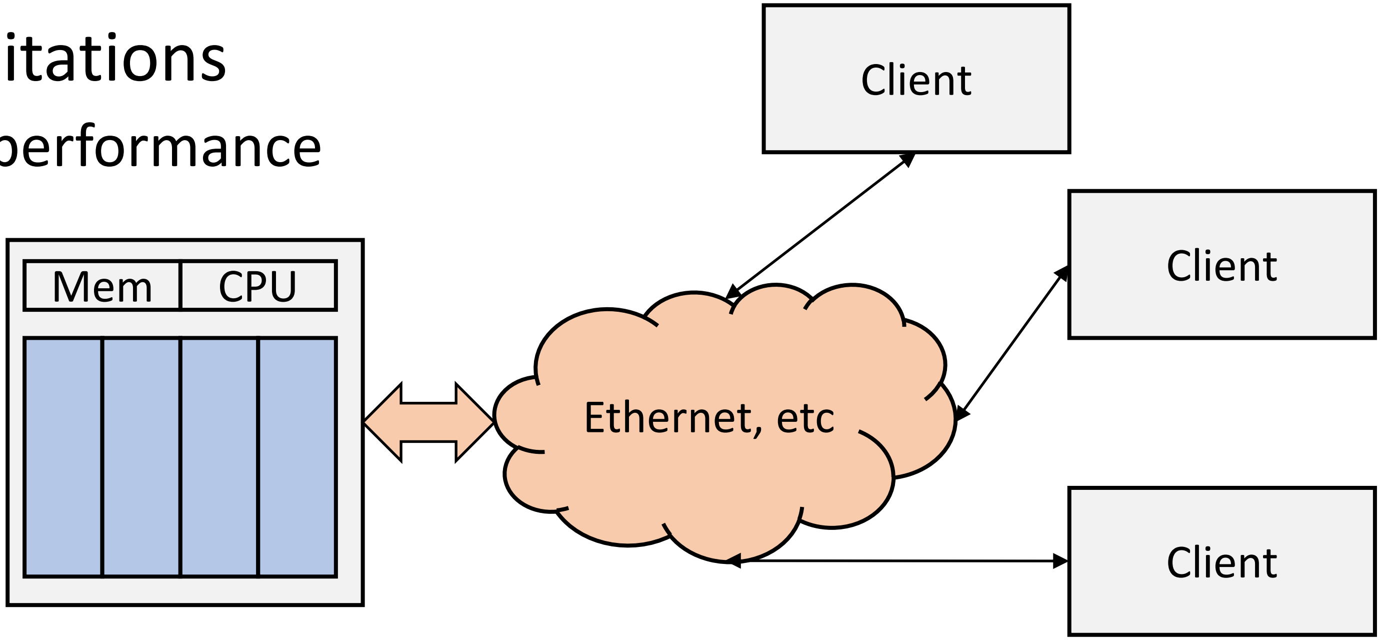
- Prepare for lightning rounds of very high-level concepts!

Network-Attached Storage (NAS)

- Intuition: Server dedicated to serving files “File Server”
  - File-level abstraction
  - NAS device own the local RAID, File system, etc
  - Accessed via file system/network protocol like NFS (Network File System), or FTP
- Fixed functionality, using embedded systems with acceleration
  - Hardware packet processing, etc
- Regular Linux servers also configured to act as NAS
- Each NAS node is a separate entity – Larger storage cluster needs additional management

Network-Attached Storage (NAS)

- Easy to scale and manage compared to direct-attached storage
  - Buy a NAS box, plug it into an Ethernet port
  - Need more storage? Plug in more drives into the box
- Difficult to scale out of the centralized single node limit
- Single node performance limitations
  - Server performance, network performance

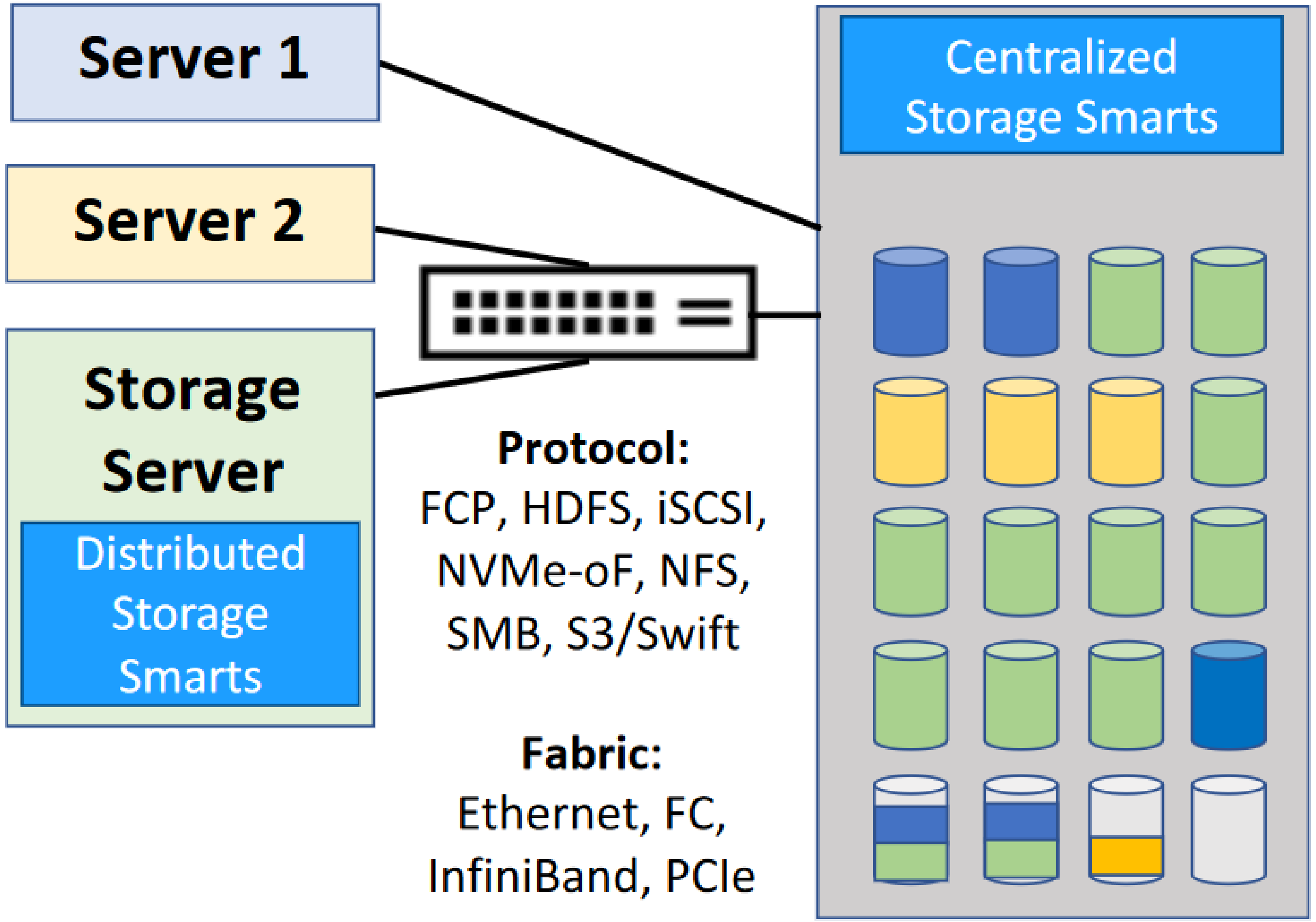


Storage-Area Networks (SAN)

- In the beginning: separate network just for storage traffic
  - Fibre Channel, etc, first created because Ethernet was too slow
  - Switch, hubs, and the usual infrastructure
- Easier to scale, manage by adding storage to the network
  - Performance distributed across many storage devices
- Block level access to individual storage nodes in the network
- Controversial opinion: Traditional separate SAN is dying out
  - Ethernet is unifying all networks in the datacenter
    - 10 GbE, 40 GbE slowly subsuming Fibre Channel, Infiniband, …

Disaggregated Storage

- ❑ Allows storage resources to scale independently



Converged Infrastructure

- Computation, Memory, Storage converged into a single unit, and replicated
- Became easier to manage compared to separate storage domains
  - Software became better (Distributed file systems, MapReduce, etc)
  - Decreased complexity – When a node dies, simply replace the whole thing
- Cost-effective by using commercial off-the-shelf parts (PCs)
  - Economy of scale
  - No special equipment (e.g., SAN)

Chris von Nieda, “How Does Google Work,” 2010

Hyper-Converged Infrastructure

- Still (relatively) homogenous units of compute, memory, storage
- Each unit is virtualized, disaggregated via software
  - E.g., storage is accessed as a pool as if on a SAN
  - Each unit can be scaled independently
  - A cloud VM can be configured to access an arbitrary amount of virtual storage
  - Example: vmware vSAN

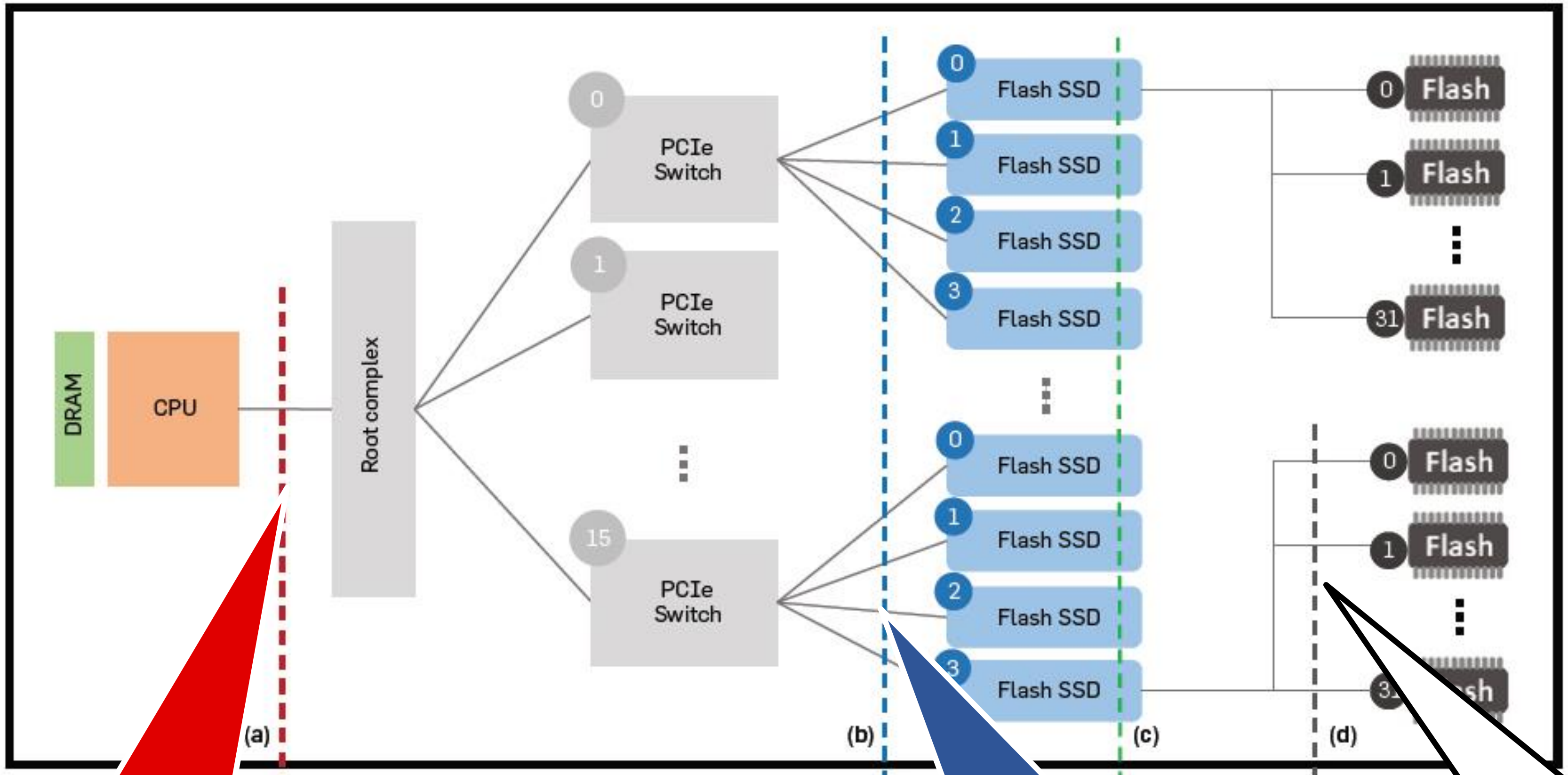
Object Storage

- Instead of managing content-oblivious blocks, the file system manages objects with their own metadata
  - Instead of directory/file hierarchies, each object addressed via global identifier
  - Kind of like key-value stores, in fact, the difference is ill-defined
  - e.g., Lustre, Ceph object store
- An “Objest Storage Device” is storage hardware that exposes an object interface
  - Still mostly in research phases
  - High level semantics of storage available to the hardware controller for optimization

Computational Storage

- Offloading computation to an engine on the storage device
- Why?
  - Modern SSDs have significant amount of embedded computation capacity (often 4 or more ARM cores), but they are not always busy
  - Some problems are latency dependent, and moving data all the way to CPU harms performance
  - The host-storage link becomes a bandwidth bottleneck with enough storage devices (4x 4-lane PCIe SSD saturates a 16 lane PCIe root complex)
    - Plus, peak internal bandwidth of a storage device is typically faster than the link bandwidth
  - Moving data to CPU consumes a lot of power

Bandwidth Bottlenecks In Storage



16 lanes PCIe Gen 3 = ~16 GB/s

8x Bandwidth Gap!

64 SSDs x ~2 GB/s = ~128 GB/s

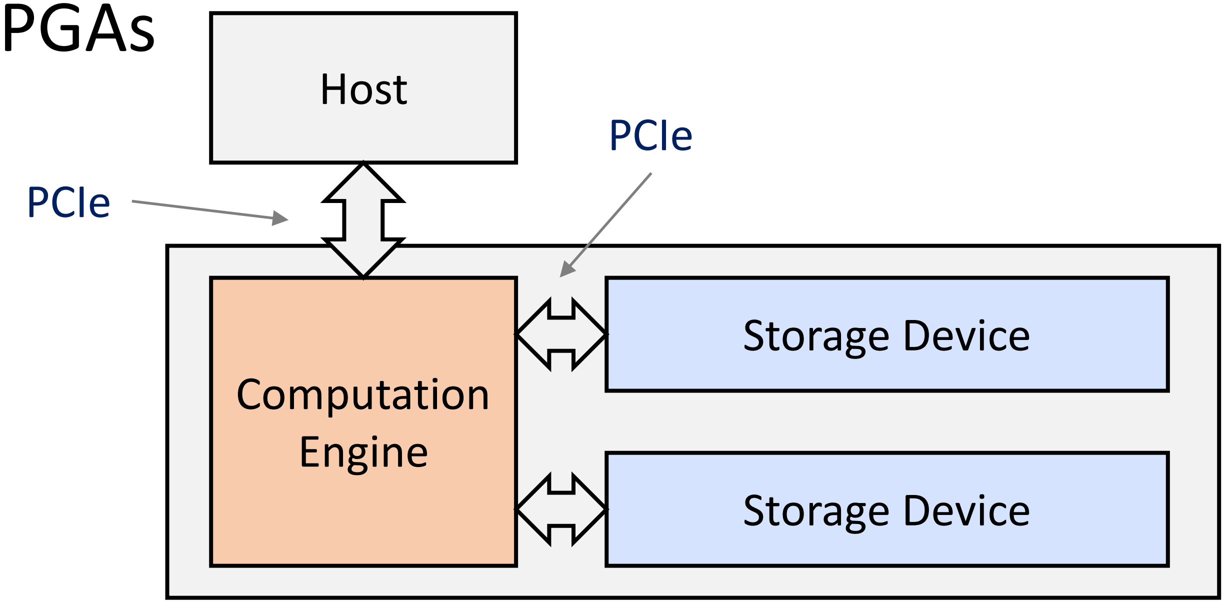
Internal bandwidth faster than its PCIe

2.5x Internal read BW (Source: Samsung)

Jaeyoung Do et.al., "Programmable Solid-State Storage in Future Cloud Datacenters," Communications of the ACM 2019

Typical Computational Storage Architecture

- Computation engine typically function both as PCIe endpoint (to host) and root complex (to storage devices)
- FTL May exist on each storage device (off-the-shelf), or computation engine (open channel, or raw chips)
- Computation may be ARM cores, FPGAs or something else
  - Some storage devices boot Linux!



Some Available Devices

- Many come with near-data FPGA acceleration
  - High-performance computation, still within the storage power budget
  - < 10W assigned to computation (PCIe power limitations, etc)

EMC Dragonfire board

BittWare 250S+

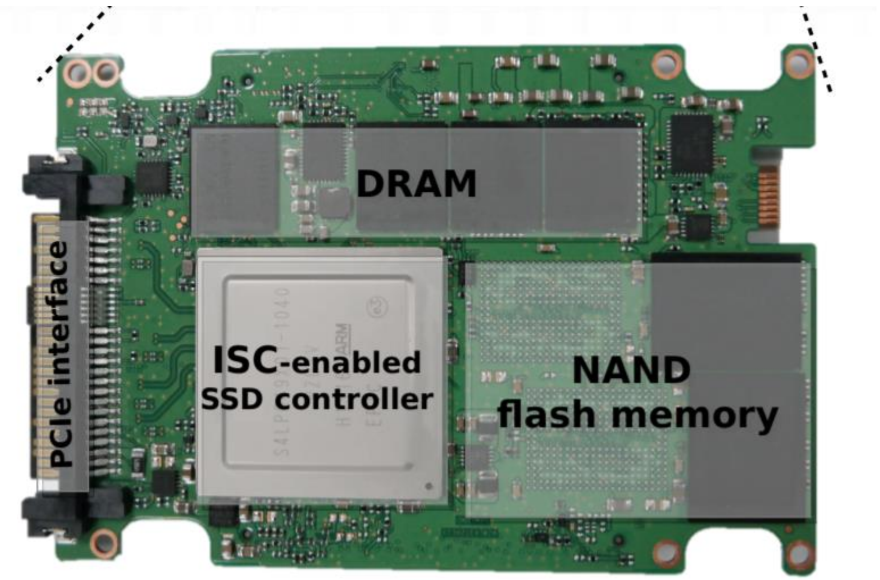
Some Points

- No standard interface or programming model yet
  - All existing implementations have custom interfaces, with varying levels of access abstraction
    - Block devices (transparent FTL), raw chip access, etc
  - Storage Networking Industry Association (SNIA) Computational Storage working group just created (2018)
- Accelerator cannot take advantage of page cache
  - Page cache exists on host, which it cannot access
  - Some database implementations saw even performance degradation because of this

Example – YourSQL

- “Early filtering” data in the storage to reduce amount of data sent to host
  - Offloads computation, saves link bandwidth
  - Query optimizer modified to move queries with low “filtering ratio” to an early position
  - Filtering ratio metric is storage aware, choosing queries that lower read page count instead of simple row count

Samsung PM1725



Jo et. al., “YourSQL: A High-Performance Database System Leveraging In-Storage Computing,” VLDB 2016

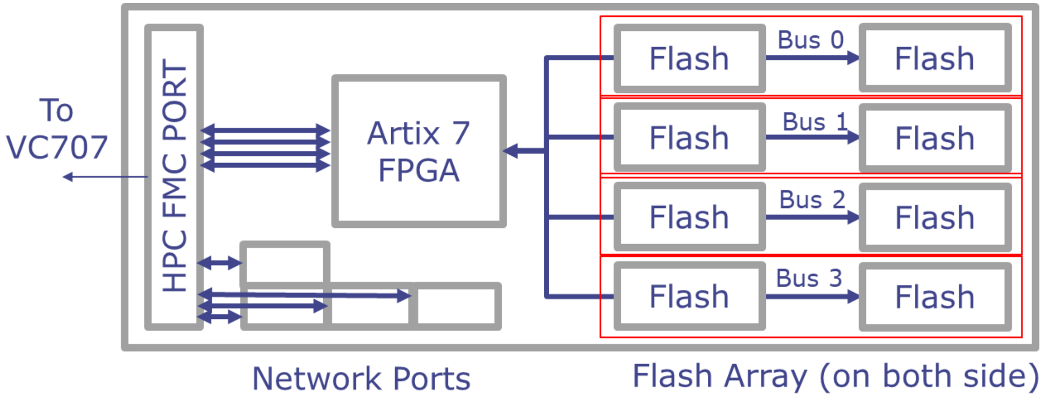
Example – YourSQL

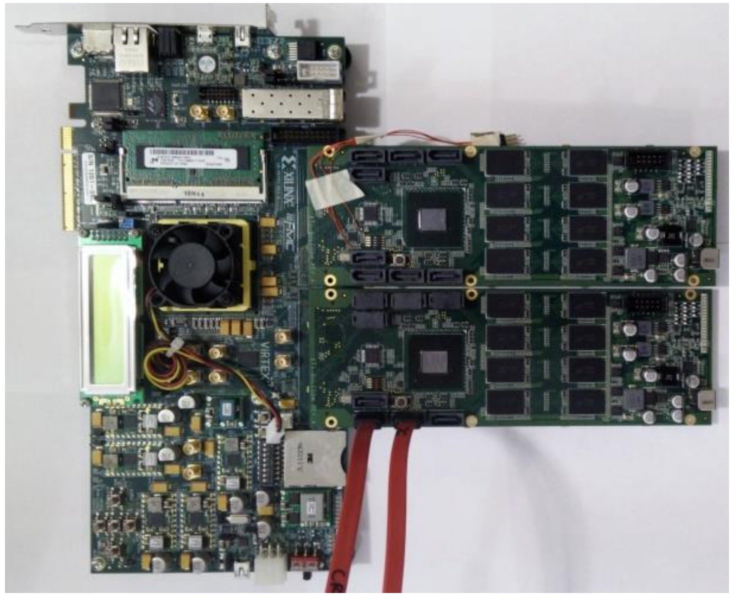
- Evaluation on 16-core Xeon, 64 GB memory, running MySQL
  - Near-storage compute has dual-core ARM R7
  - Query planner and storage engine significantly re-written
- Improves TPC-H benchmark by 3.6x over baseline
  - Most improved query improved by 15x
- Query type 1: Selection improved 7x
  - Storage bandwidth used inefficiently in baseline MySQL
- Query type 2: Join improved 40x
  - Size of joined tables reduced by early filtering
    - Baseline not fitting in memory?

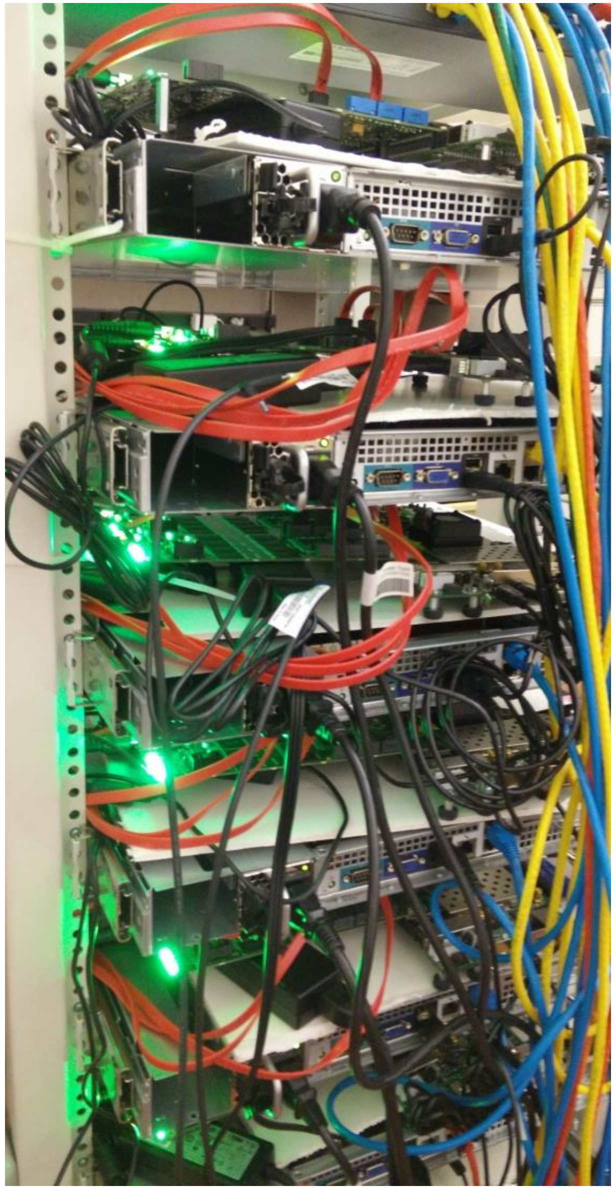
Jo et. al., "YourSQL: A High-Performance Database System Leveraging In-Storage Computing," VLDB 2016

Example – BlueDBM

- Research prototype at MIT (2015) for distributed computational storage
  - 20-node cluster, 20 Virtex 7 FPGAs, total 20 TB flash
  - Each virtex 7 FPGA networked directly to each other via low-latency serial links
    (8x 10 Gbps per link)

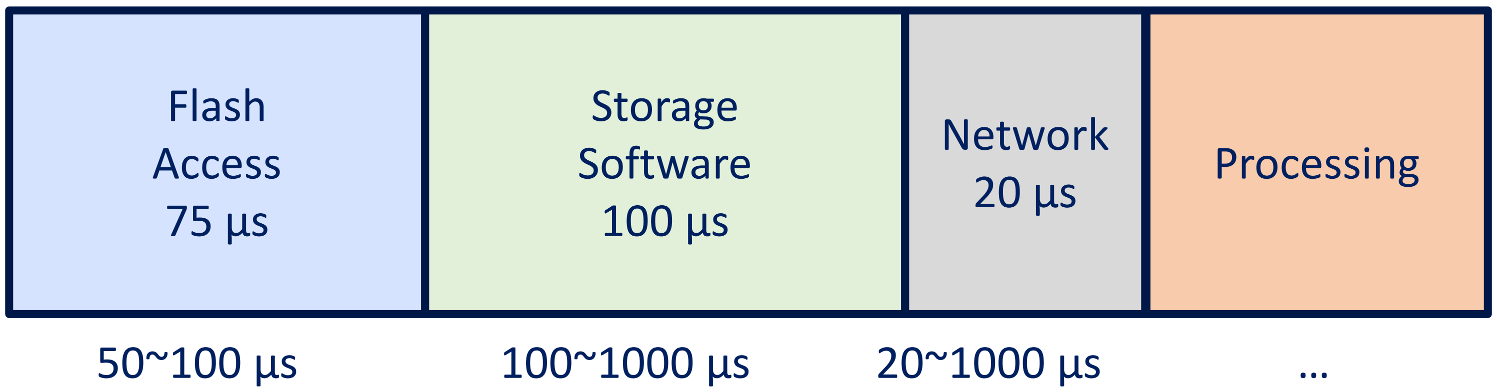






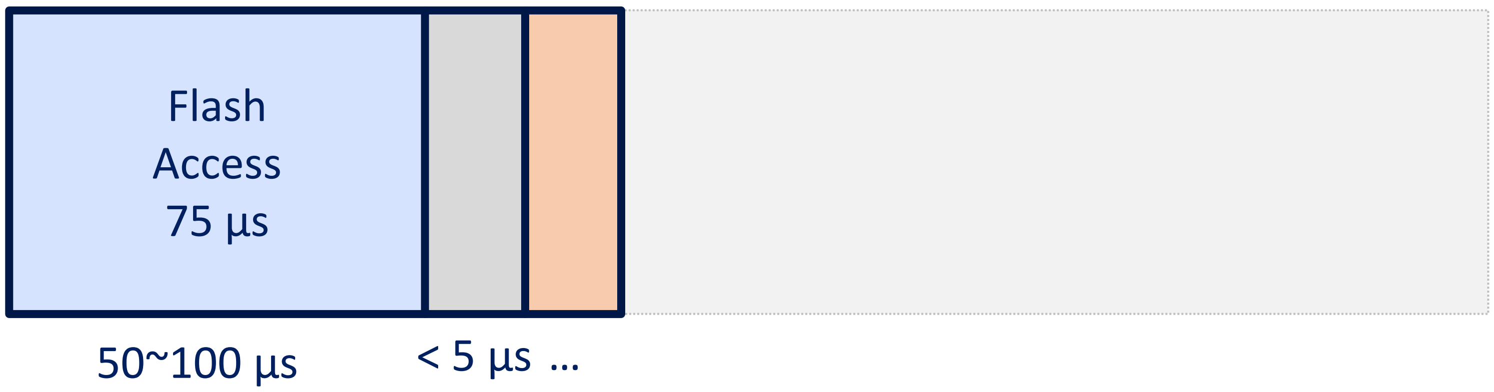
Latency Profile of Analytics on Distributed Flash Storage

- Distributed processing involves many system components
  - Flash device access
  - Storage software (OS, FTL, …)
  - Network interface (10gE, Infiniband, …)
  - Actual processing



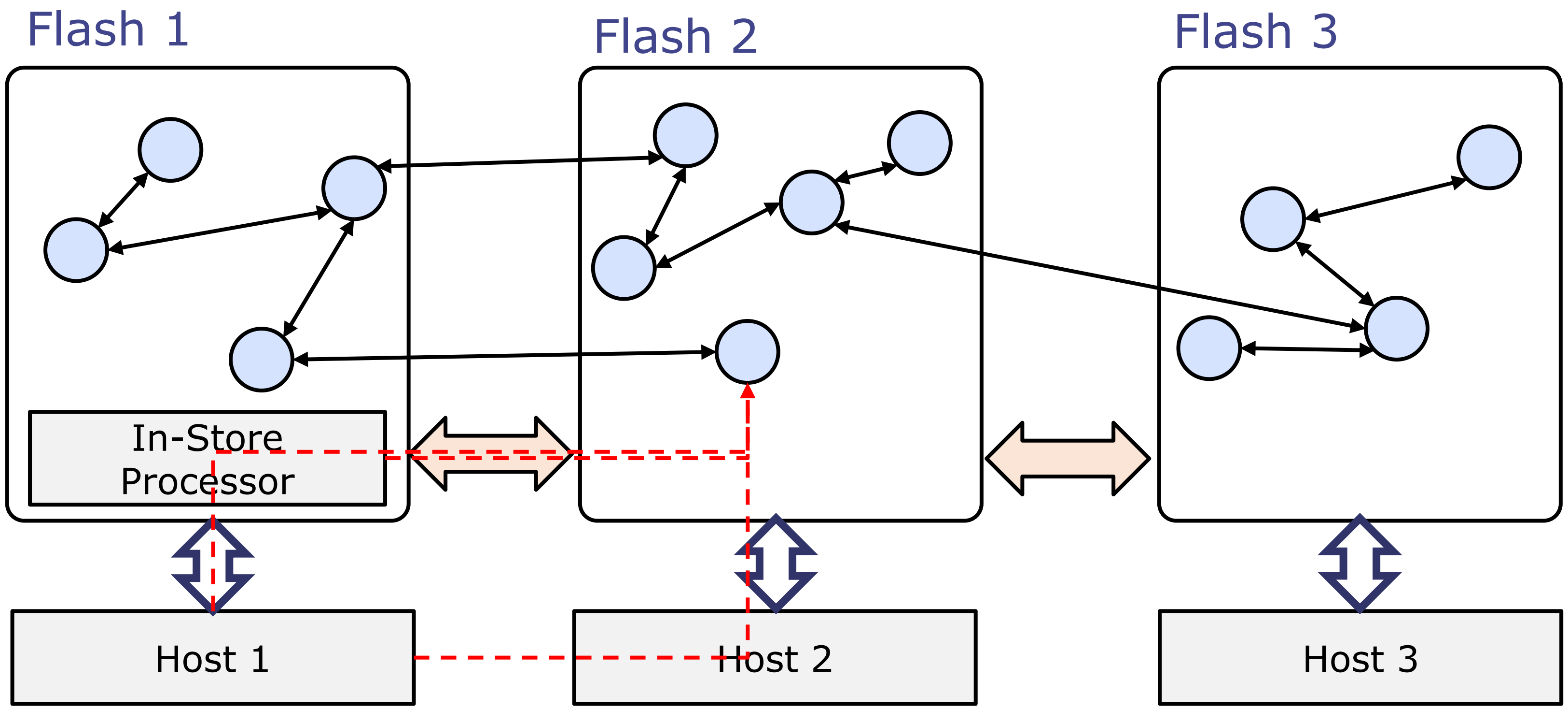
Latency Profile of Analytics on Distributed Flash Storage

- Architectural modifications can remove unnecessary overhead
  - Near-storage processing
  - Cross-layer optimization of flash management software*
  - Dedicated storage area network
  - Computation Accelerator



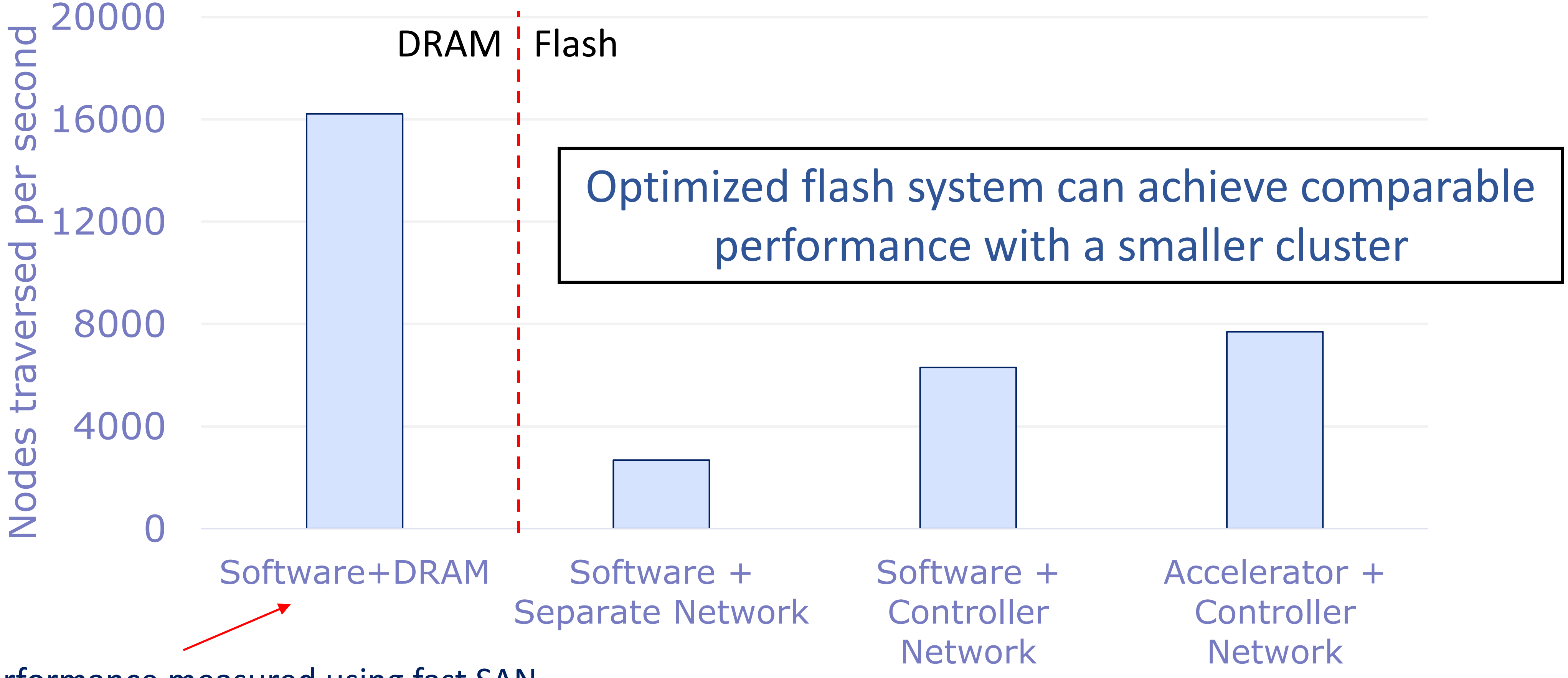
Latency-Emphasized Example – Graph Traversal

- Latency-bound problem because the next node to be visited cannot be predicted
  - Completely bound by storage access latency in the worst case



Latency improved by
1. Faster SAN
2. Near-Storage Acceleraor

Latency-Emphasized Example – Graph Traversal



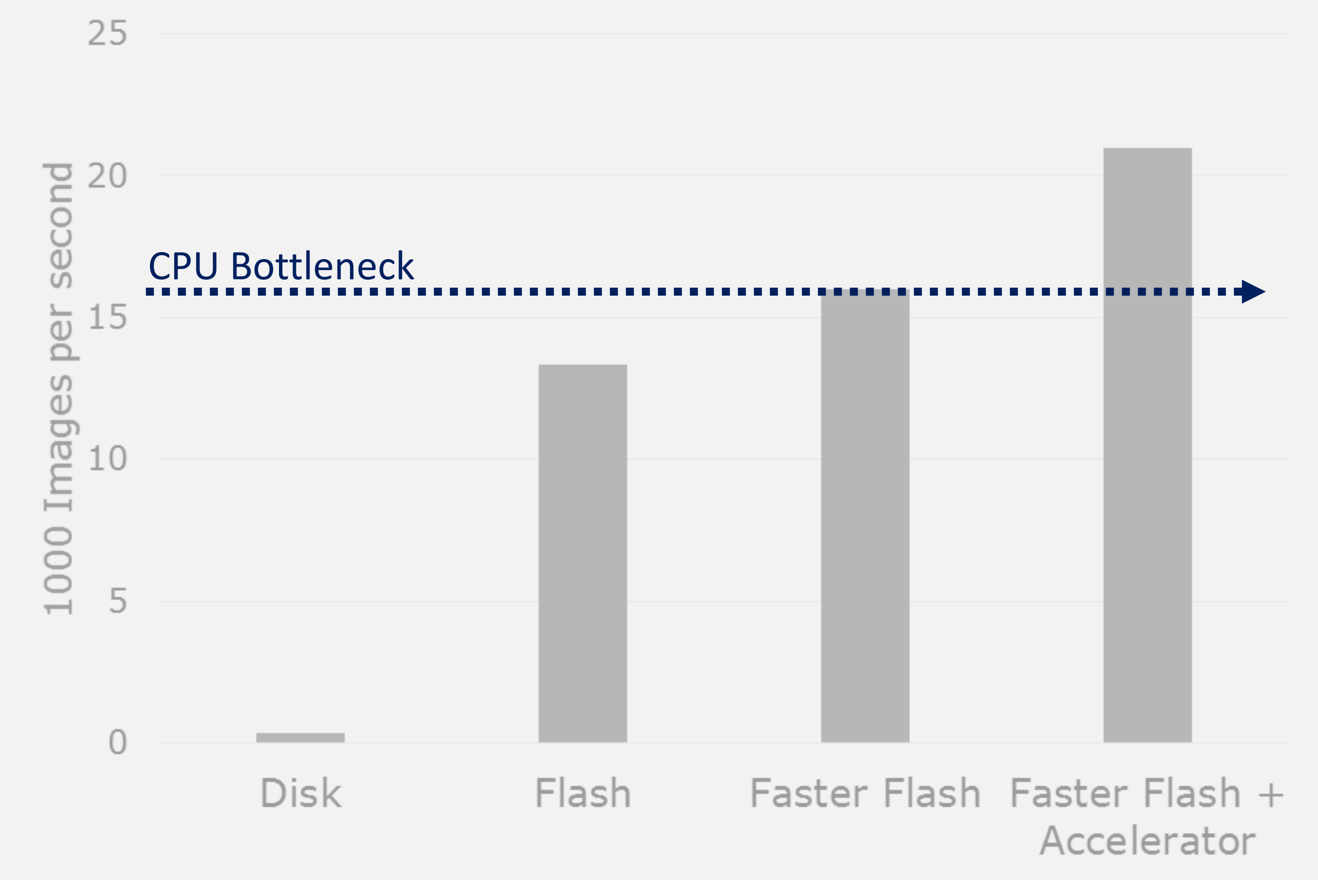
Software performance measured using fast SAN

Acceleration-Emphasized Example -- High-Dimensional Search

- Curse of dimensionality: Difficult to create effective index structure for high-dimensional data
  - Typically, index structure reduces problem space, and direct comparison against remaining data
  - Low locality between queries → Caching ineffective → Everything comes from storage anyways → Storage good place for accelerator
- Computation naturally scales as more storage is added

Acceleration-Emphasized Example -- High-Dimensional Search

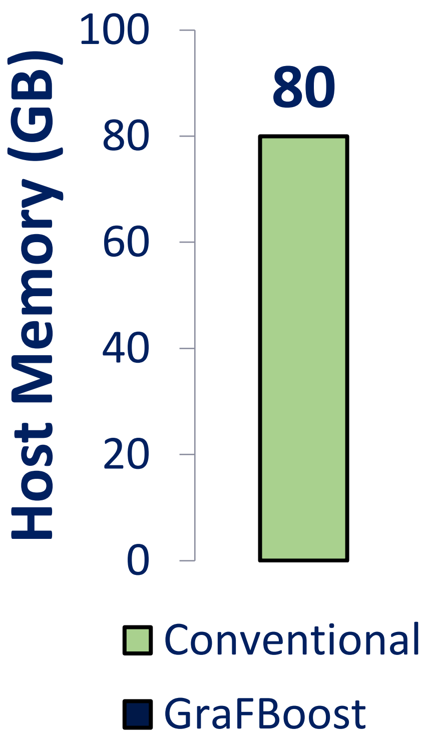
- Image similarity search example
  - Effective way to overcome CPU performance bottleneck
  - Much lower power consumption thanks to FPGA



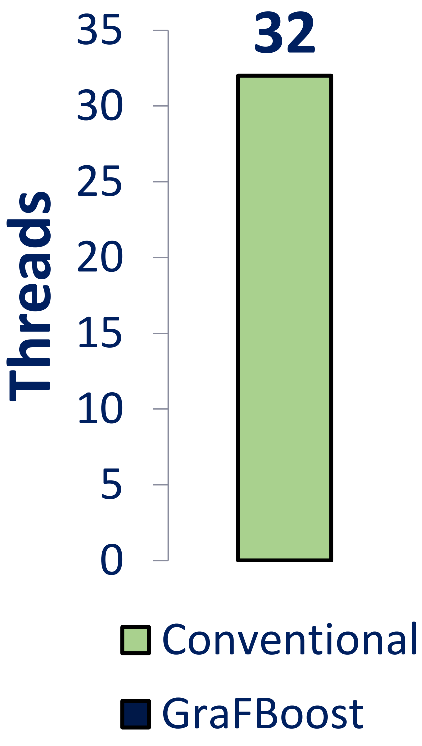
A More Complex Example -- Graph Analytics

- Graph algorithms are often random-access intensive
  - Cannot predict which vertex to visit next, before processing the current one
  - Search, Statistical analytics, Subgraph isomorphism…
  - Requires fast random-access into TBs of memory
    - Large, multi-TB machine, or distributed systems with fast networking
- Algorithmic changes required to make access amenable to storage
  - Coarse granularity, high latency, but acceptable bandwidth
  - New algorithms increased computational overhead, offloaded to FPGAs

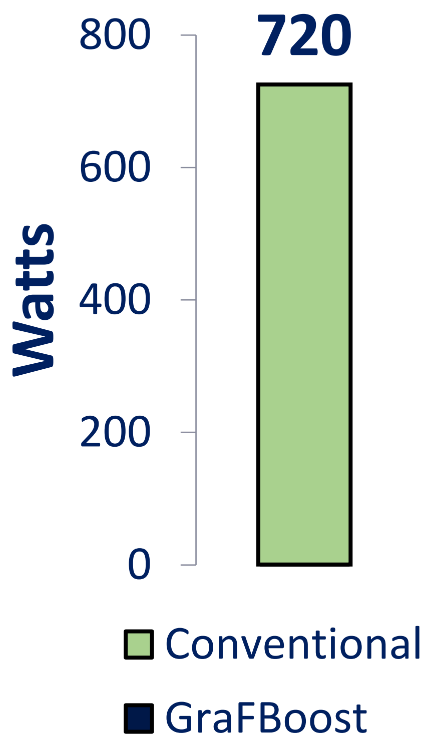
A More Complex Example
-- Graph Analytics -- GraFBoost (2018)



External analytics



Hardware Acceleration



External Analytics
+ Hardware Acceleration