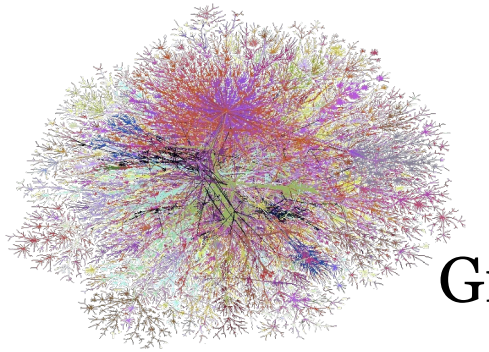
Disaggregated & Heterogeneous Platform for Data Management

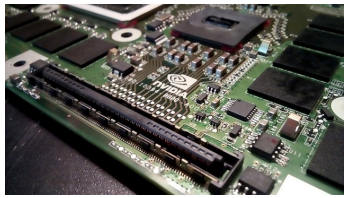
M. Tamer Özsu
Cheriton School of Computer Science
tamer.ozsu@uwaterloo.ca

UNIVERSITY OF WATERLOO | DSG Data Systems Group

My research

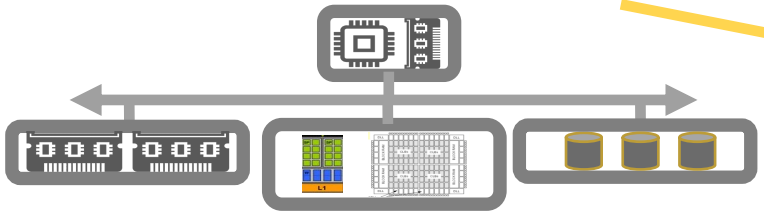


Graph Processing

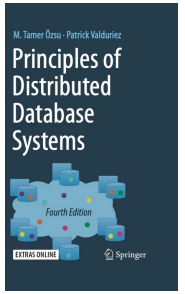


Graph Processing On GPUs[1,2]

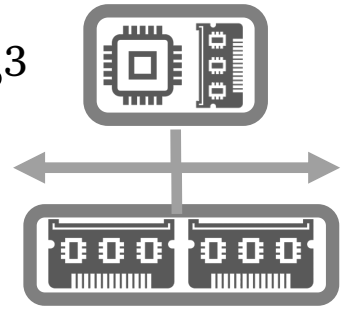
Graph Processing on Heterogeneous Platform



Distributed/Parallel Data Management



Disaggregated Systems[3]

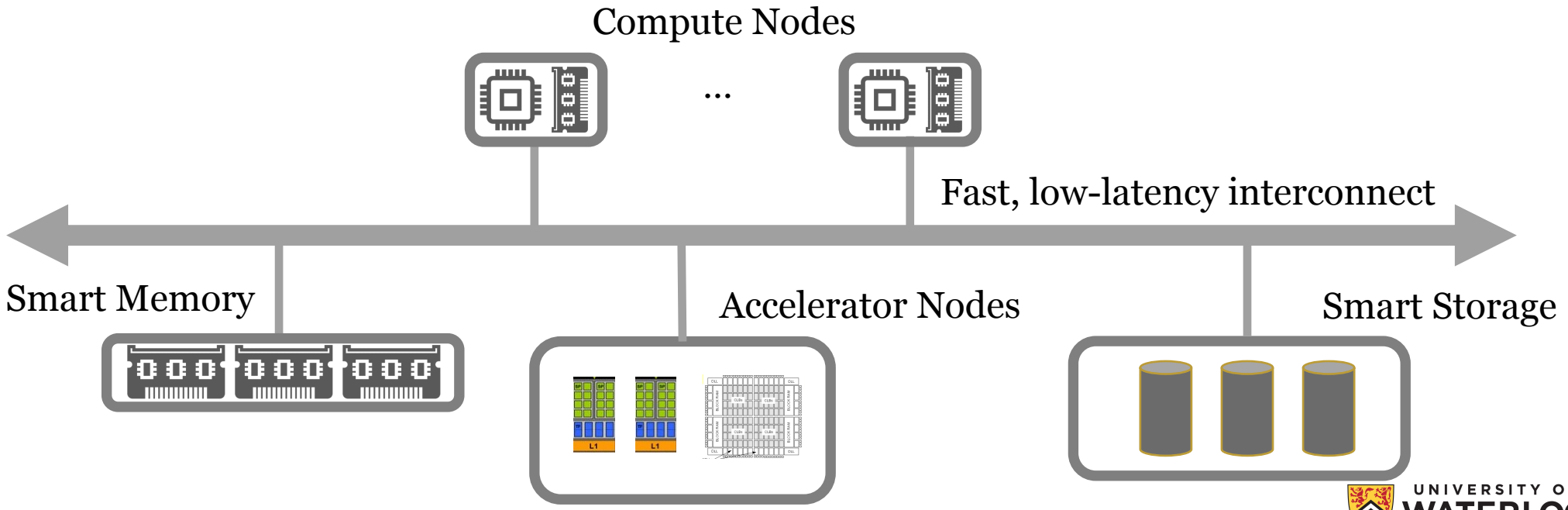


Graph Processing on Disaggregated Heterogeneous Platform

[1] L. Hu, L. Zou and M. T. Özsu. GAMMA: A Graph Pattern Mining Framework for Large Graphs on GPU, *Proc. ICDE*, 2023.
[2] L. Zeng, et al. GSI:GPU-friendly Subgraph Isomorphism, *Proc. ICDE*, 2020.
[3] R.Wang et al. The Case for Shared-Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Disaggregated Heterogeneous Platform

- Disaggregated
  - Separate components with a fast interconnect
- Heterogeneous
  - "Executing programs on a computing platform with computing nodes of different characteristics."[1]



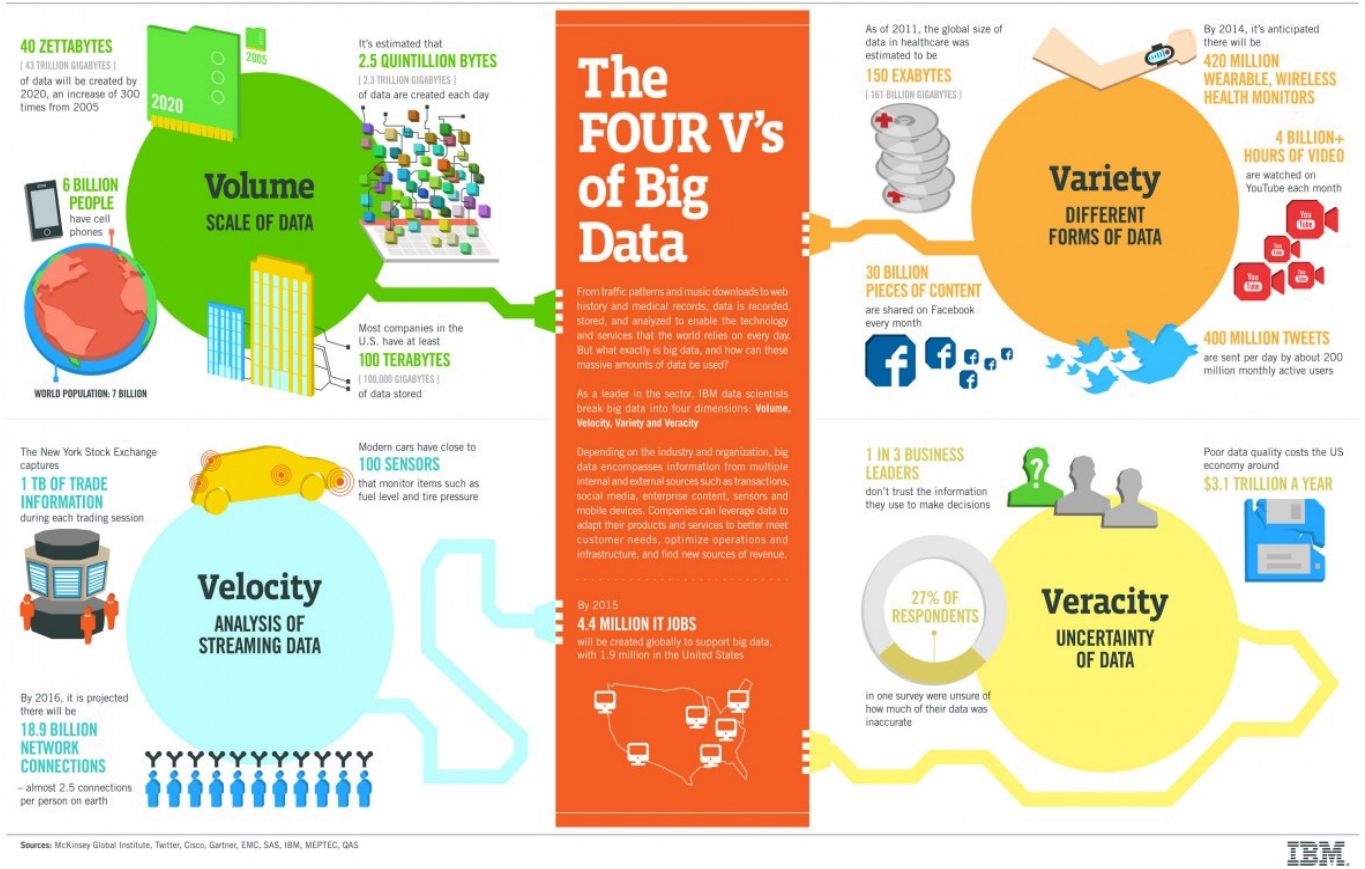
[1] M. Zahran. *Heterogeneous Computing*, ACM Books, 2019.

Why?

New Application Demands Technological Changes

New Application Demands

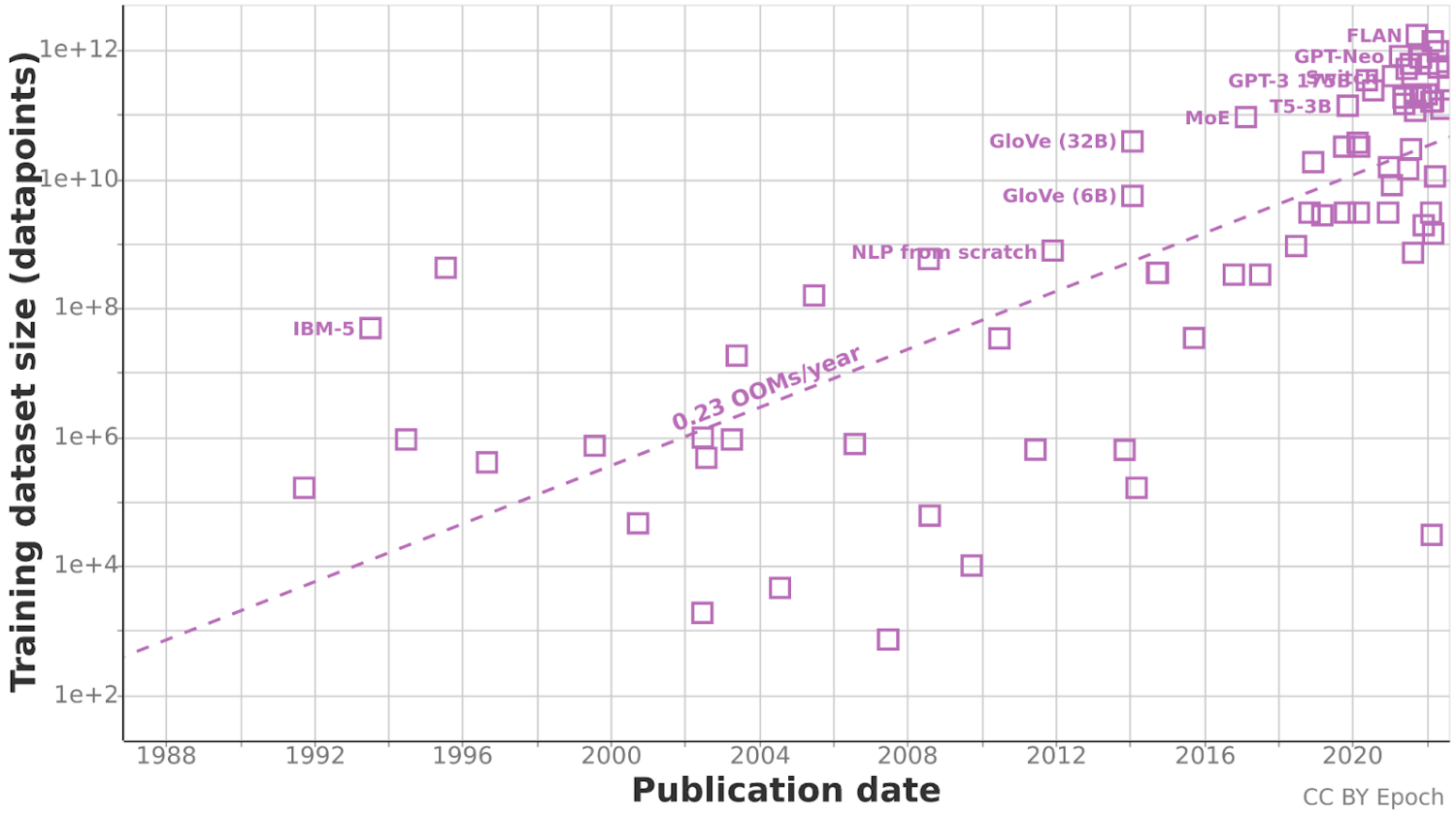
Big data applications



New Application Demands

Big data applications

- Very large storage
  - 10x rule in ML workloads



https://epochai.org/blog/trends-in-training-dataset-sizes

New Application Demands

Big data applications

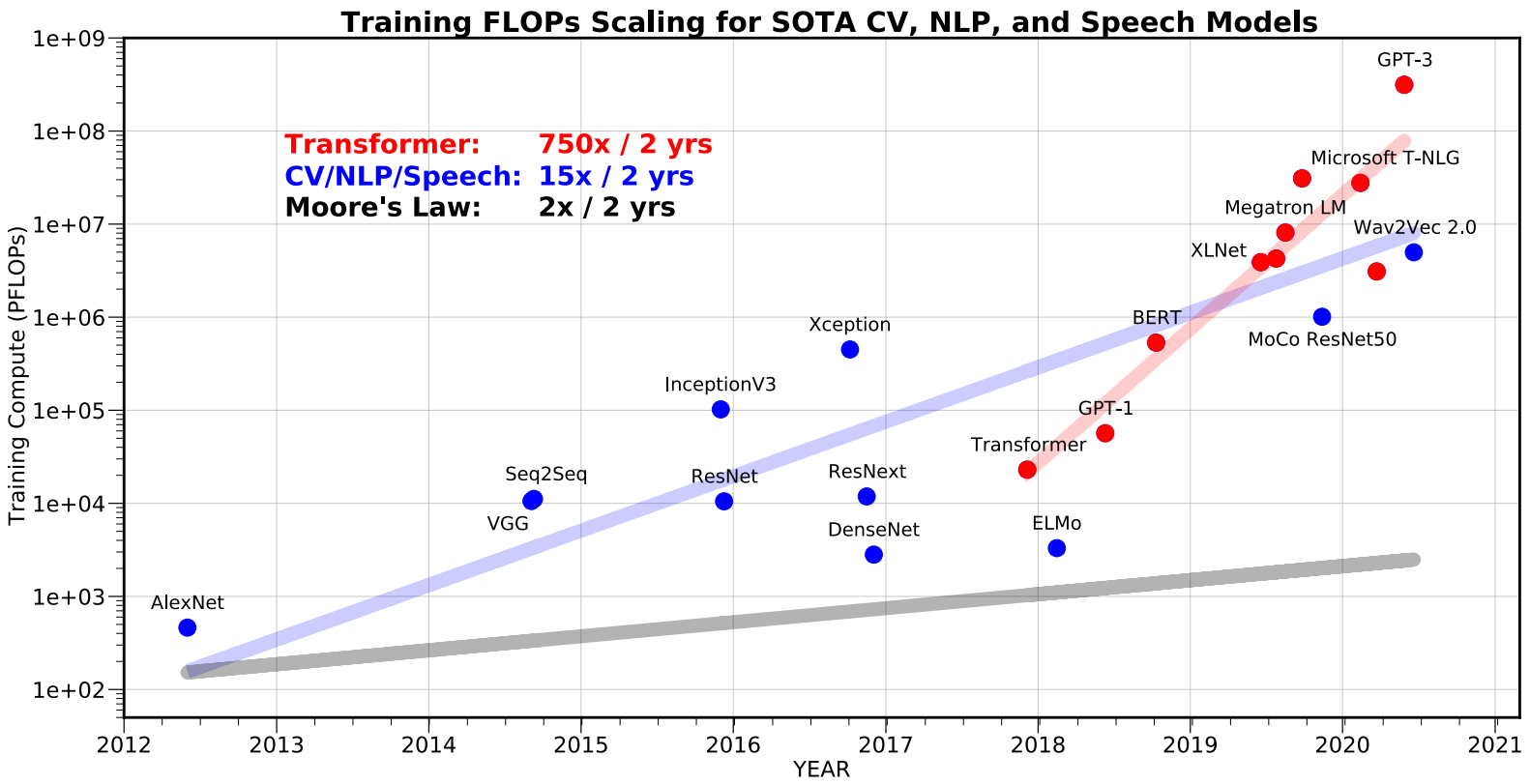
- Very large storage
  - 10x rule in ML workloads
- Repetitive computation

```
class Vertex:
    def __init__(self, name):
        self.name = name
        self.children = []
        self.parents = []
        self.auth = 1.0
        self.hub = 1.0
        self.pagerank = 1.0

#One iteration computation
def PageRank_one_iter(graph, d):
    vertex_list = graph.vertices
    for vertex in vertex_list:
        vertex.update_pagerank(d, len(graph.vertices))

#Go over all vertices until convergence
def PageRank(graph, d):
    finished = False
    while not finished:
        PageRank_one_iter(graph, d)
        finished = converge()
```

New Application Demands

Big data applications

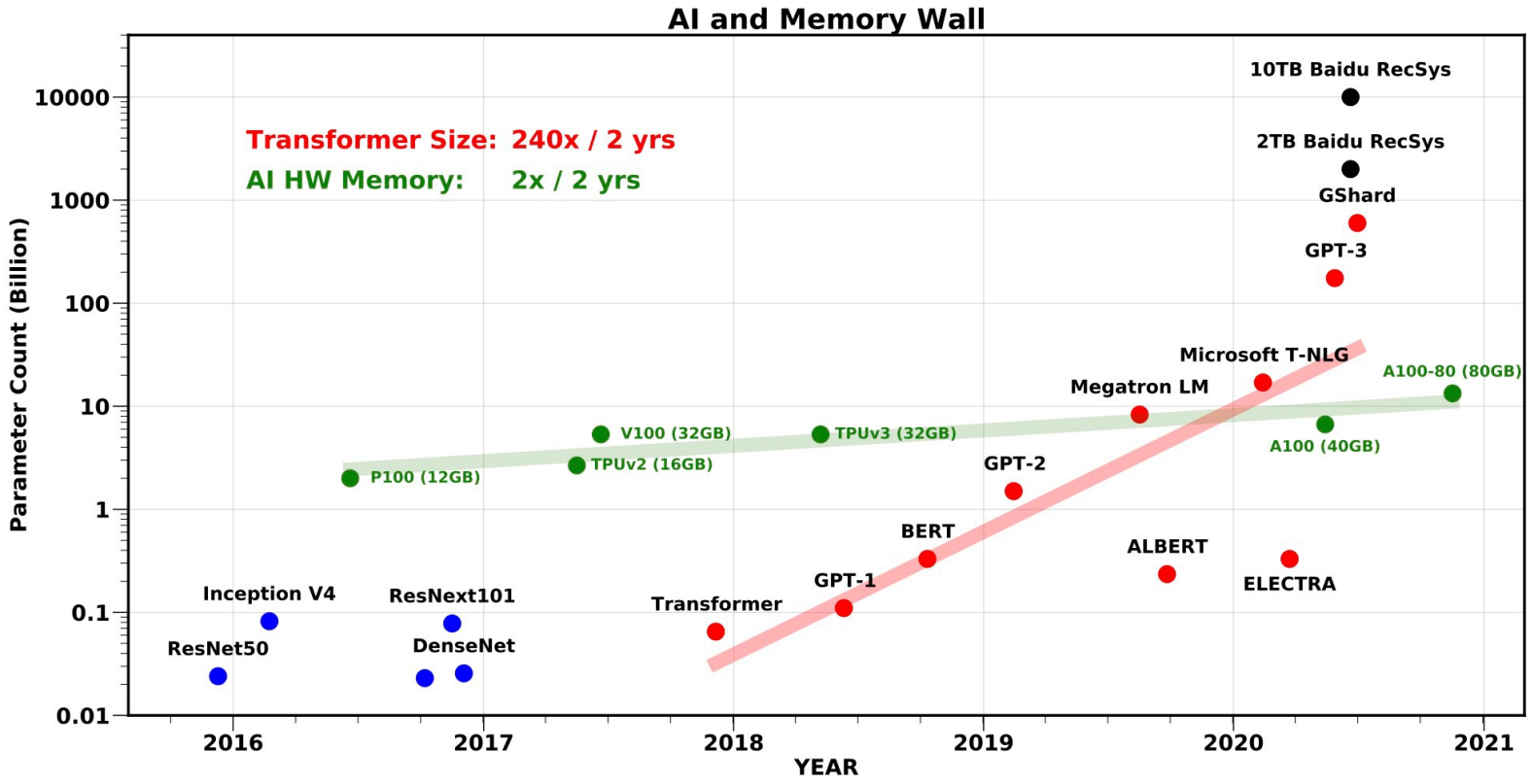
- Very large storage
  - 10x rule in ML workloads
- Repetitive computation
  - Excessive computing cycles



New Application Demands

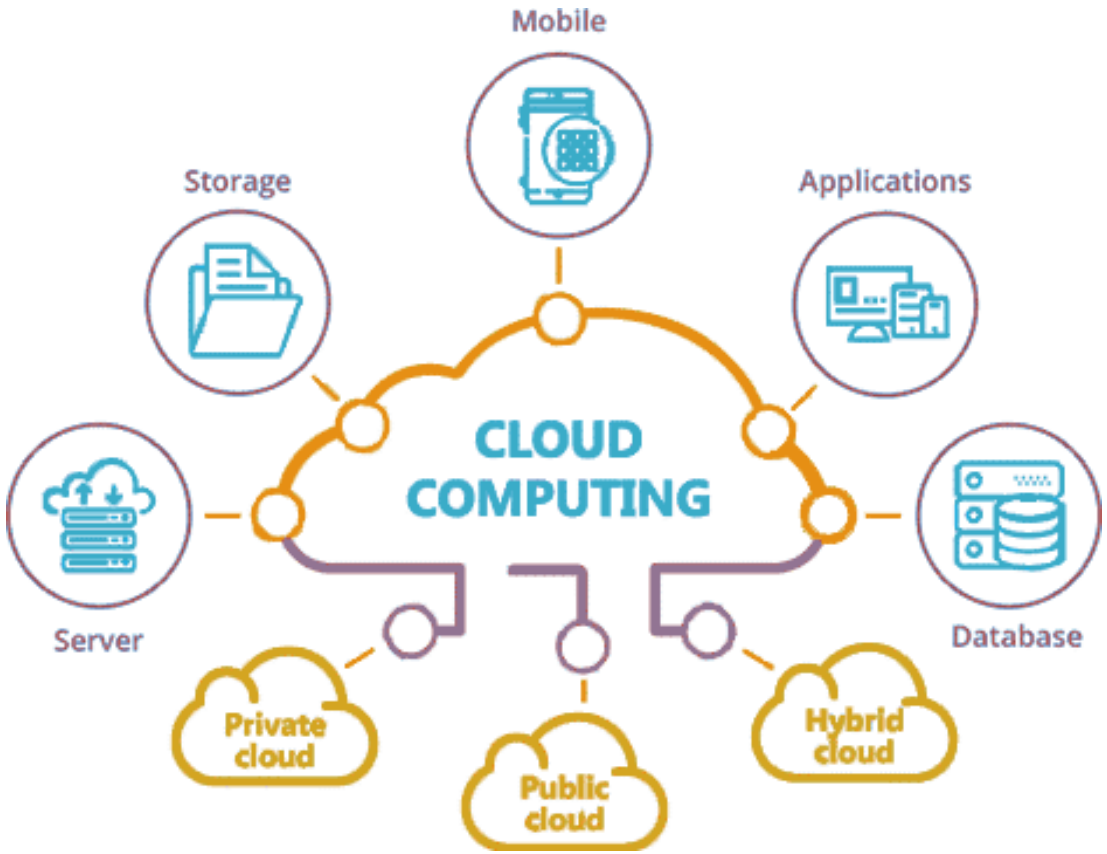
Big data applications

- Very large storage
  - 10x rule in ML workloads
- Repetitive computation
  - Excessive computing cycles
- Large memory



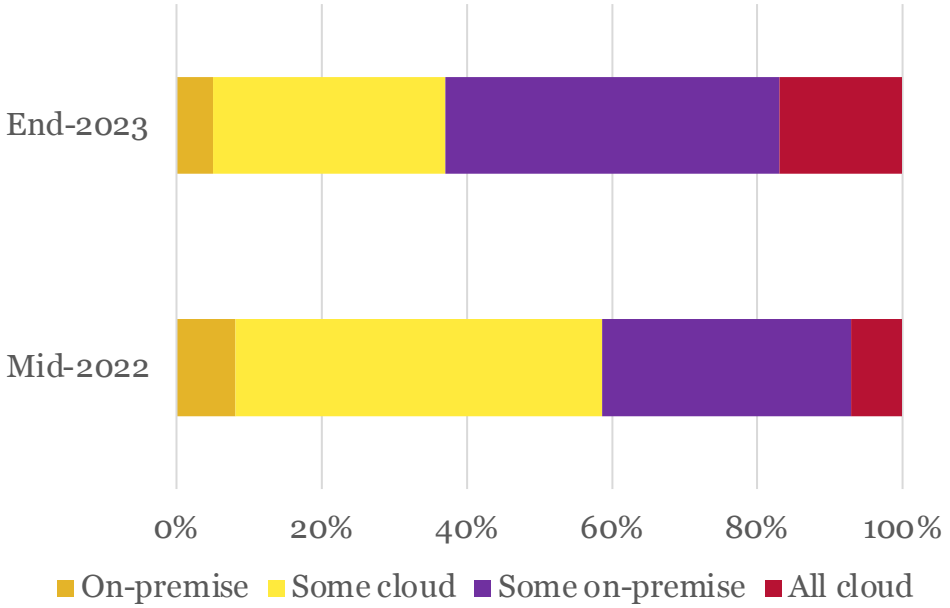
TECHNOLOGY CHANGES

Move to the Cloud

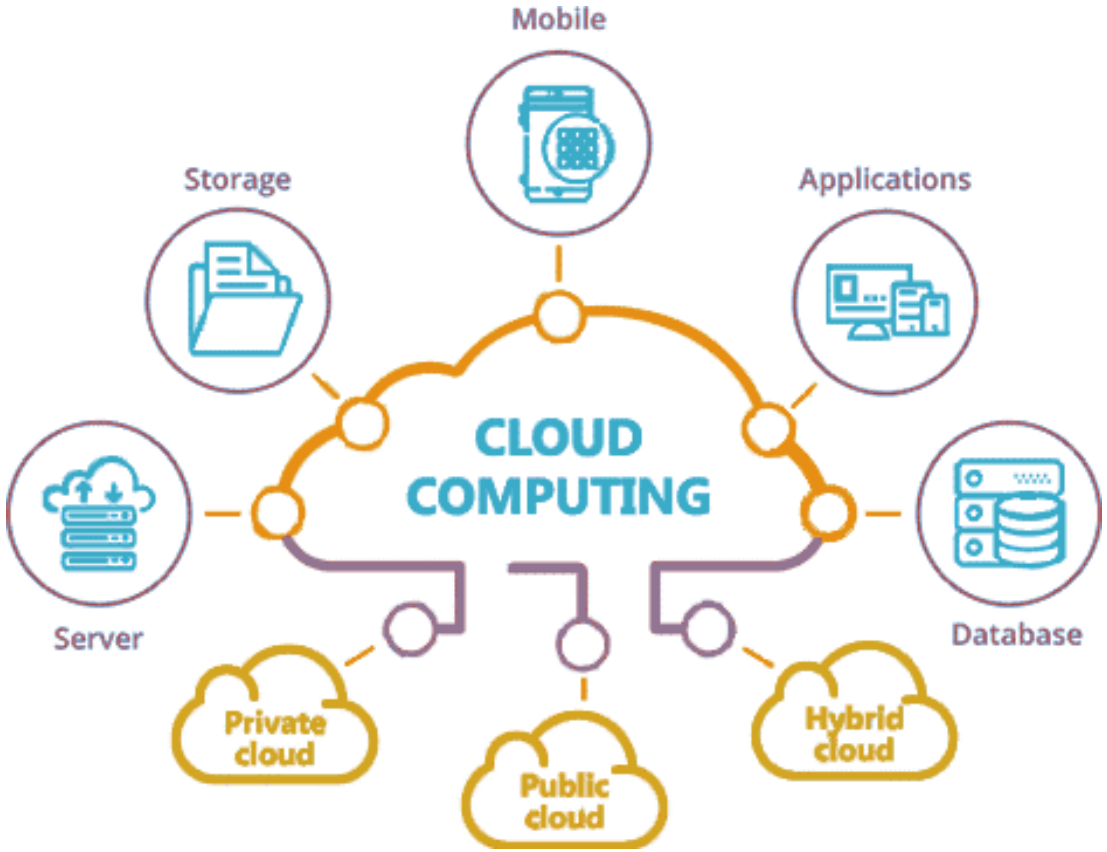


Move to the Cloud

- Move is expanding

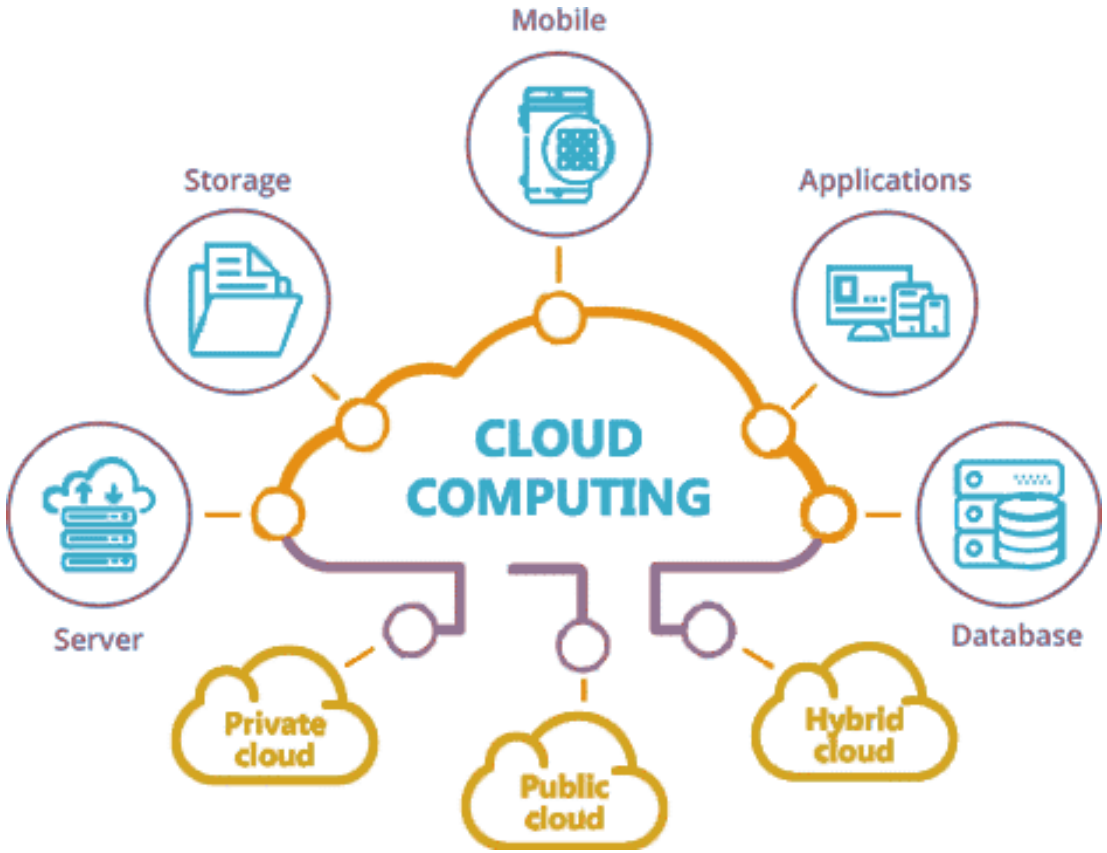


Foundry Cloud Computing Study, 2022



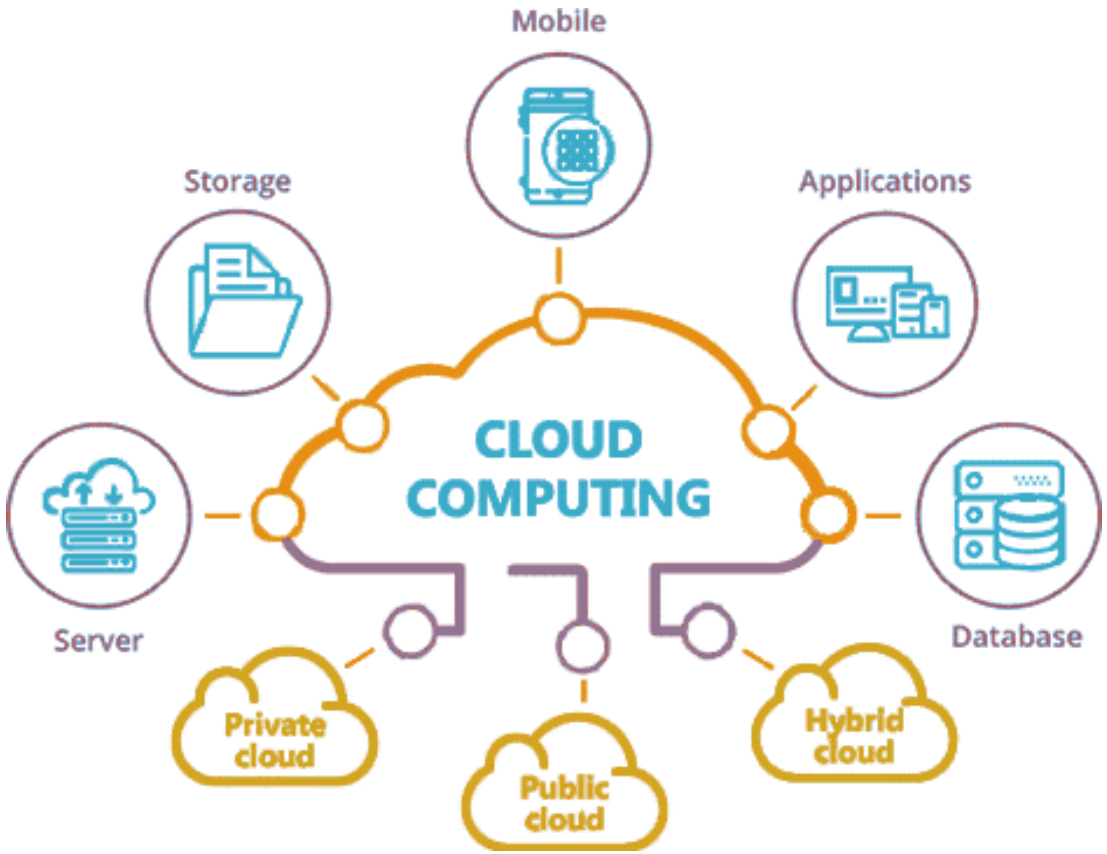
Move to the Cloud

- Move is expanding
- Reasons
  - Elasticity
  - Availability
  - Cost savings
  - …

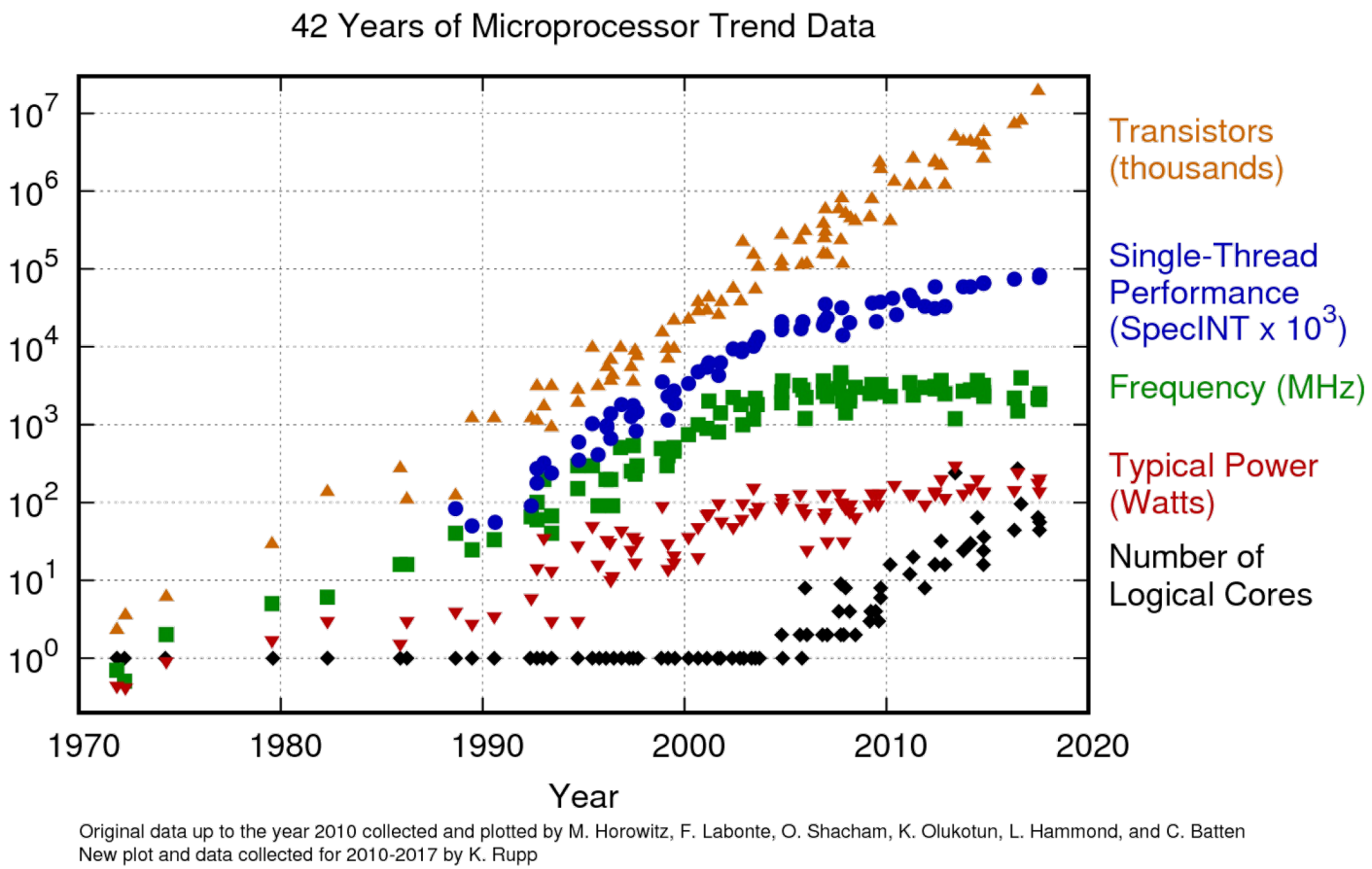


Move to the Cloud

- Move is expanding
- Reasons
  - Elasticity
  - Availability
  - Cost savings
  - …
- Stress points
  - Elasticity in demand
  - Configuration to meet SLAs
  - …



Is Traditional Processor Keeping Up?

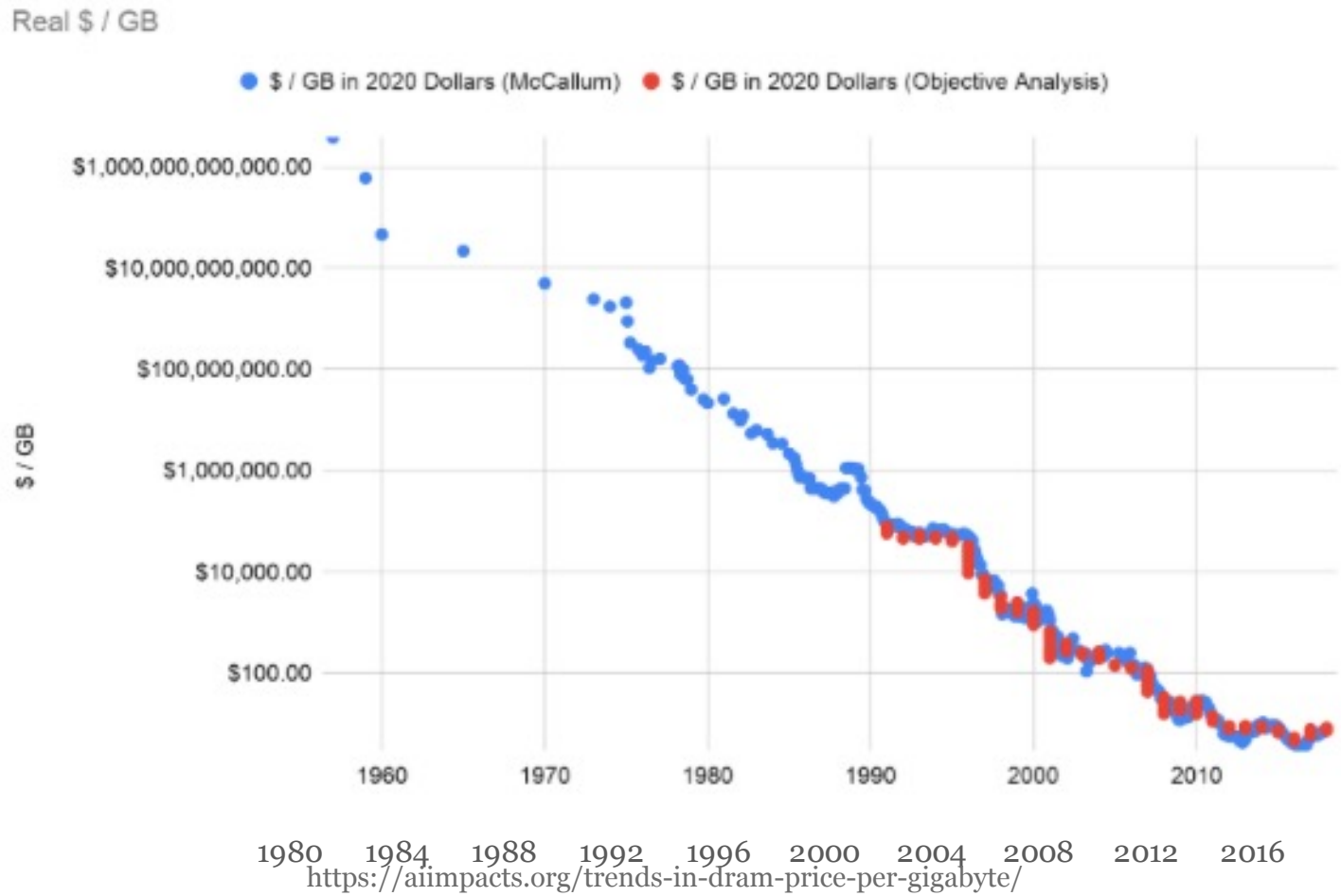


https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/

- Moore's Law application questionable
  - *The number of transistors on a microchip doubles every two years*
- Pollack's Rule
  - *Microprocessor performance increase due to microarchitecture advances is roughly proportional to the square root of the increase in complexity.*

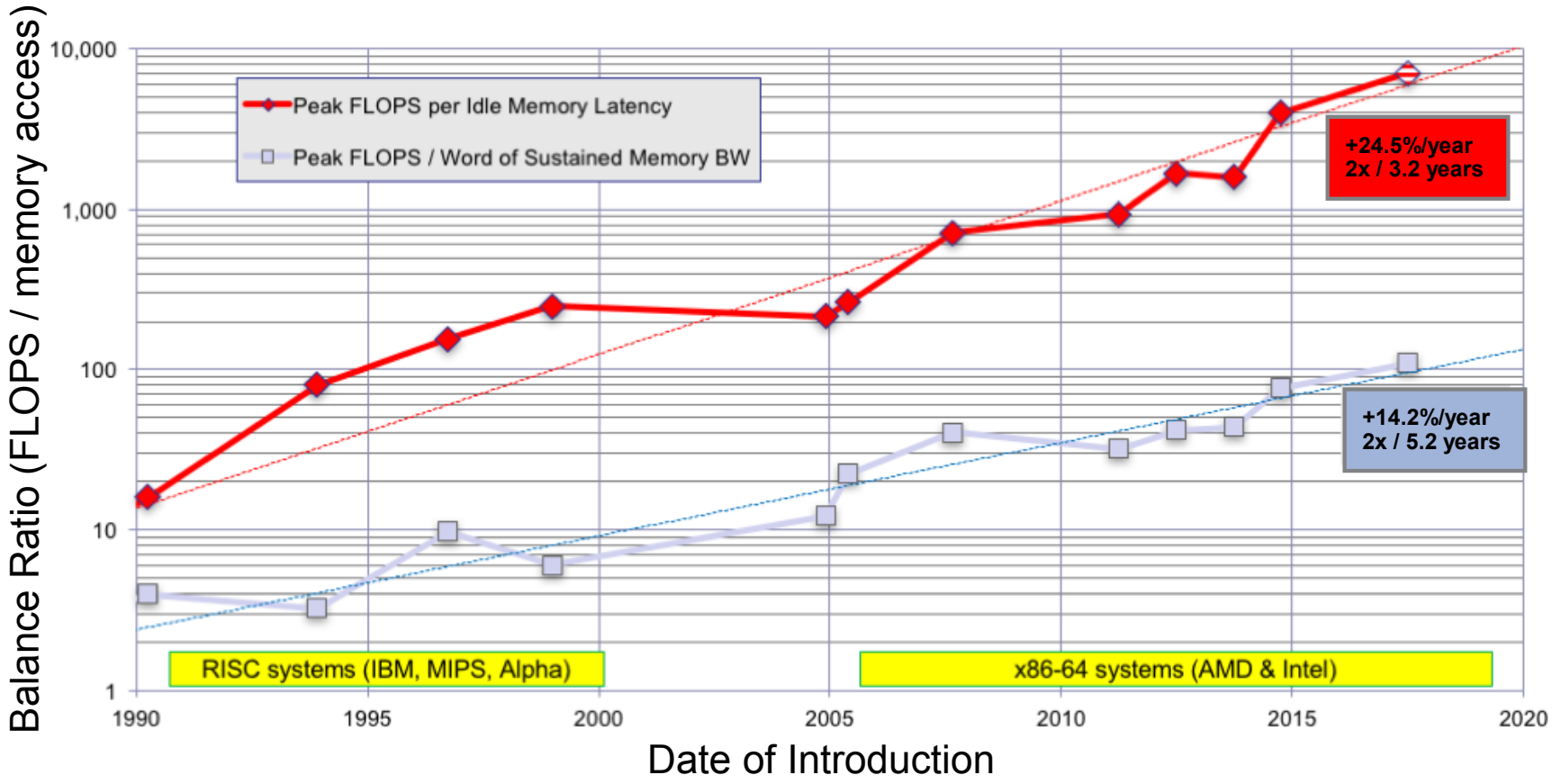
Changes in RAM

- RAM capacities increasing 10x every four years (?)
  - Not single memory chip
- RAM prices are going down
- Memory bandwidth increasing ~23% per year[1]
- Memory latency **increasing** ~4% per year[1]



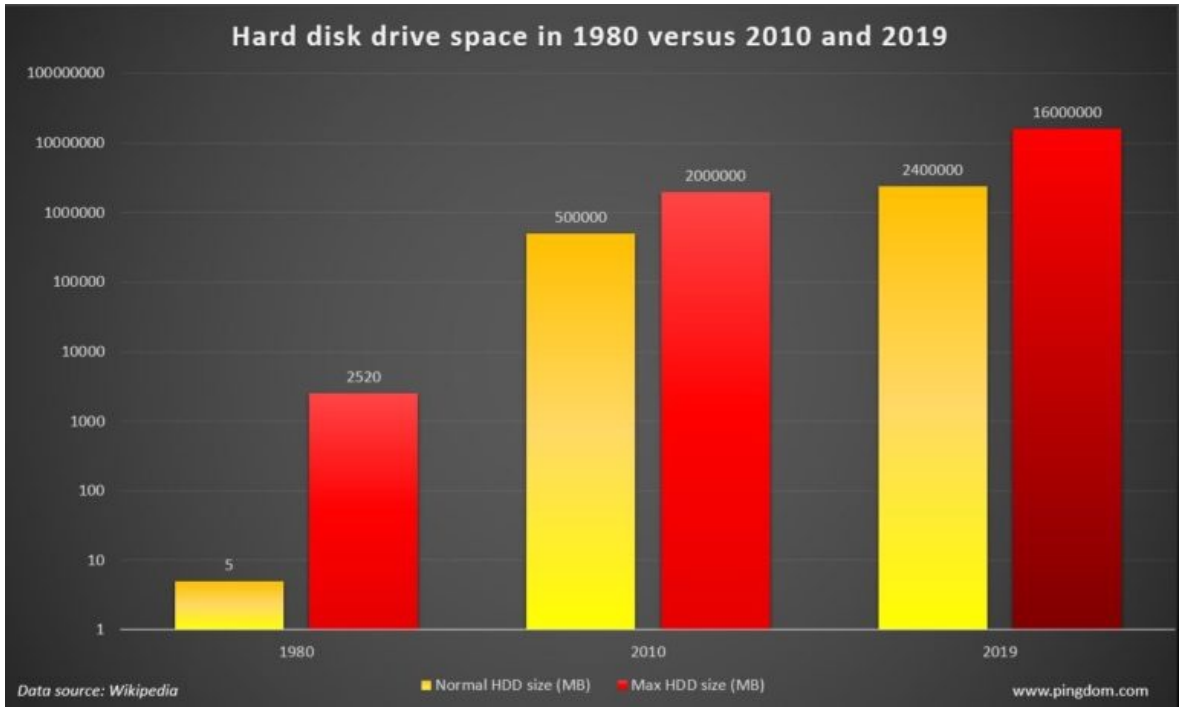
J. McCalpin. Memory Bandwidth and System Balance in HPC Systems: 2021 Update, Talk at Erlangen National High Performance Computing Center. https://hpc.fau.de/files/2021/12/memorybw_systembalance_slides_2021-12-15.pdf

Is The Memory Keeping Up?

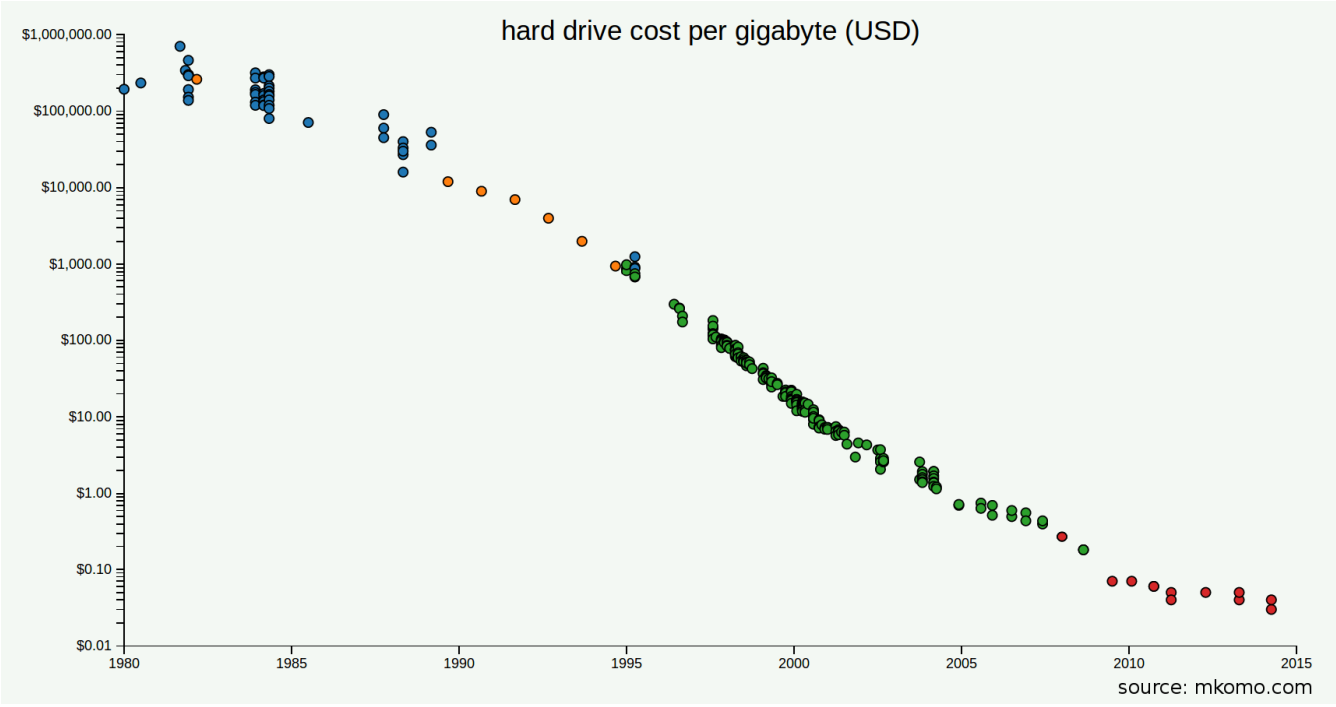


J. McCalpin. Memory Bandwidth and System Balance in HPC Systems, Invited talk at Supercomputing Symposium, 2016.
http://sites.utexas.edu/jdm4372/2016/11/22/sc16-invited-talk-memory-bandwidth-and-system-balance-in-hpc-systems/

Storage Capacity and Price Over Time



https://www.pingdom.com/



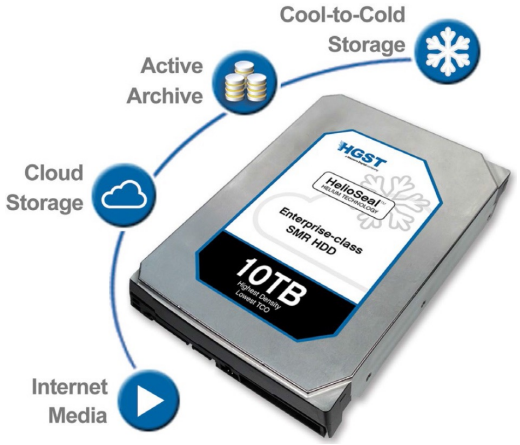
https://aiimpacts.com/

Storage Devices

- 10TB disks are about US$600
  - *Approximately 4,000 DVD movies,1.8M digital photos or 2.5M mp3 music files*

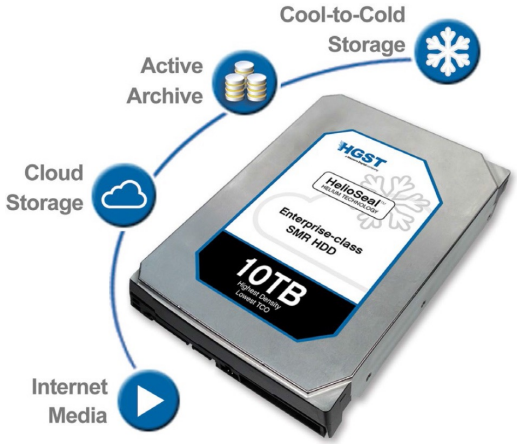
Storage Devices

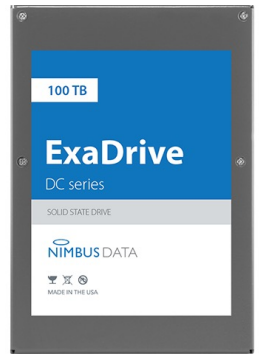
- 10TB disks are about US$600
  - *Approximately 4,000 DVD movies,1.8M digital photos or 2.5M mp3 music files*
- 10TB single drives are now available



Storage Devices

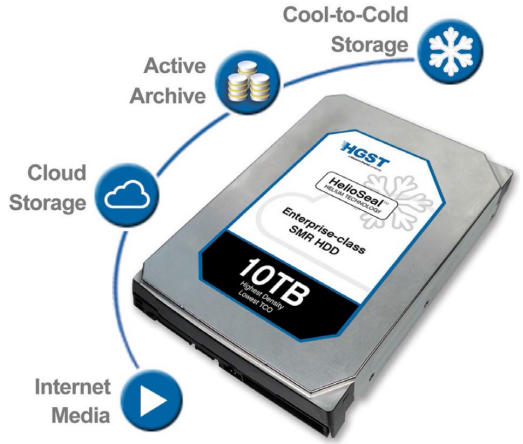
- 10TB disks are about US$600
  - *Approximately 4,000 DVD movies,1.8M digital photos or 2.5M mp3 music files*
- 10TB single drives are now available
- Nimbus ExaDrive DC100: 100TB SSD

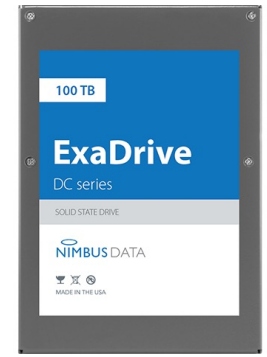


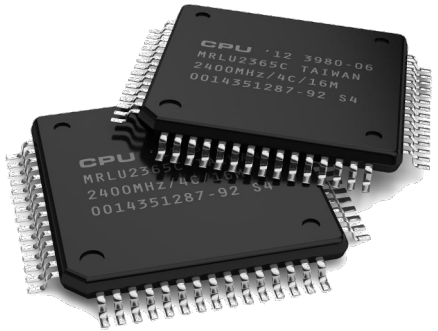


Storage Devices

- 10TB disks are about US$600
  - *Approximately 4,000 DVD movies,1.8M digital photos or 2.5M mp3 music files*
- 10TB single drives are now available
- Nimbus ExaDrive DC100: 100TB SSD
- Flash is now mainstream
  - 1TB flash is common
  - Different storage hierarchy

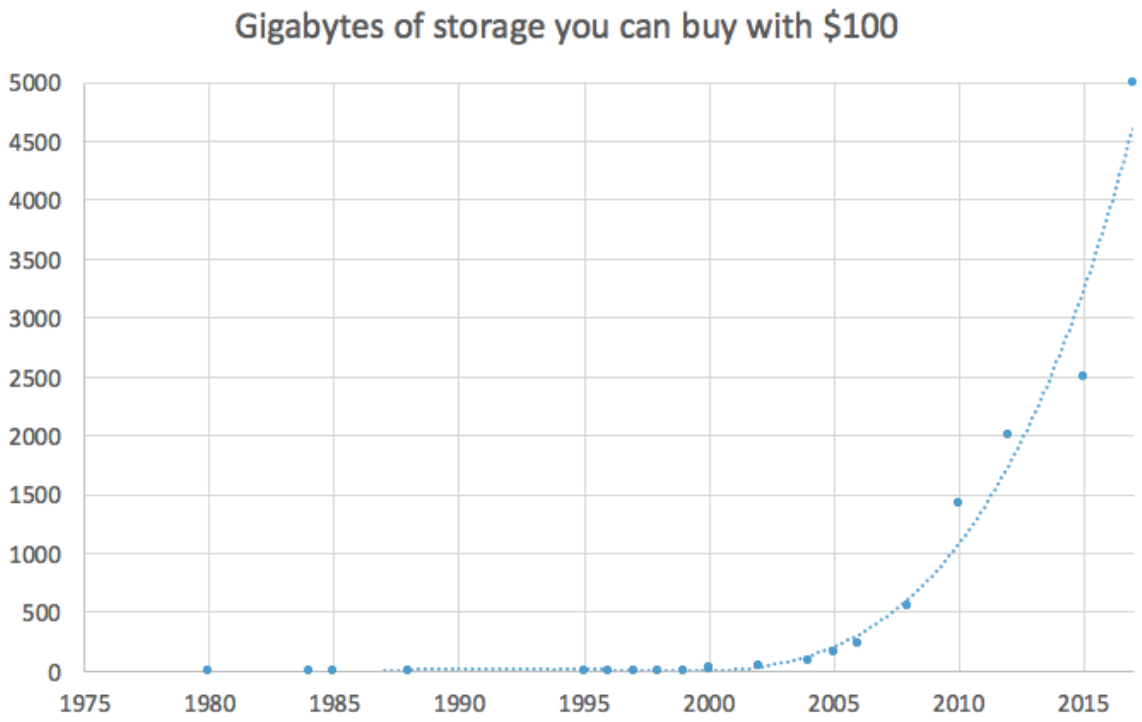




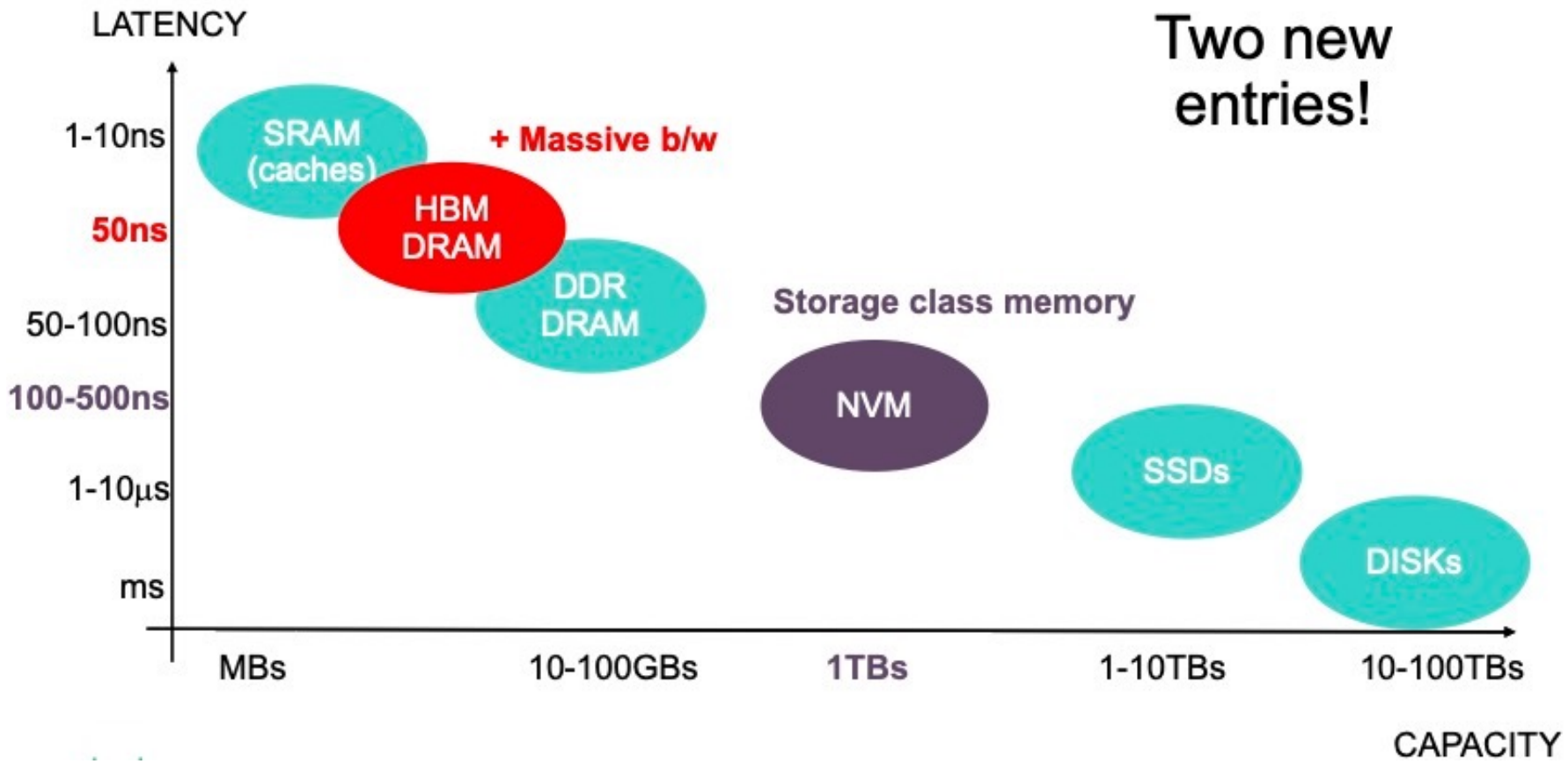


Storage Devices

- 10TB disks are about US$600
  - *Approximately 4,000 DVD movies,1.8M digital photos or 2.5M mp3 music files*
- 10TB single drives are now available
- Nimbus ExaDrive DC100: 100TB SSD
- Flash is now mainstream
  - 1TB flash is common
  - Different storage hierarchy
- Money goes a long way



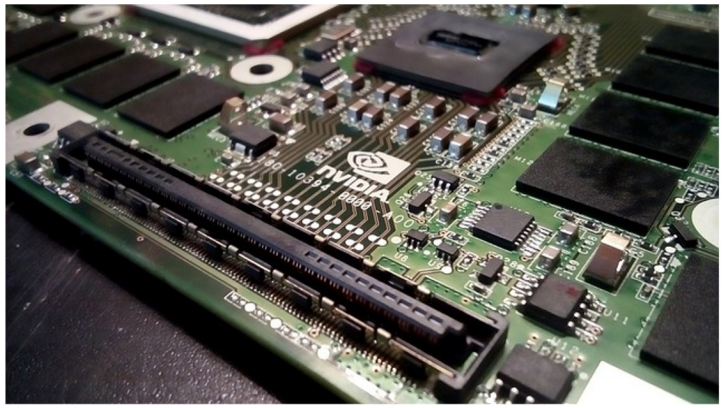
Changing Memory+Storage Hierarchy

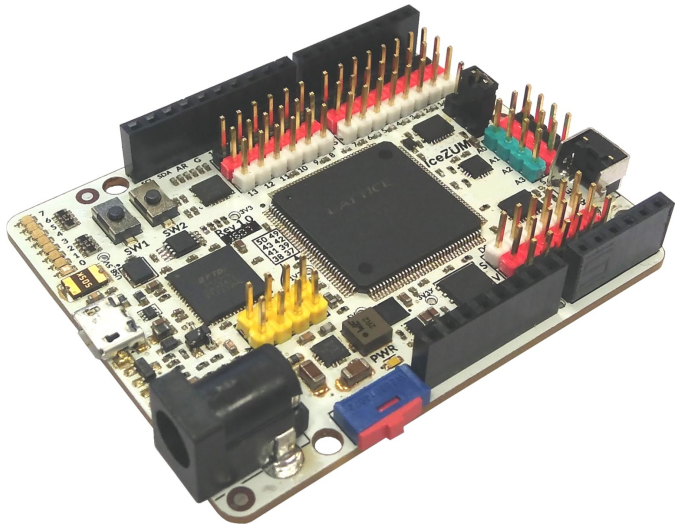


K. Keeton. Memory-Driven Computing, *Proc. USENIX FAST*, 2017.

Accellerators

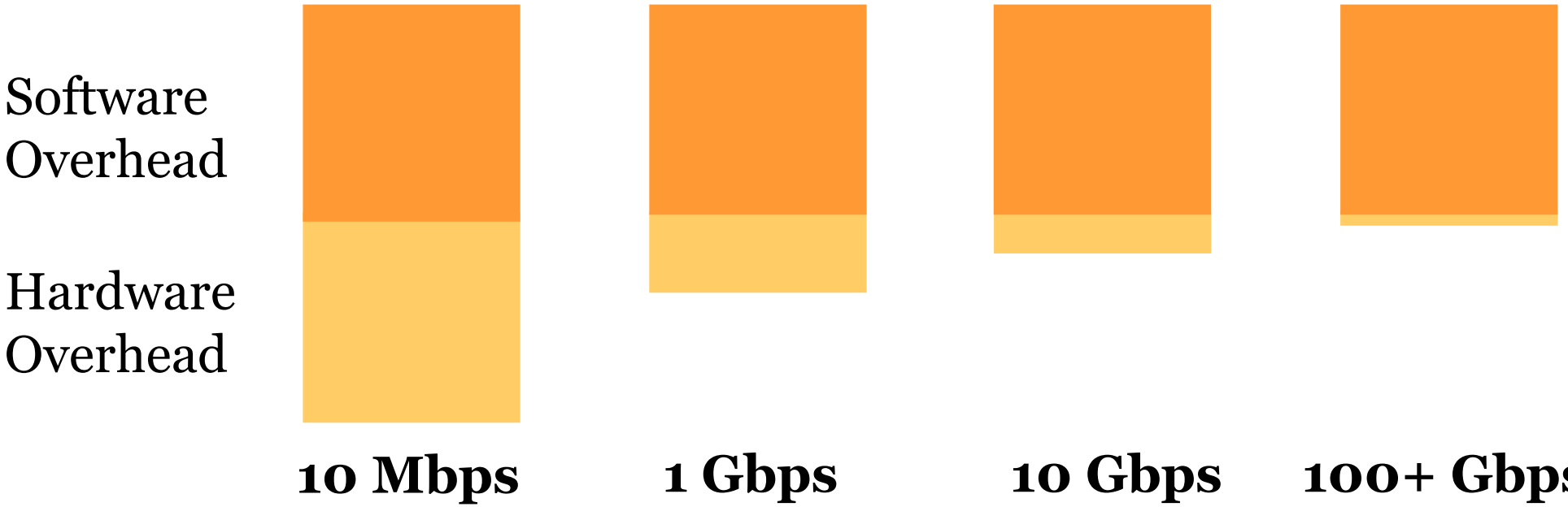
- GPU
  - Large thread parallelism
  - SIMD computation
  - Limited on-chip memory
  - Large global memory w/ increasing bandwidth
- FPGA
  - A set of programmable logic blocks
  - Logic blocks can be configured to perform complex functions
  - On-chip memory configurable
- ASIC
  - Once you know what you are doing





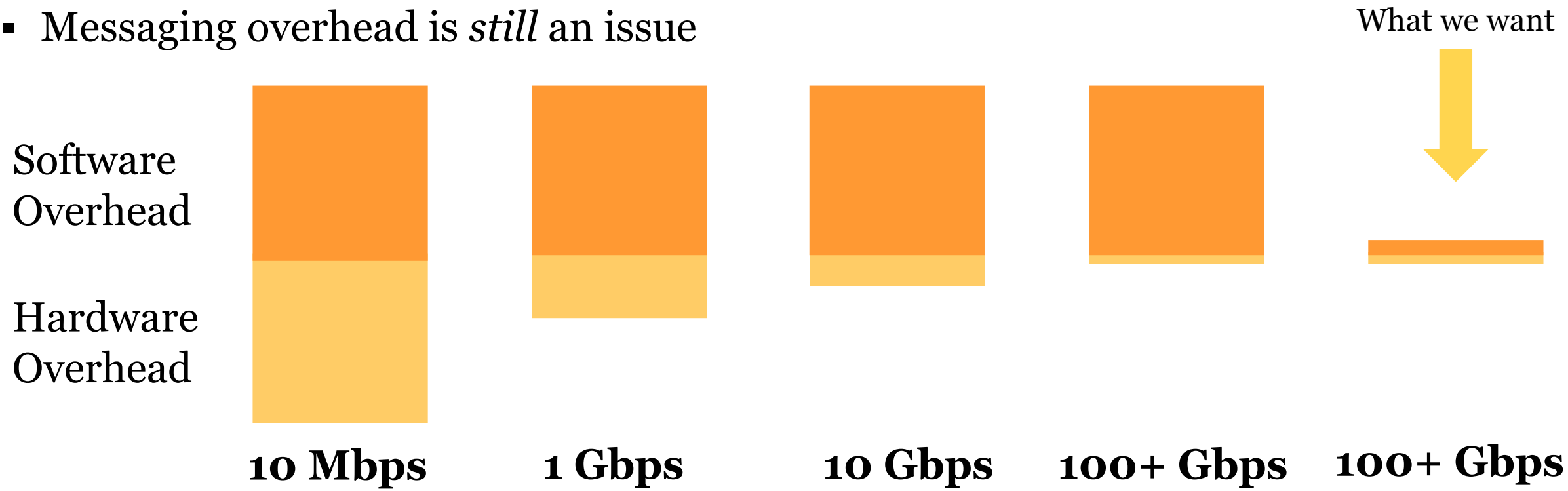
Networking Capabilities

- Bandwidth is increasing considerably
- Transmission speeds have improved (optical networking, etc)
- Messaging overhead is *still* an issue



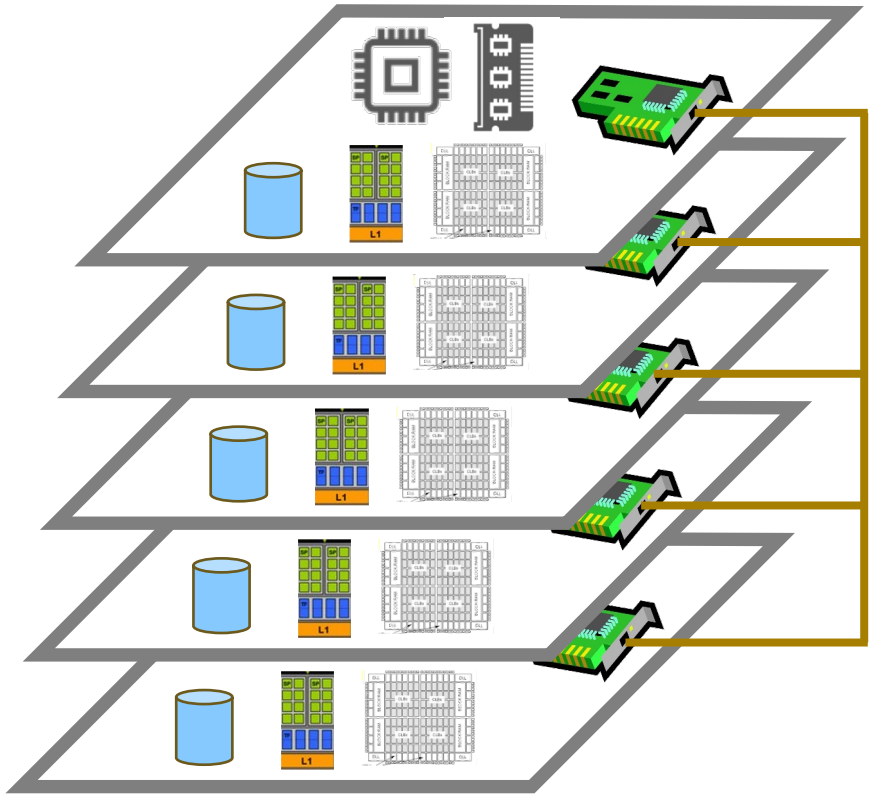
Networking Capabilities

- Bandwidth is increasing considerably
- Transmission speeds have improved (optical networking, etc)
- Messaging overhead is *still* an issue

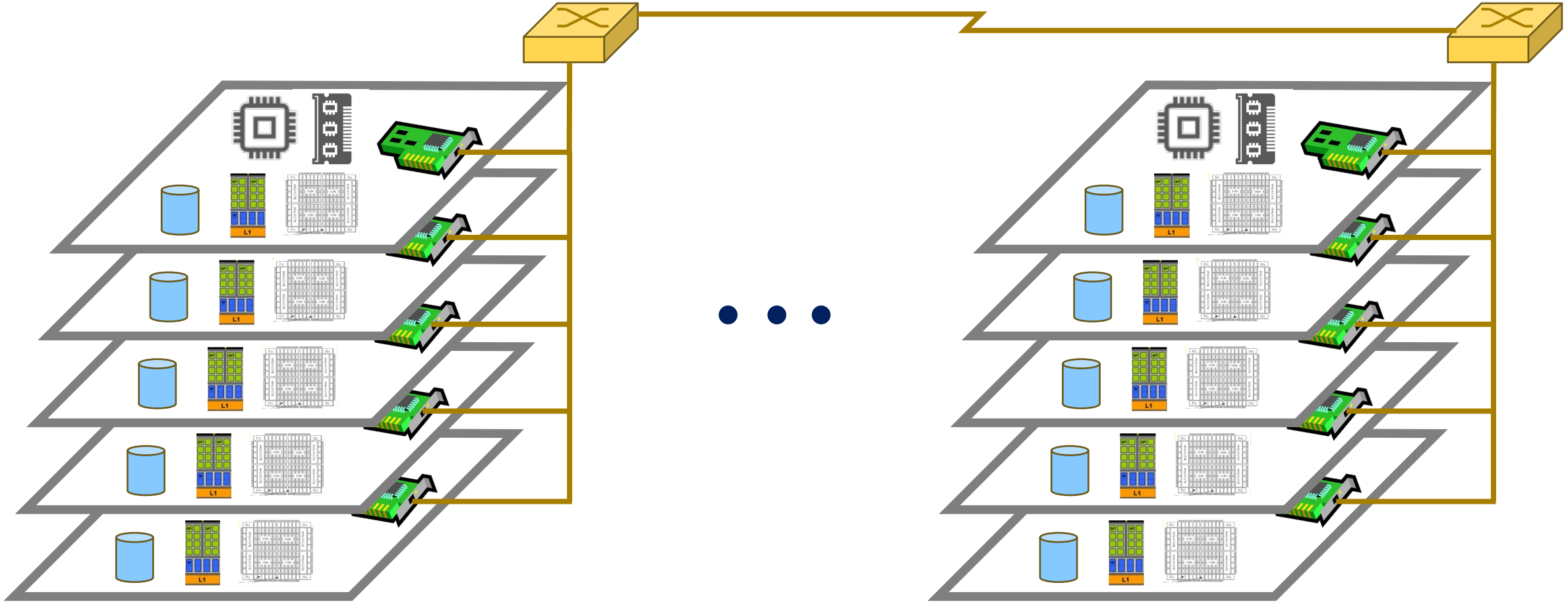


CURRENT ENVIRONMENT

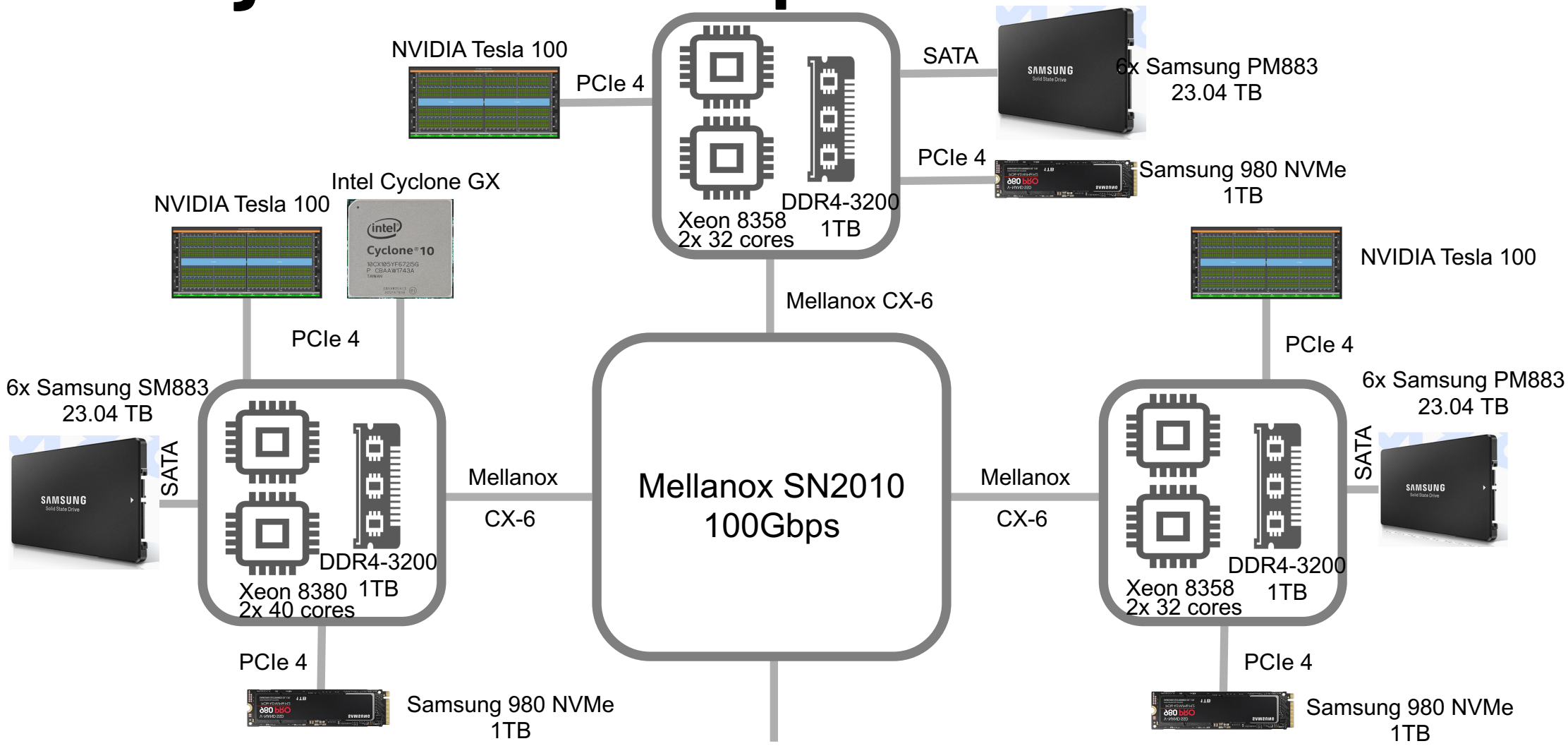
Current Data Centre Rack and Blade Design



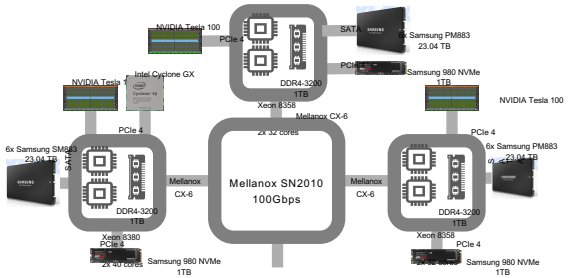
Current Data Centre Rack and Blade Design

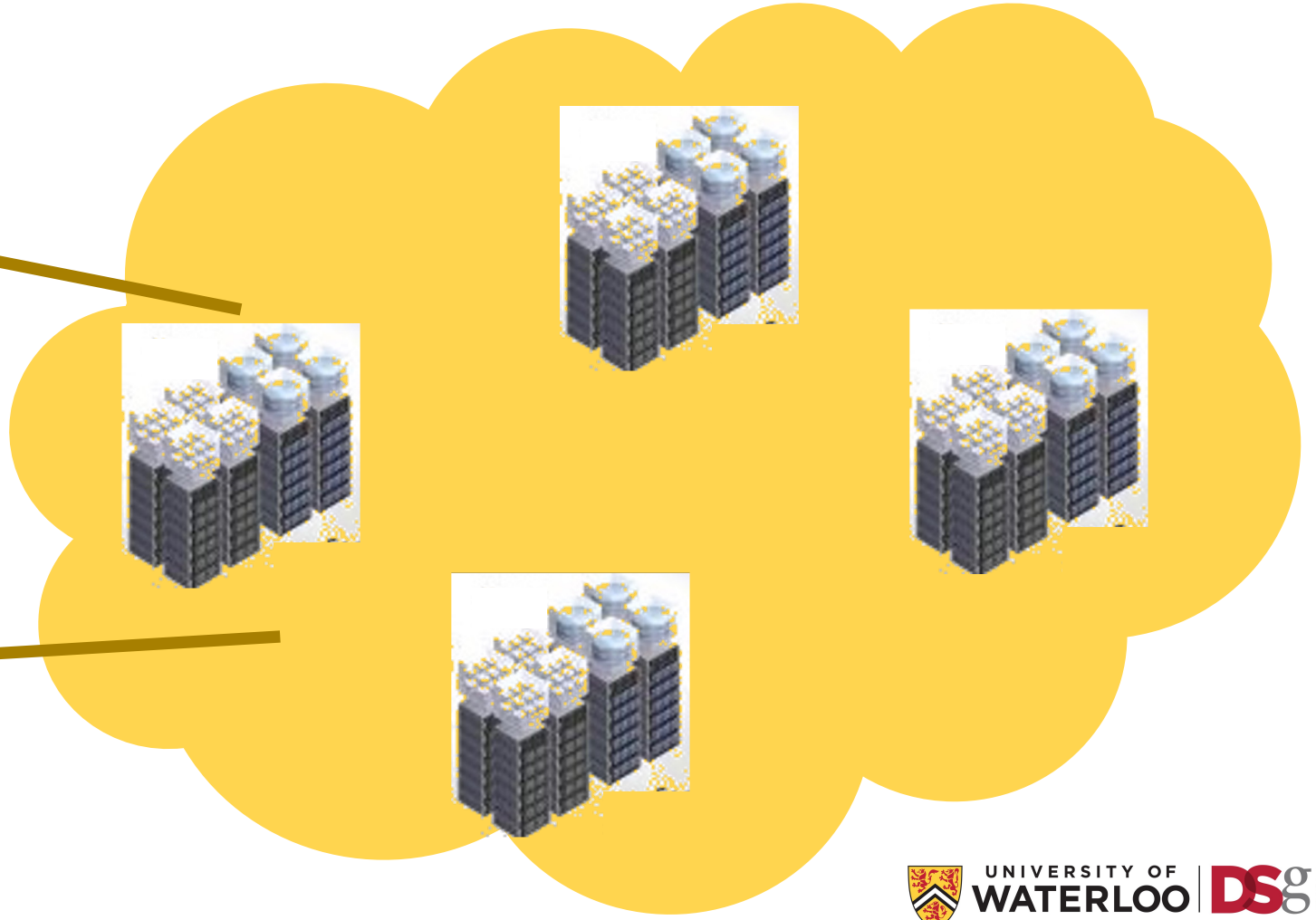


Our Existing Platform as Example



Our Existing Platform as Example





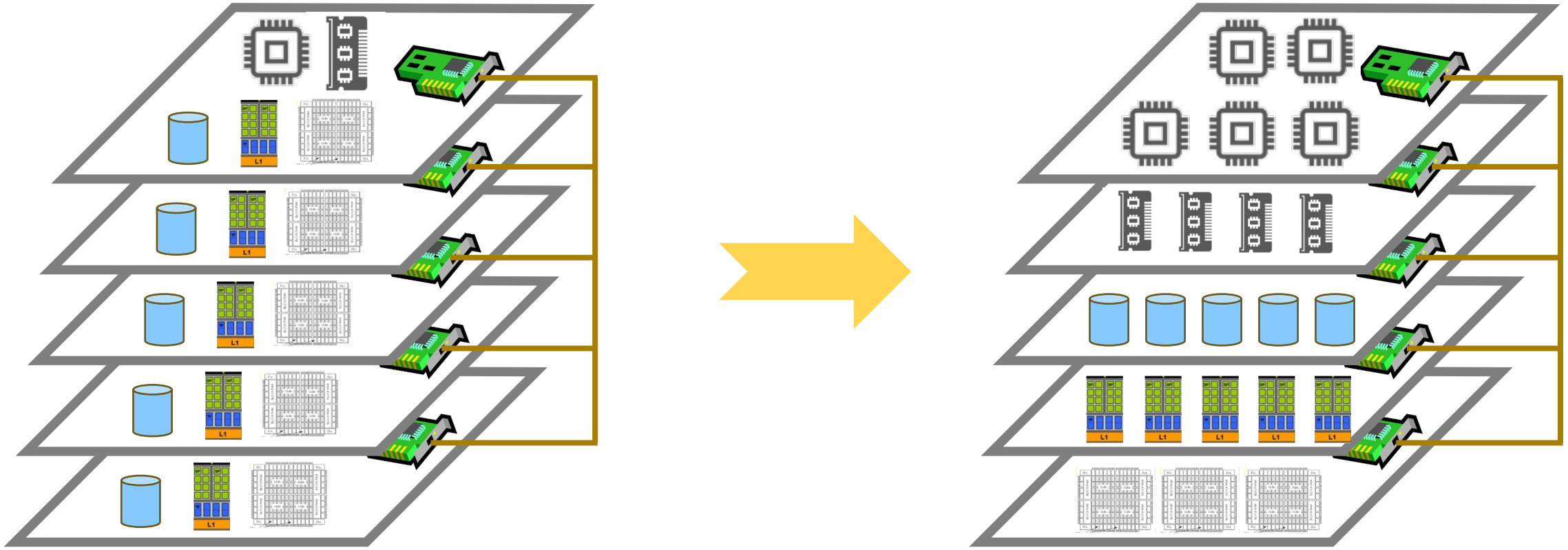
UNIVERSITY OF WATERLOO | DSG Data Systems Group

DISAGGREGATED HETEROGENEOUS PLATFORM

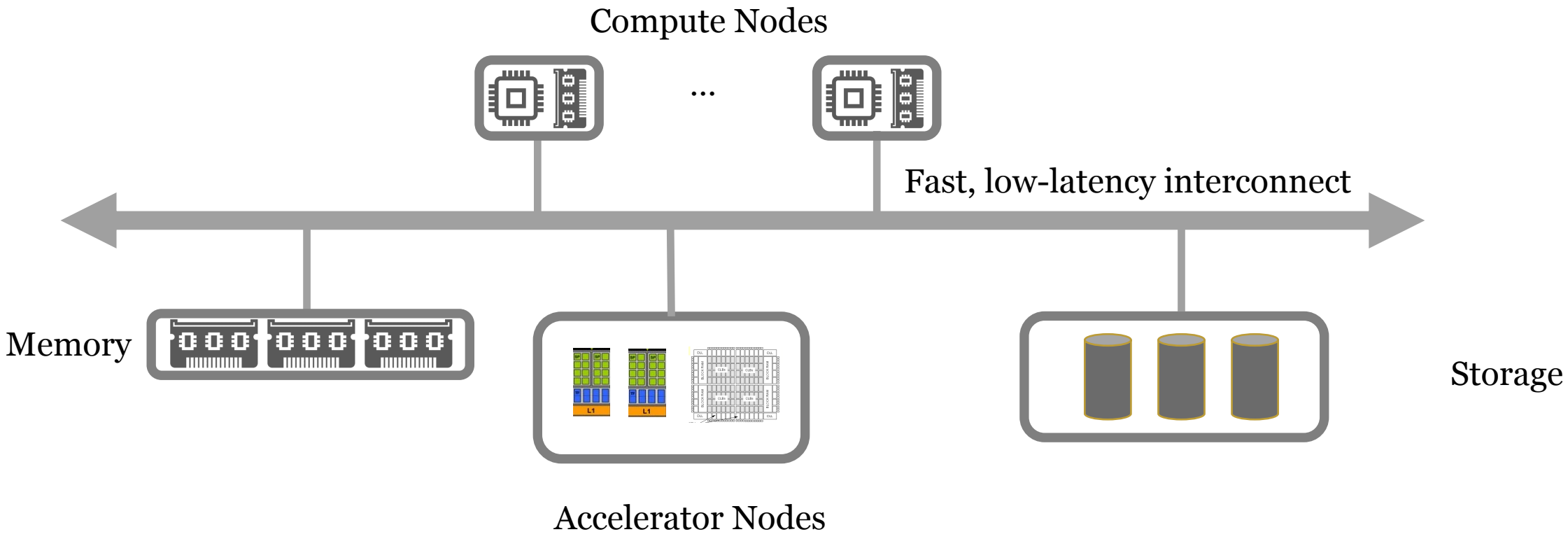
DISAGGREGATED HETEROGENEOUS PLATFORM

Disaggregation

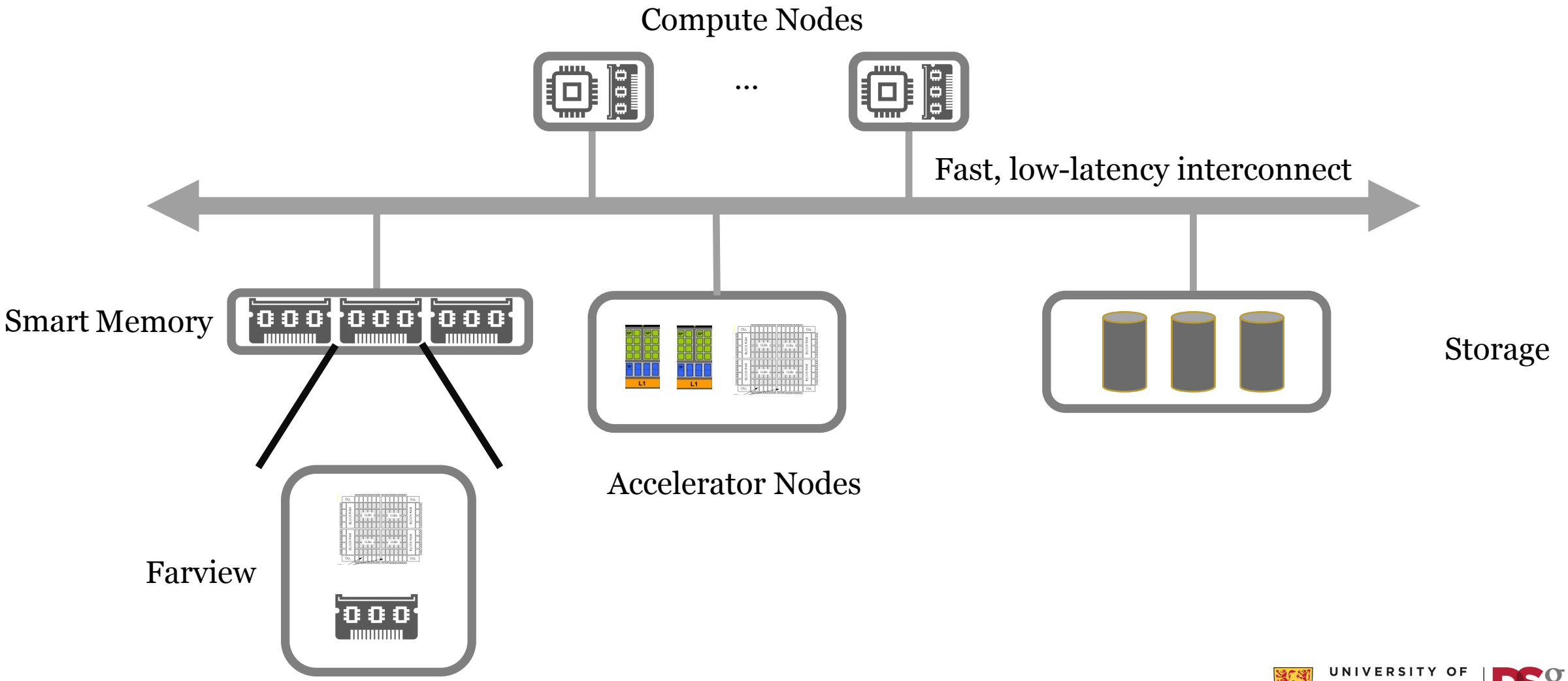
Disaggregated Rack and Blade Design



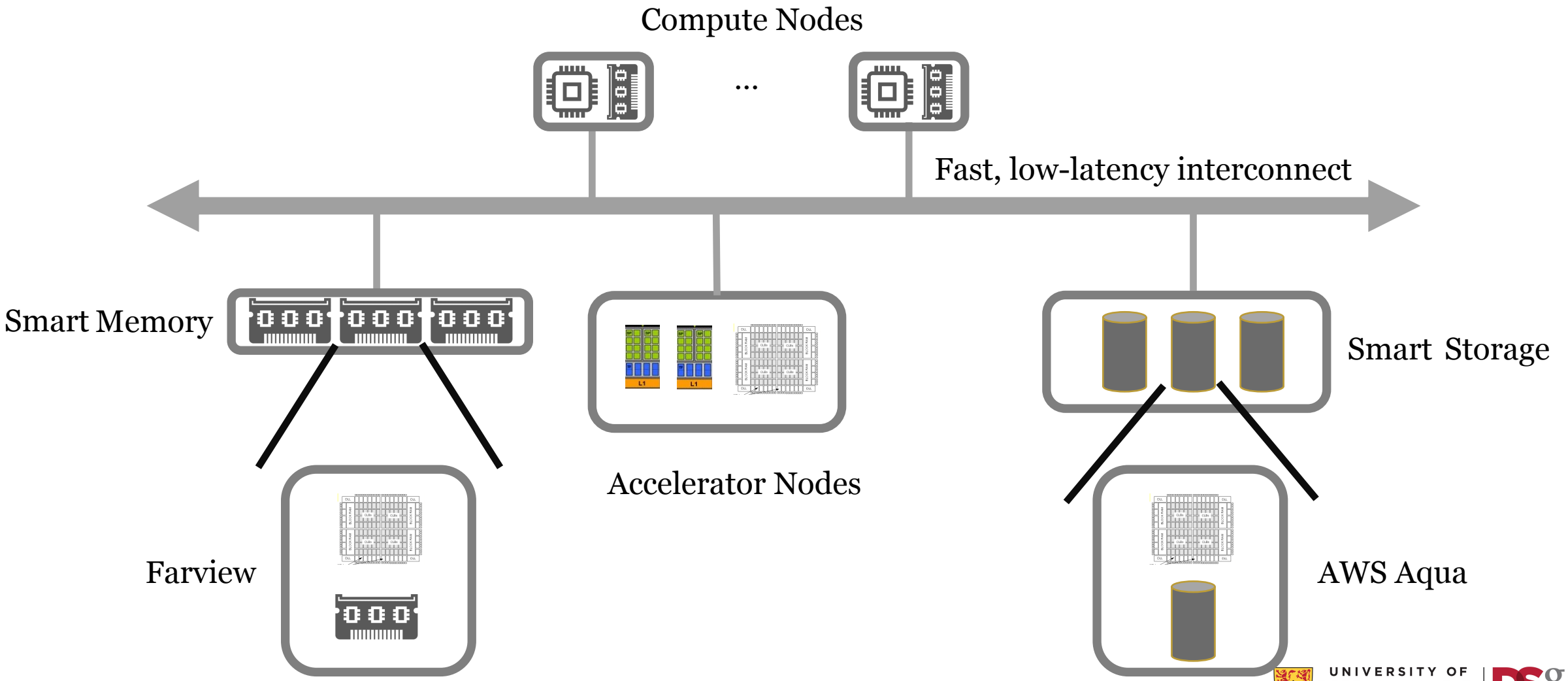
Disaggregated Heterogeneous Platform



Disaggregated Heterogeneous Platform



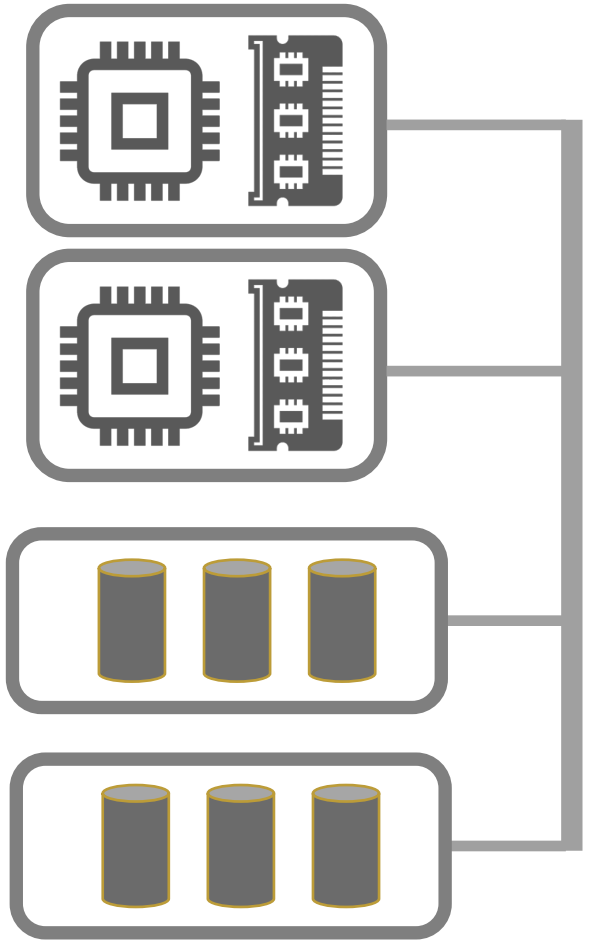
Disaggregated Heterogeneous Platform



What Are We Disaggregating?

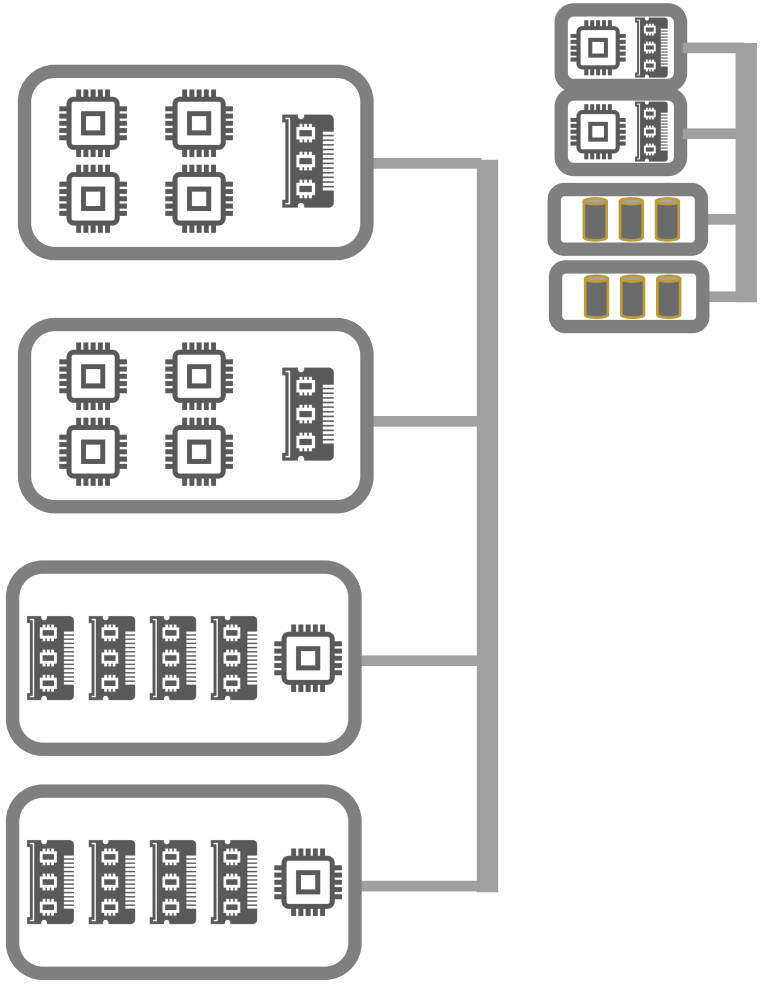
What Are We Disaggregating?

- Storage



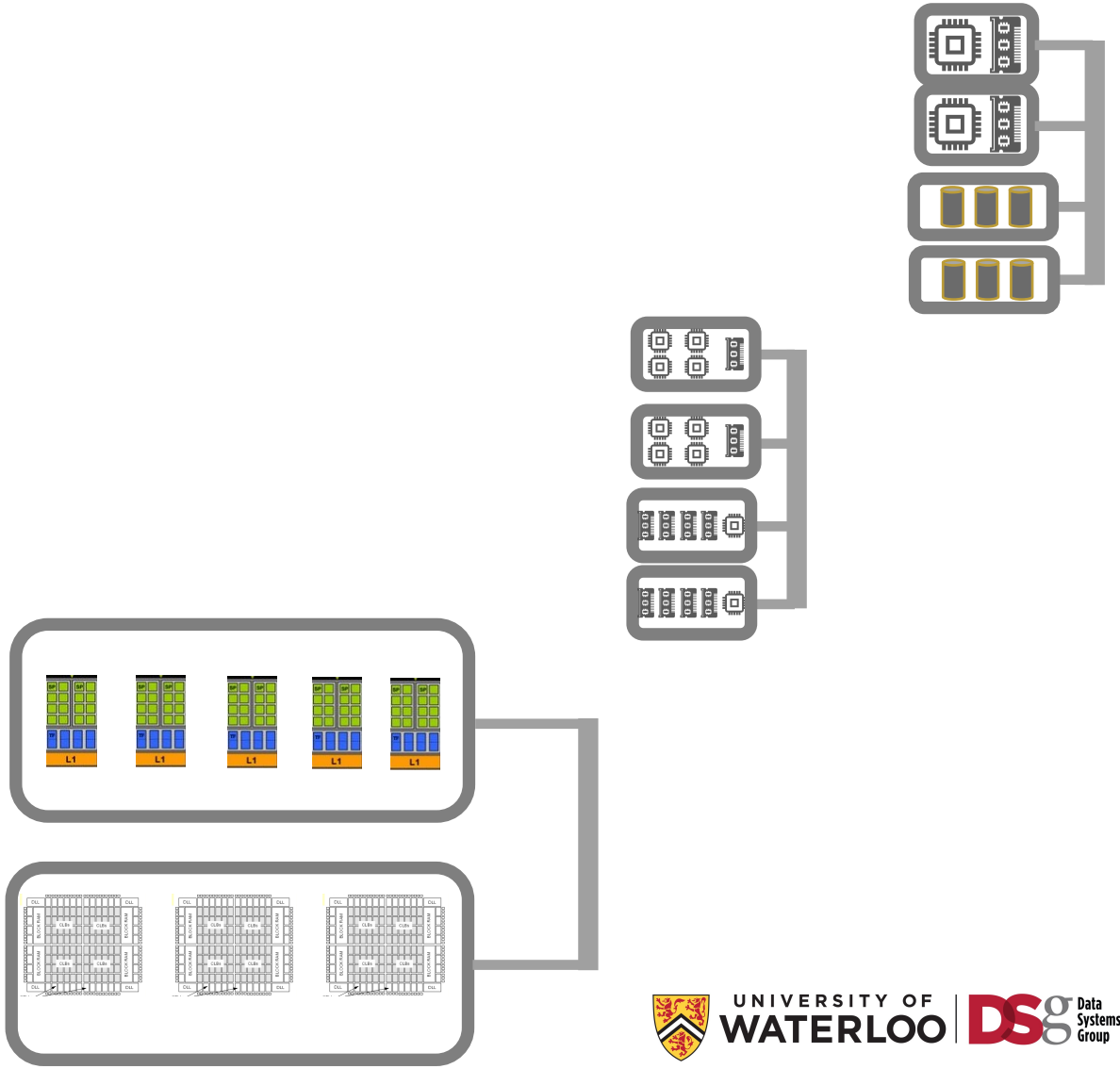
What Are We Disaggregating?

- Storage
- Memory



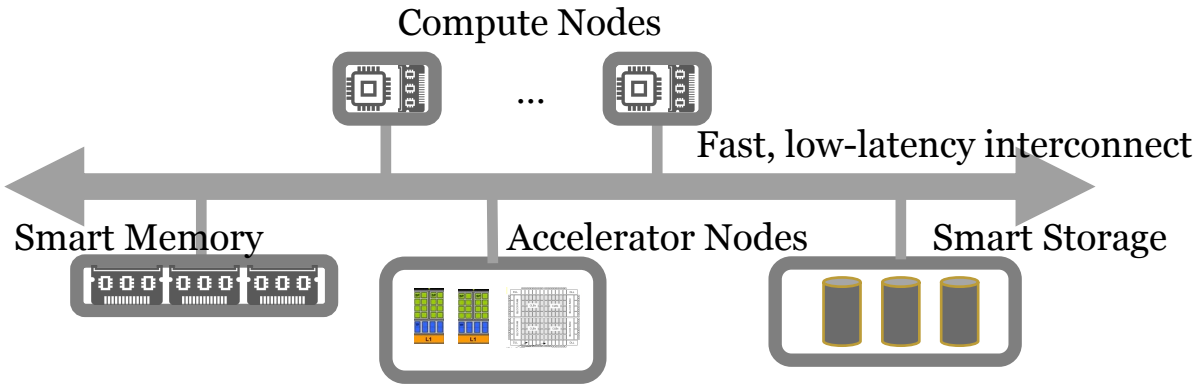
What Are We Disaggregating?

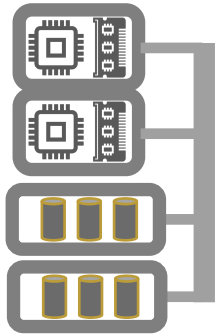
- Storage
- Memory
- Accelerators

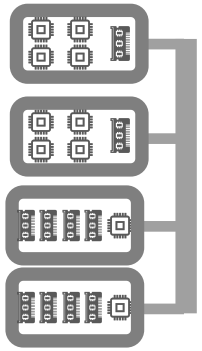


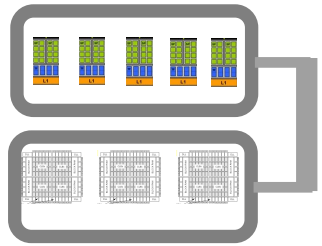
What Are We Disaggregating?

- Storage
- Memory
- Accelerators



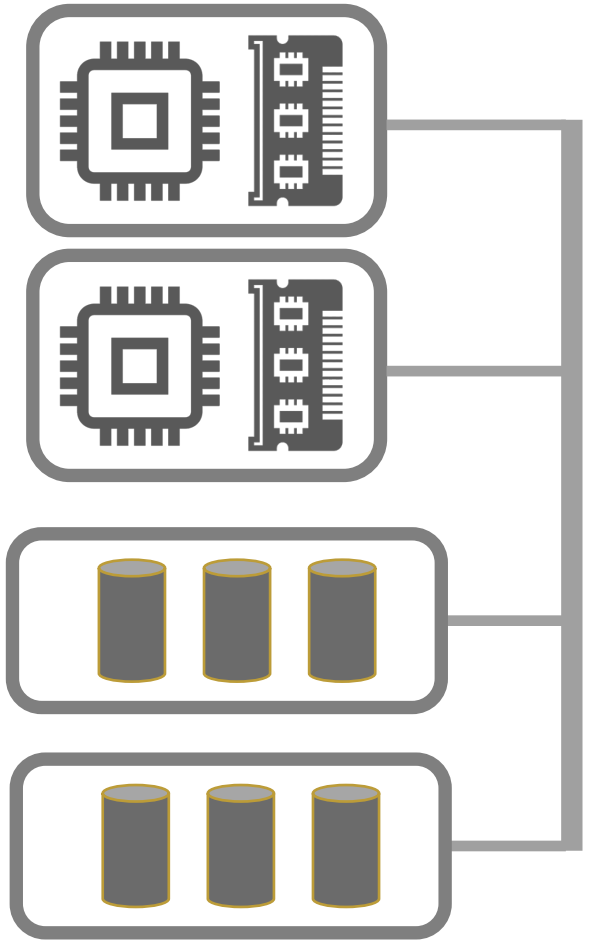






Storage Disaggregation

- Cloud deployments prefer shared disk[1]
  - Easier for crash recovery; no need for migration
- Easier for serverless computing
  - Faster startup, easier shutdown
- Independent and easy scaling of storage
- TCP/IP for connection may be sufficient



[1] J. Tan et al. Choosing a Cloud DBMS: Architectures and Tradeoffs, *Proc. VLDB*, 2019.

Storage Disaggregation is Well Understood

OLTP Systems

- Amazon Aurora[1]
- Microsoft Socrates[2]
- Google AlloyDB[3]
- Alibaba PolarDB[4]

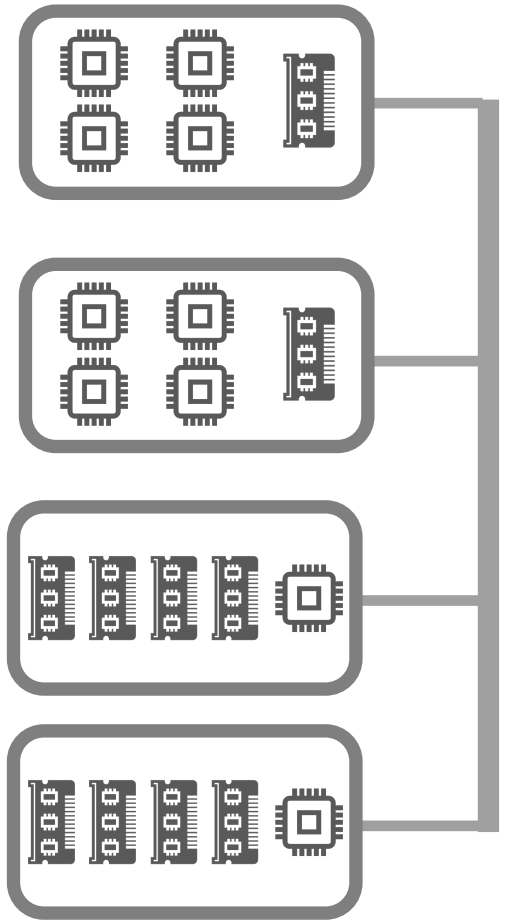
OLAP Systems

- Snowflake[5]
- Amazon Redshift[6]

[1] A. Verbitsky et al. Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases, *Proc. SIGMOD*, 2017.
[2] P. Antonopoulos et al. Socrates: The New SQL Server in the Cloud, *Proc. SIGMOD*, 2019.
[3] https://cloud.google.com/blog/products/databases/alloydb-for-postgresql-intelligent-scalable-storage
[4] F. Li. Cloud-Native Database Systems at Alibaba: Opportunities and Challenges, *Proc. VLDB Endow.*, 2019.
[5] B. Dageville et al. The Snowflake Elastic Data Warehouse, *Proc. SIGMOD*, 2016.
[6] N. Armenatzoglou et al. Amazon Redshift Re-invented, *Proc. SIGMOD*, 2022.

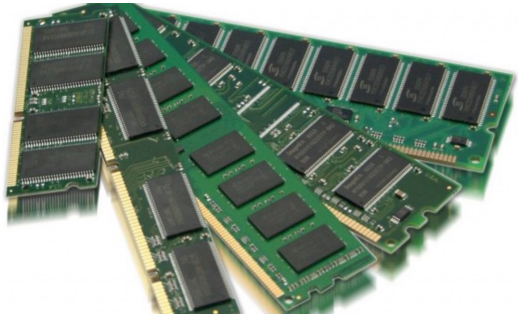
Memory Disaggregation

- Newer direction



Memory Disaggregation

- Newer direction
- The case for memory disaggregation
  - DRAM is an expensive resource in the cloud – 50% of server cost on Azure[1]

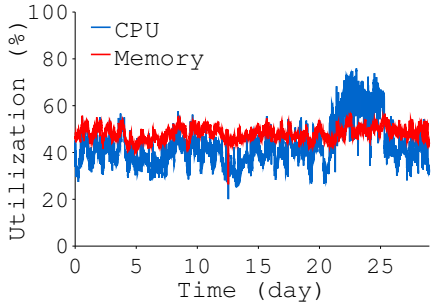


Azure

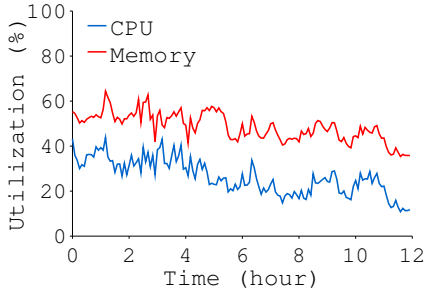
[1] H. Li et al. Pond: CXL-Based Memory Pooling Systems for Cloud Platforms, arXiv 2203.00241, 2022.

Memory Disaggregation

- Newer direction
- The case for memory disaggregation
  - DRAM is an expensive resource in the cloud – 50% of server cost on Azure[1]
  - Memory utilization is low in the (current) cloud[2]



Google Cluster



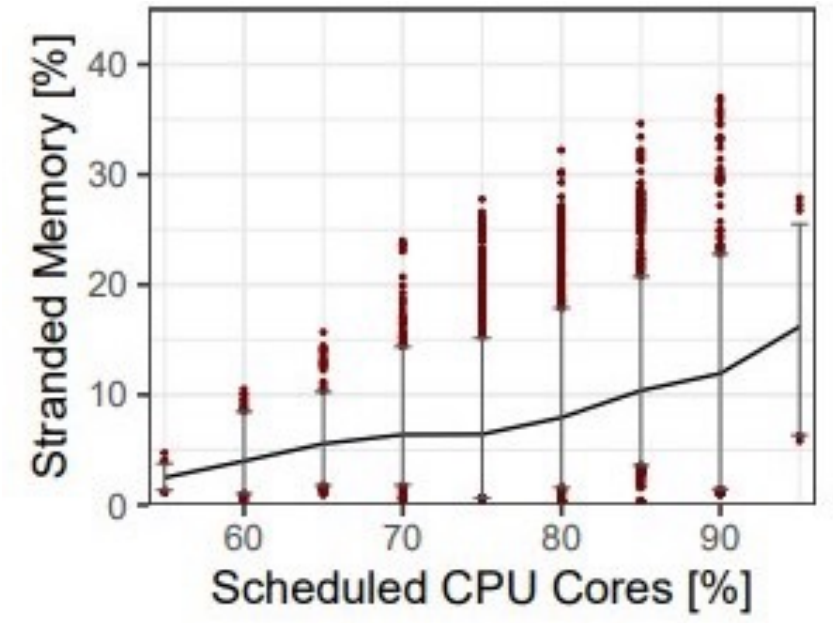
Alibaba Cluster

[1] H. Li et al. Pond: CXL-Based Memory Pooling Systems for Cloud Platforms, arXiv 2203.00241, 2022.

[2] Y. Shen et al. Pond: LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation, *Proc. OSDI*, 2018.

Memory Disaggregation

- Newer direction
- The case for memory disaggregation
  - DRAM is an expensive resource in the cloud – 50% of server cost on Azure[1]
  - Memory utilization is low in the (current) cloud[2]
  - Stranded memory (shows Azure[3])



[1] H. Li et al. Pond: CXL-Based Memory Pooling Systems for Cloud Platforms, arXiv 2203.00241, 2022.
[2] Y. Shen et al. Pond: LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation, *Proc. OSDI*, 2018.
[3] https://www.nextplatform.com/2022/07/11/microsoft-azure-blazes-the-disaggregated-memory-trail-with-znuma/

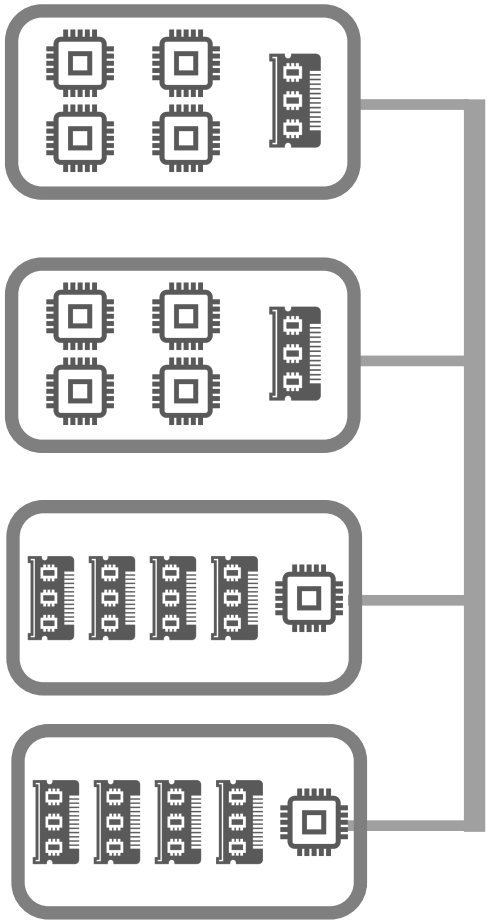
Memory Disaggregation

- Newer direction
- The case for memory disaggregation
  - DRAM is an expensive resource in the cloud – 50% of server cost on Azure[1]
  - Memory utilization is low in the (current) cloud[2]
  - Stranded memory (shows Azure[3])
  - Independent and elastic scaling of compute and memory[4]

AWS EC2 Instance	vCPU	DRAM (GB)
r6g.medium	1	8
r6g.large	2	16
r6g.xlarge	4	32
r6g.2xlarge	8	64
r6g.4xlarge	16	128
r6g.8xlarge	32	256
r6g.16xlarge	64	512

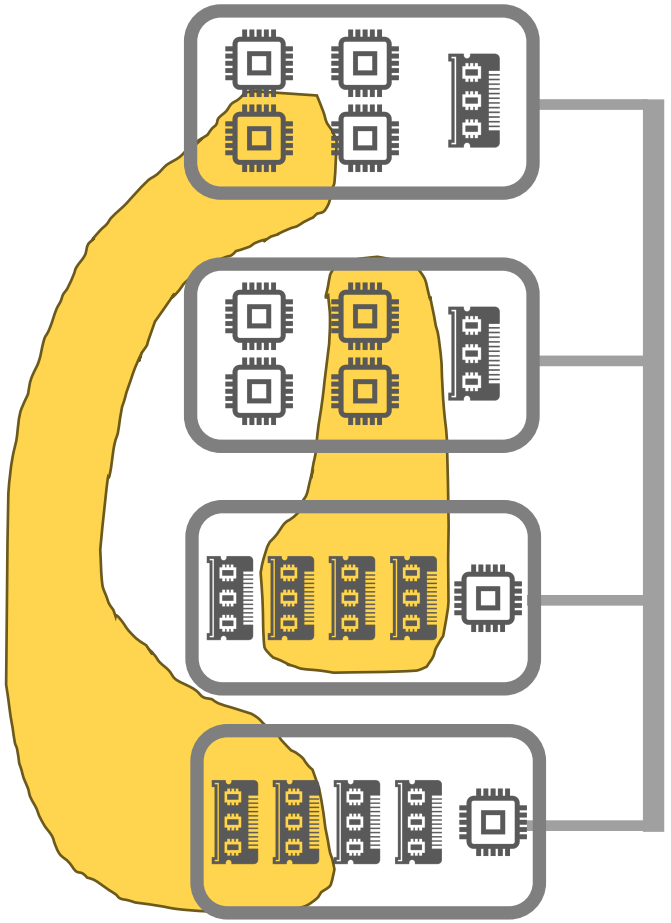
[1] H. Li et al. Pond: CXL-Based Memory Pooling Systems for Cloud Platforms, arXiv 2203.00241, 2022.
[2] Y. Shen et al. Pond: LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation, *Proc. OSDI*, 2018.
[3] https://www.nextplatform.com/2022/07/11/microsoft-azure-blazes-the-disaggregated-memory-trail-with-znuma/
[4] https://aws.amazon.com/ec2/instance-types/

The Case for Memory Disaggregation



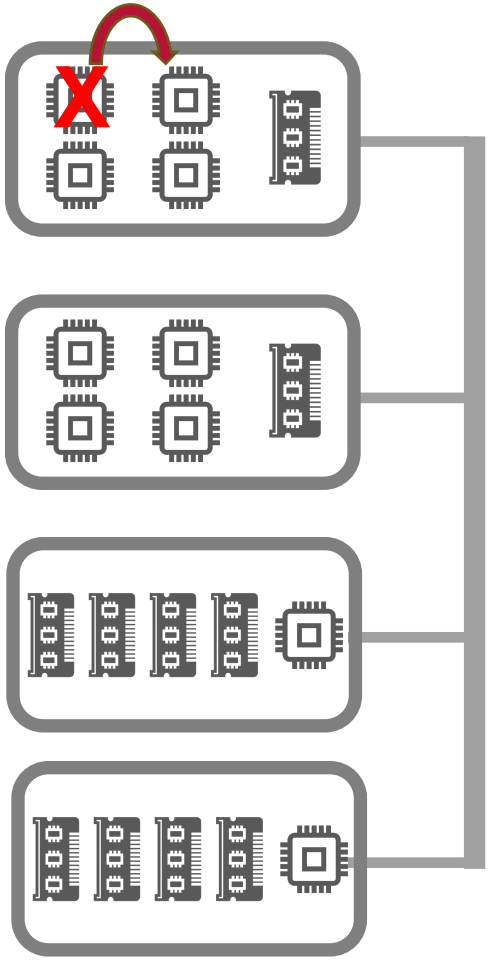
The Case for Memory Disaggregation

- Flexibility in VM configuration
  - Independent allocation



The Case for Memory Disaggregation

- Flexibility in VM configuration
  - Independent allocation
- Fault tolerance
  - Independent failure

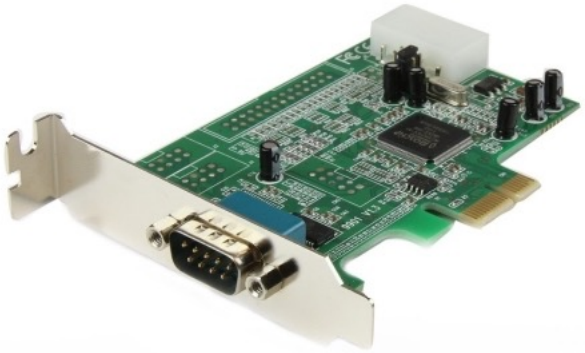


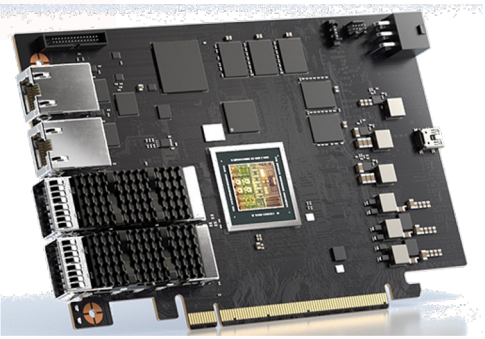
Critical Component in Disaggregation (Beyond Storage)

Low latency, high bandwidth interconnect

Networking Hardware Improvements

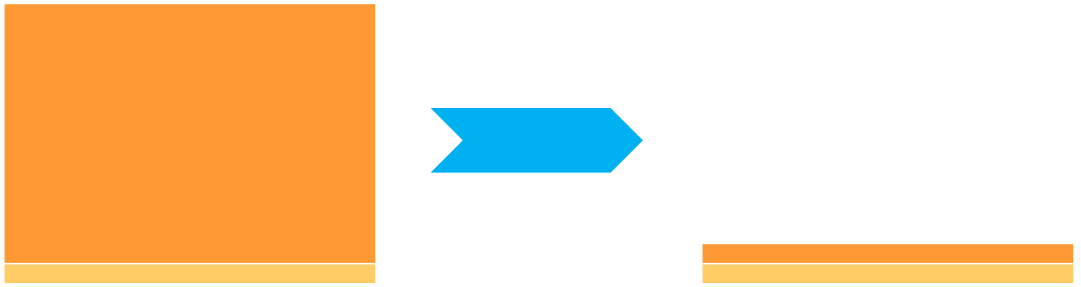
- Basic/Foundational NIC
  - Simple network connection
  - Usually 1Gbps – 25Gbps
  - Relies on CPU for protocol processing – ≥ 30% server CPU for higher speeds
- Smart NIC
  - Offload network protocol processing
  - Have their own processor, memory & OS
  - ≥ 50Gbps
- DPU
  - Smart NIC + security + storage + …
  - Custom chips and/or FPGAs





T. Döring et al., SmartNICs: Current Trends in Research and Industry, 2020
https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2021-05-1/NET-2021-05-1_05.pdf

Networking Software Improvements



100+ Gbps

Networking Software Improvements

- High overhead of TCP

100+ Gbps

Networking Software Improvements

- High overhead of TCP
  - "Data center tax"[1]

100+ Gbps

[1] L. Barroso, et al. Attack of the Killer Microseconds. *Commun. ACM*, 60(4):48–54, March 2017.

Networking Software Improvements

- High overhead of TCP
  - "Data center tax"[1]
  - "Modern networking hardware enables roundtrip times of a few microseconds for short messages. The transport protocol must not add significantly to this latency"[2]

It's Time to Replace TCP in the Datacenter

John Ousterhout
Stanford University

arXiv 2210.00714

January 18, 2023

100+ Gbps

[1] L. Barroso, et al. Attack of the Killer Microseconds. *Commun. ACM*, 60(4):48–54, March 2017.
[2] J. Ousterhout. It's Time to Replace TCP in the Datacenter, arXiv 2210.00714, 2023.

Networking Software Improvements

- High overhead of TCP
  - "Data center tax"[1]
  - "Modern networking hardware enables roundtrip times of a few microseconds for short messages. The transport protocol must not add significantly to this latency"[2]
- Low-overhead protocol
  - RDMA (Infiniband, RoCE)
  - CXL

It's Time to Replace TCP in the Datacenter

John Ousterhout
Stanford University

arXiv 2210.00714

January 18, 2023

100+ Gbps

[1] L. Barroso, et al. Attack of the Killer Microseconds. *Commun. ACM*, 60(4):48–54, March 2017.
[2] J. Ousterhout. It's Time to Replace TCP in the Datacenter, arXiv 2210.00714, 2023.

Networking Software Improvements

- High overhead of TCP
  - "Data center tax"[1]
  - "Modern networking hardware enables roundtrip times of a few microseconds for short messages. The transport protocol must not add significantly to this latency"[2]
- Low-overhead protocol
  - RDMA (Infiniband, RoCE)
  - CXL
- Efficient RPC protocol – higher level

It's Time to Replace TCP in the Datacenter

John Ousterhout
Stanford University

arXiv 2210.00714

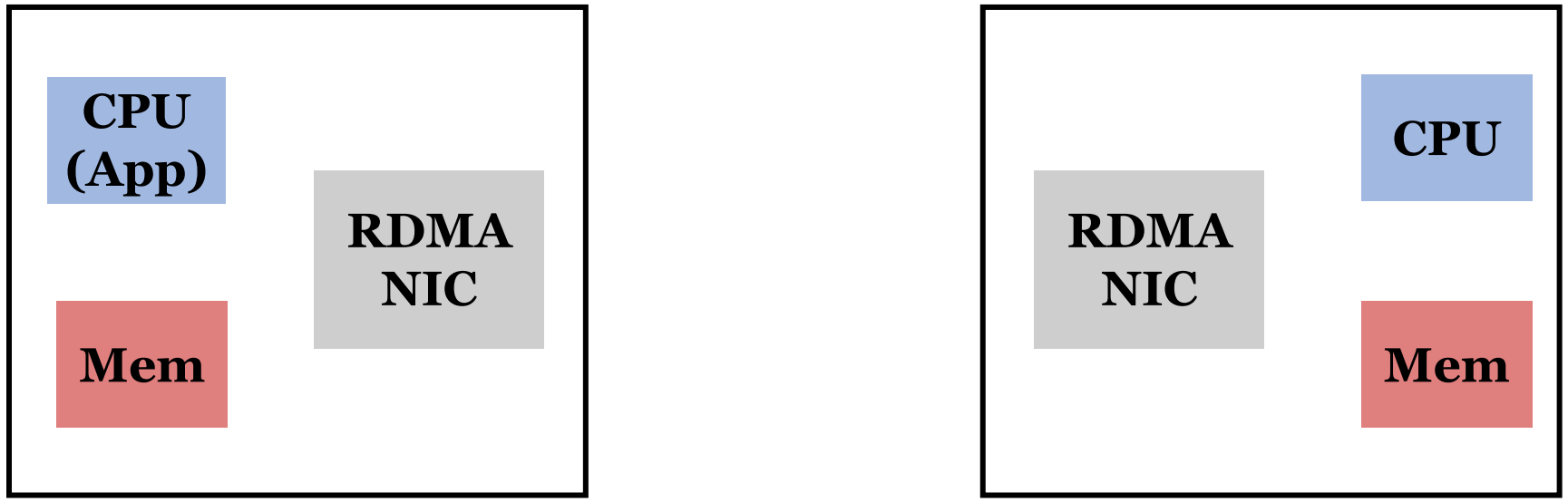
January 18, 2023

100+ Gbps

[1] L. Barroso, et al. Attack of the Killer Microseconds. *Commun. ACM*, 60(4):48–54, March 2017.
[2] J. Ousterhout. It's Time to Replace TCP in the Datacenter, arXiv 2210.00714, 2023.

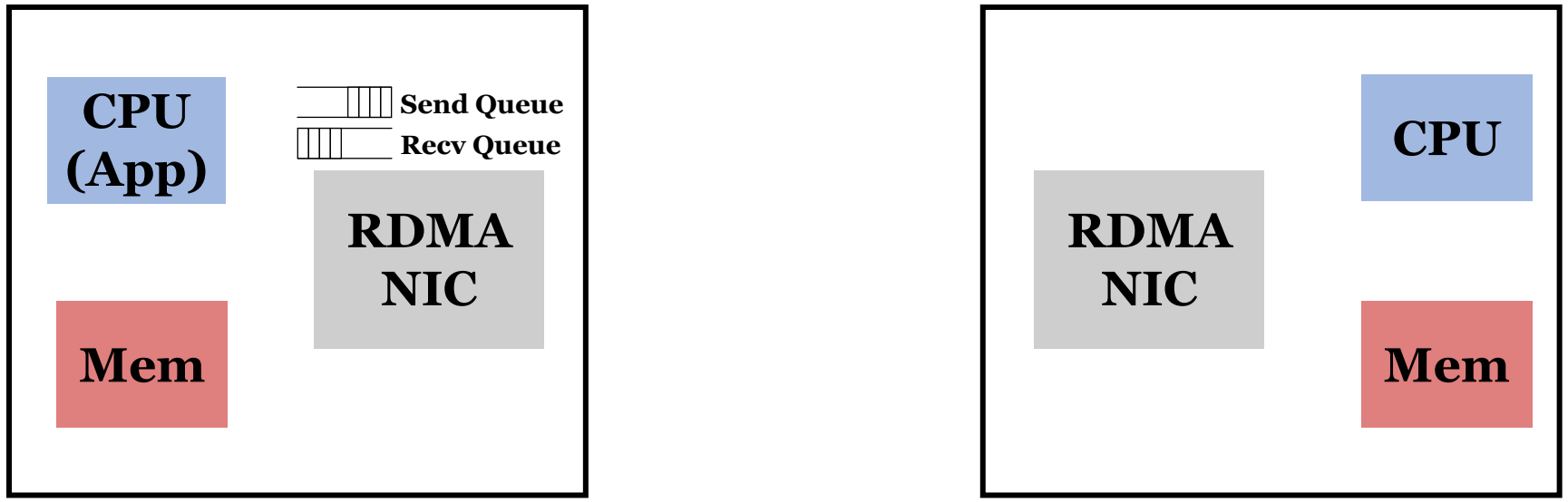
RDMA – Remote Direct Memory Access

Physical View

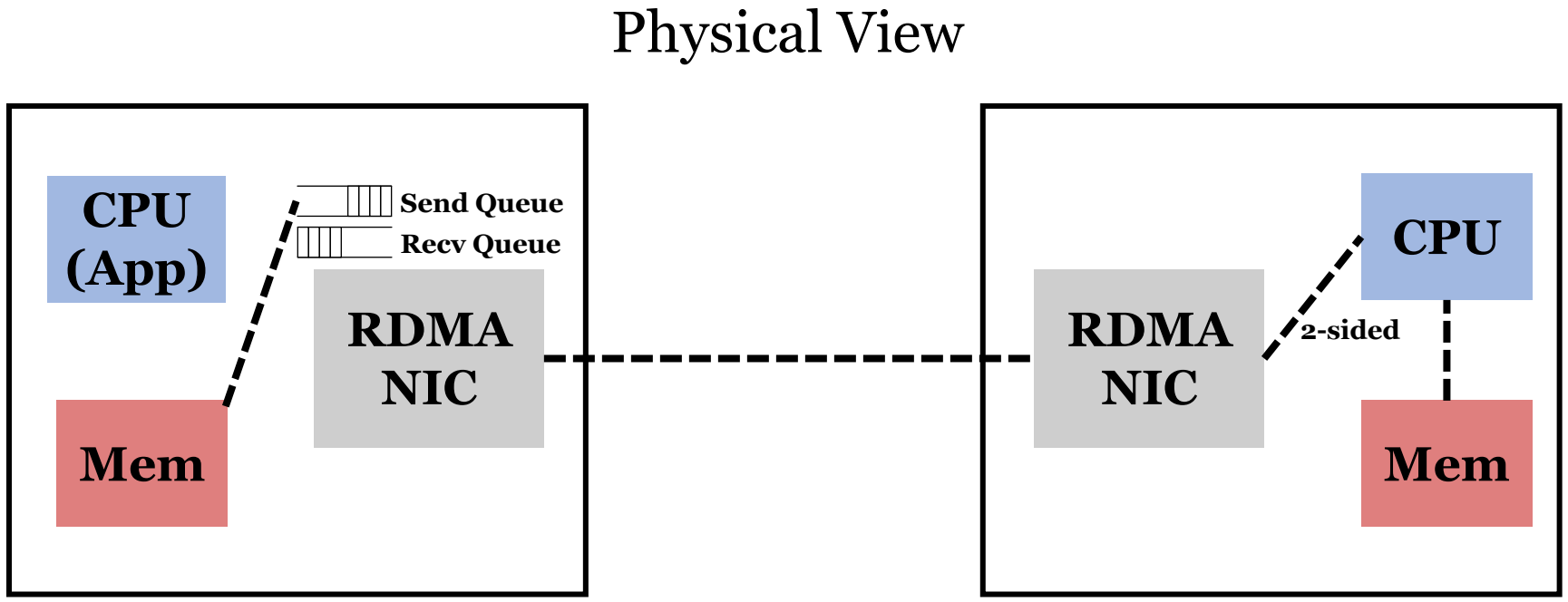


RDMA – Remote Direct Memory Access

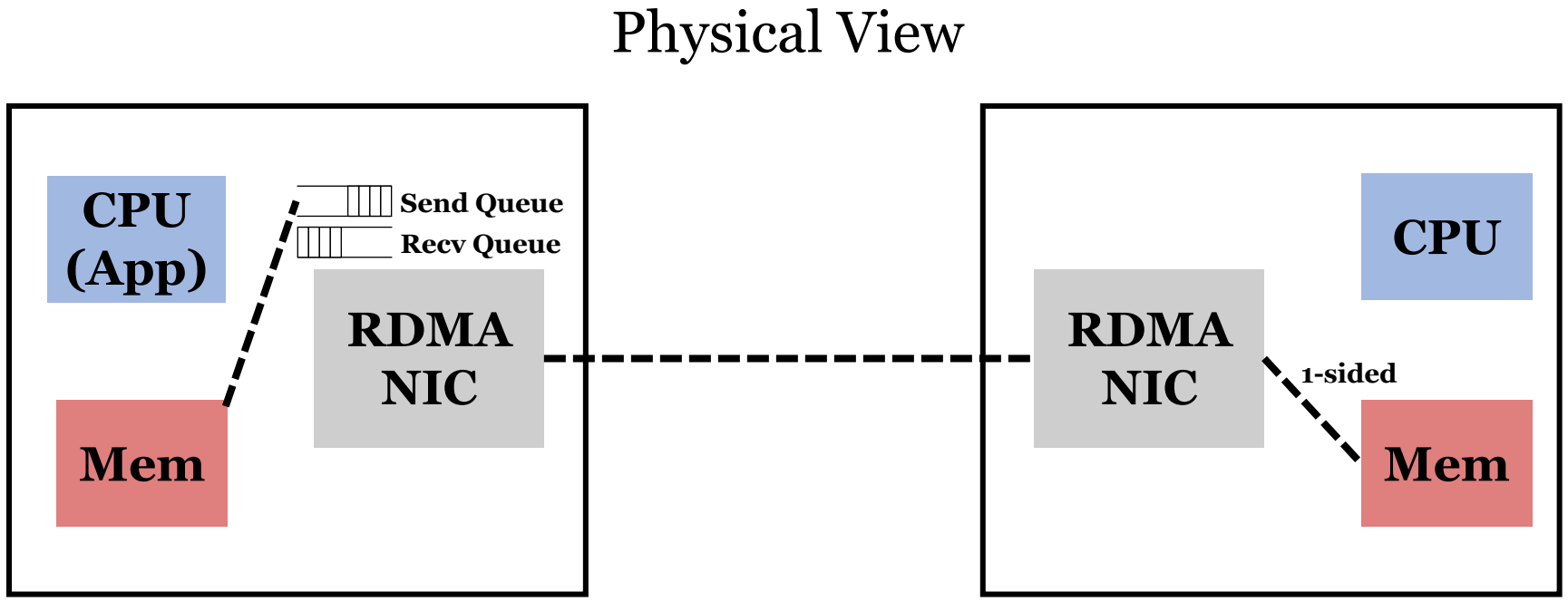
Physical View



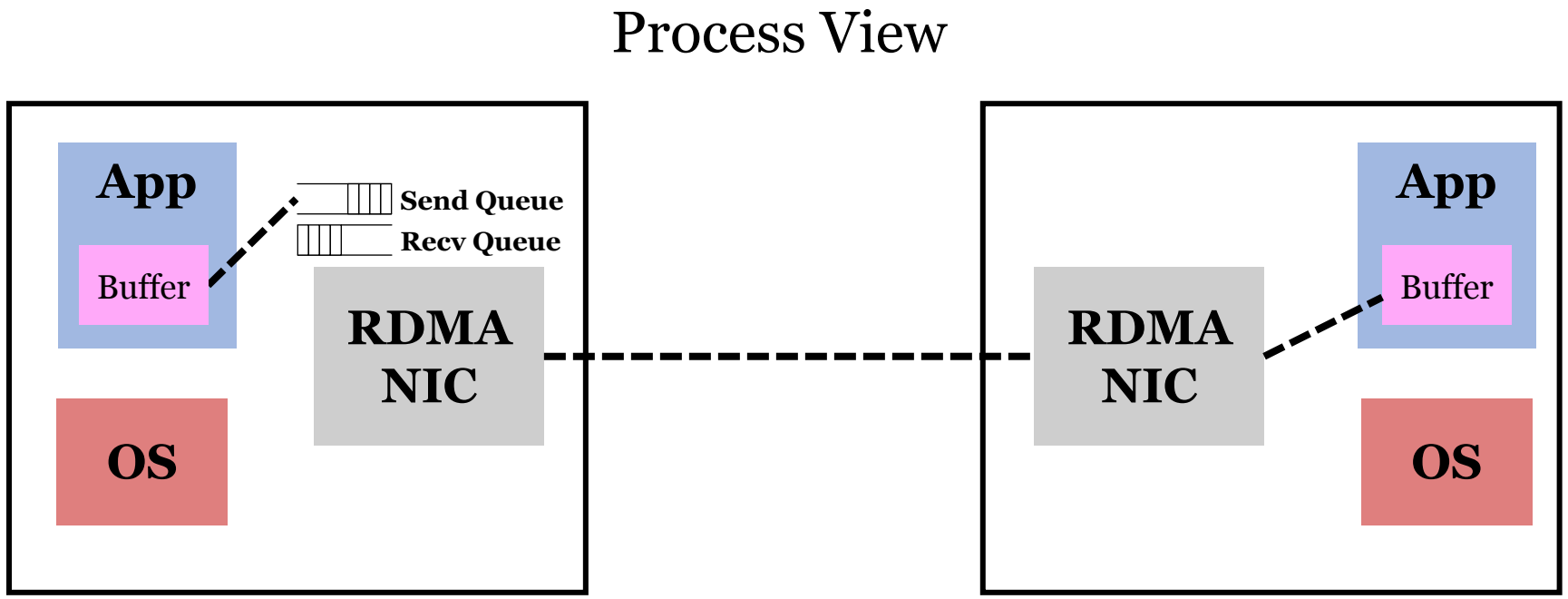
RDMA – Remote Direct Memory Access



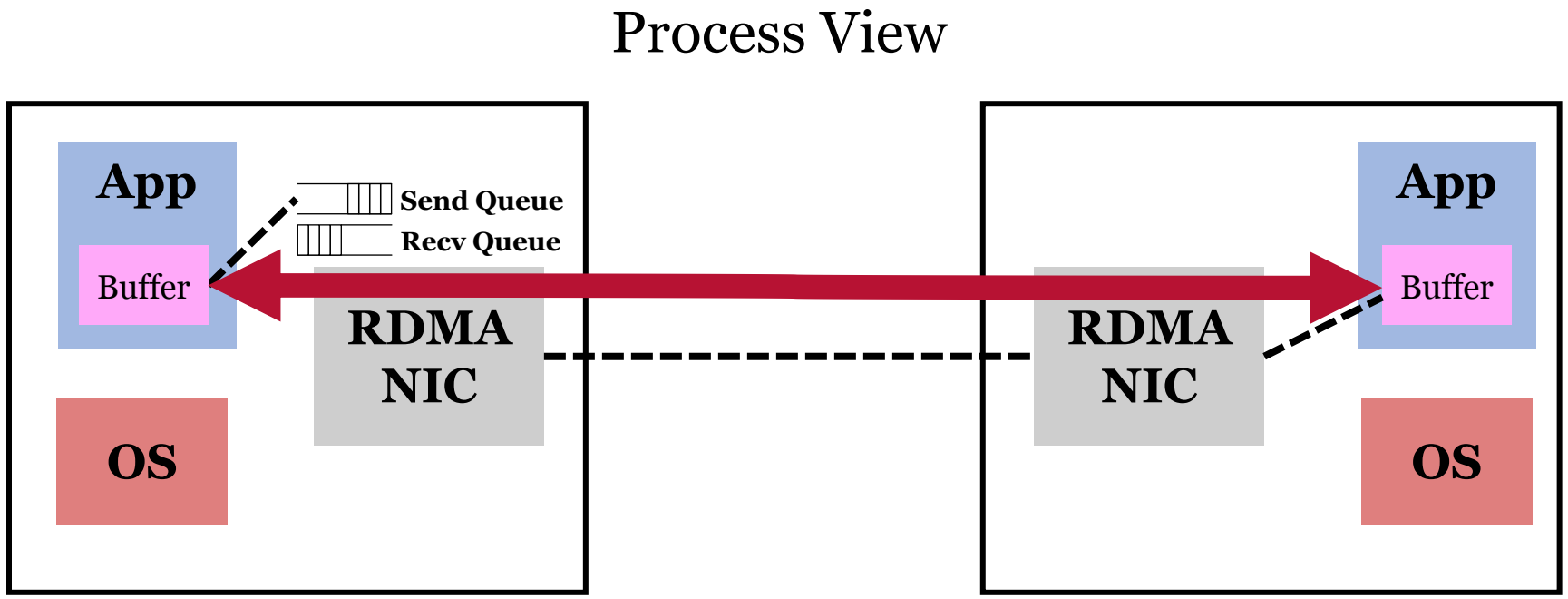
RDMA – Remote Direct Memory Access



RDMA – Remote Direct Memory Access

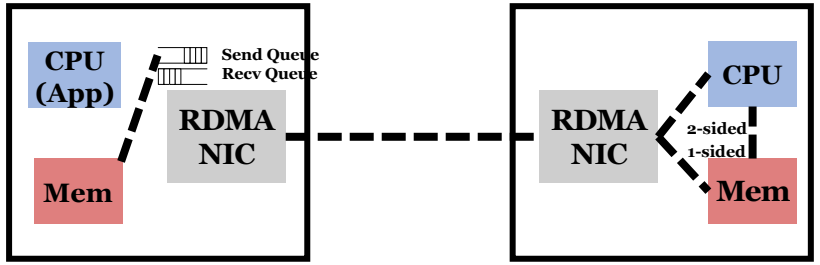


RDMA – Remote Direct Memory Access

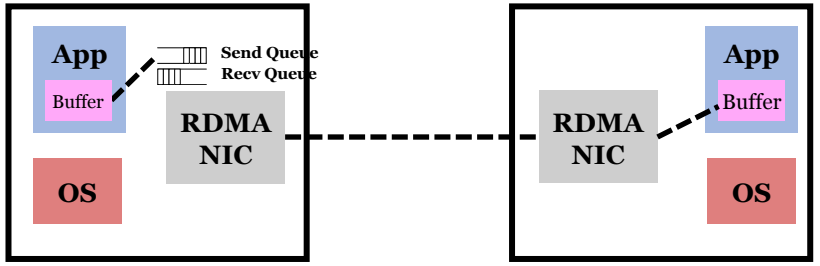


RDMA – Remote Direct Memory Access

Physical View



Process View



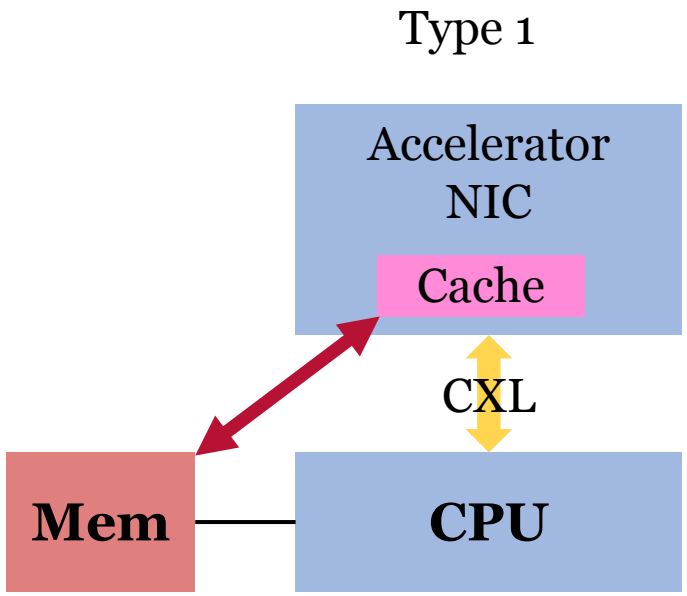
- Zero-copy networking
- OS bypass
- CPU bypass
- Message-based communication

CXL – Compute Express Link

- High bandwidth, low latency, cache coherent interconnect
- Avoids memory copy between application memory and NIC buffers
- Does not require a CPU (or controller) on memory nodes

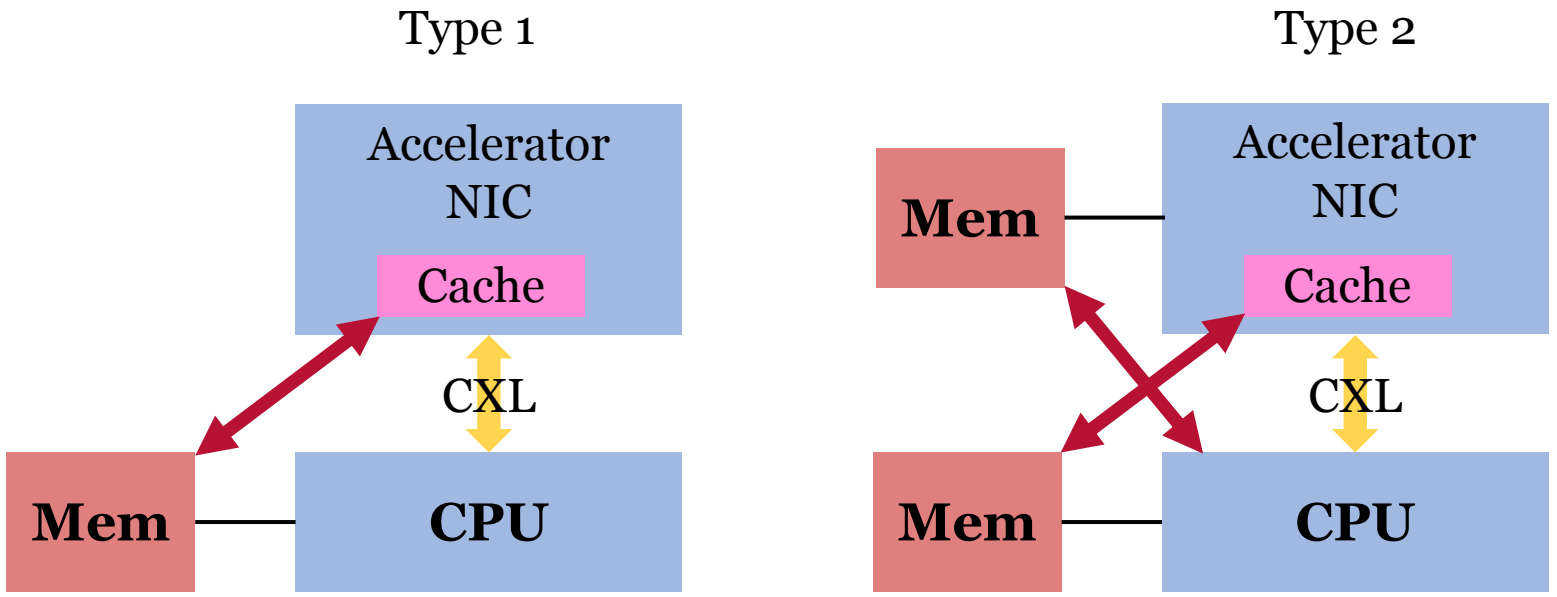
CXL – Compute Express Link

- High bandwidth, low latency, cache coherent interconnect
- Avoids memory copy between application memory and NIC buffers
- Does not require a CPU (or controller) on memory nodes



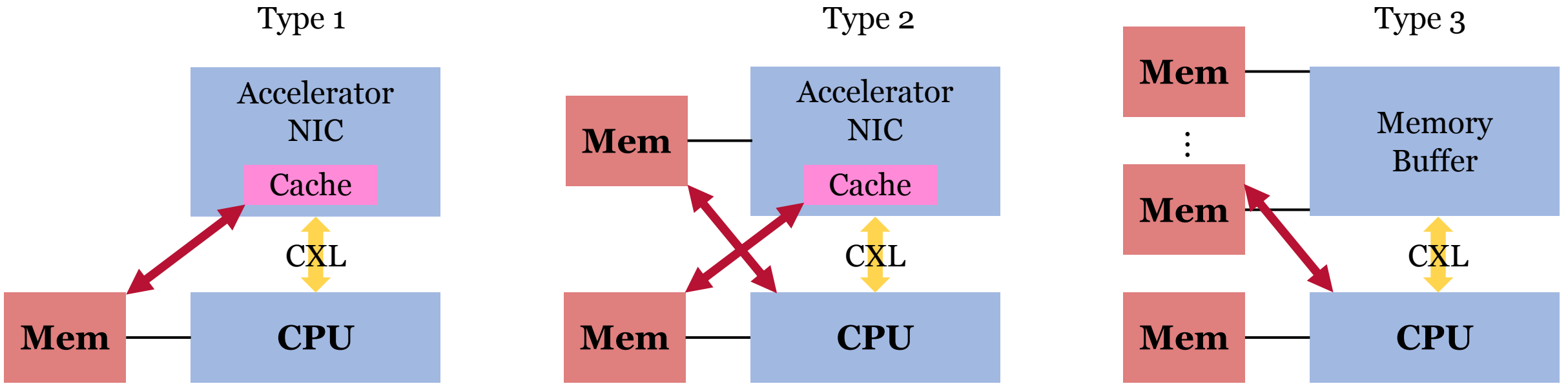
CXL – Compute Express Link

- High bandwidth, low latency, cache coherent interconnect
- Avoids memory copy between application memory and NIC buffers
- Does not require a CPU (or controller) on memory nodes



CXL – Compute Express Link

- High bandwidth, low latency, cache coherent interconnect
- Avoids memory copy between application memory and NIC buffers
- Does not require a CPU (or controller) on memory nodes



CXL vs RDMA

RDMA

- Copies data from application memory to NIC buffers
- Copies data across the network
- Requires handling of cache coherence (in some configurations)
- Requires a CPU/controller at memory nodes

CXL

- Does not copy data into NIC buffers
- Does not copy data across the network, accesses remote memory
- Provides a hardware supported coherent cache
- No need for CPU/controller at memory nodes

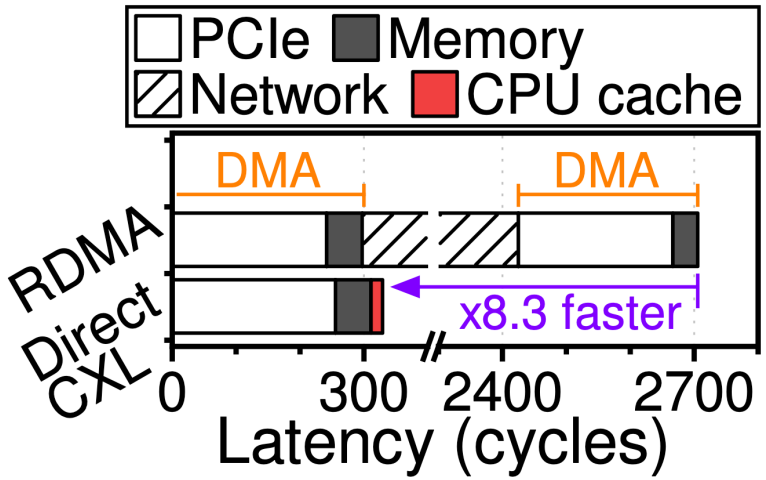
CXL vs RDMA

RDMA

- Copies data from application memory to NIC buffers
- Copies data across the network
- Requires handling of cache coherence (in some configurations)
- Requires a CPU/controller at memory nodes

CXL

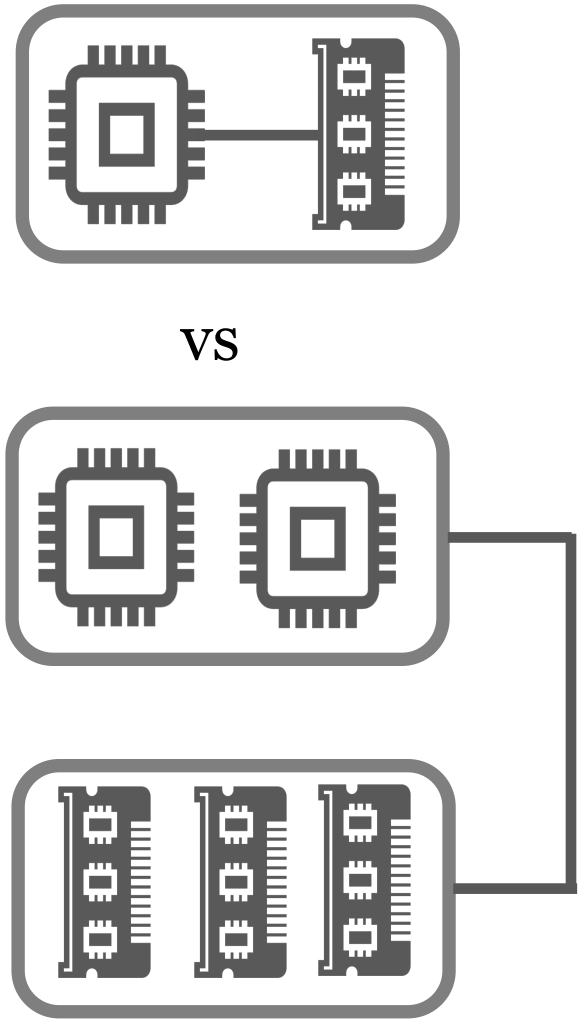
- Does not copy data into NIC buffers
- Does not copy data across the network, accesses remote memory
- Provides a hardware supported coherent cache
- No need for CPU/controller at memory nodes



[1] D. Gouk, et al. Direct Access, High Performance Memory Disaggregation with DirectCXL. *Proc. USENIX ATC*, 2022.

Main Issue with Memory Disaggregation

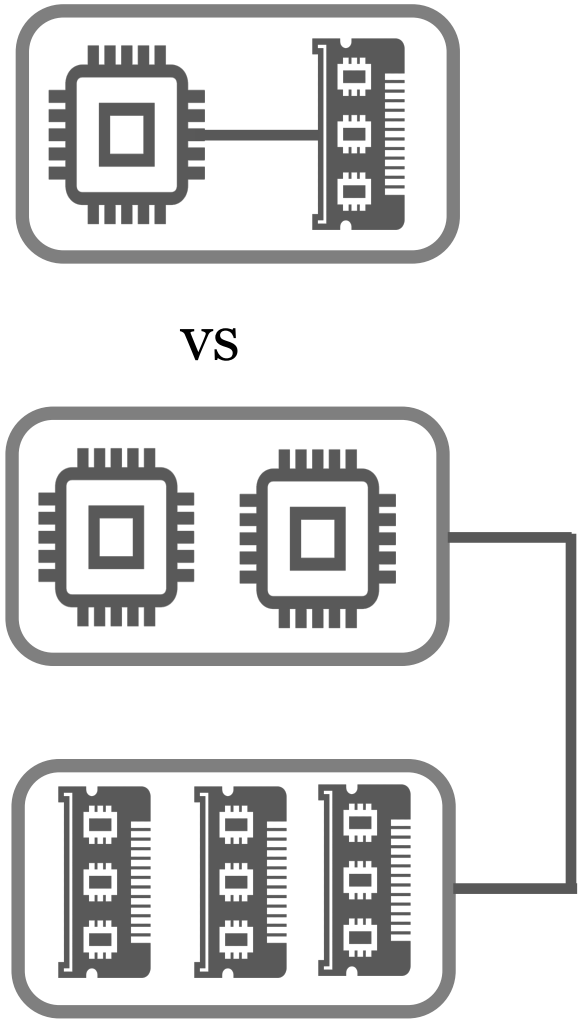
Access Latency



Main Issue with Memory Disaggregation

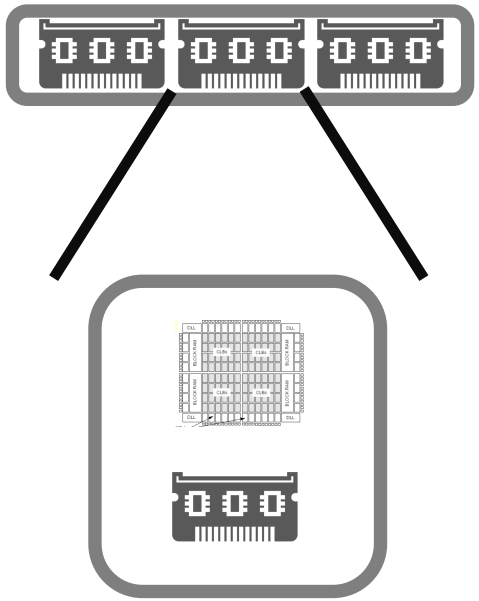
Access Latency

Near-data Processing
+
Optimized Design



Near-Data Processing: The Case for Smart Memory

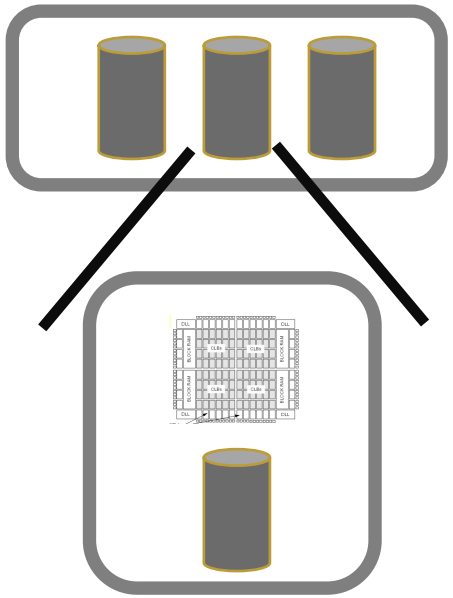
- Minimize data movement from memory to CPU
- Also
  - DRAM growing but not at the same speed as the demand
  - The independent provisioning issue again
- Farview: buffer pool on disaggregated memory
  - Push down operators to memory → reduce data movement
  - Centralize buffer cache on disaggregated memory → reduce memory requirement on compute nodes



D. Korolija et al. Farview: Disaggregated Memory with Operator Off-loading for Database Engines, *Proc. CIDR*, 2002

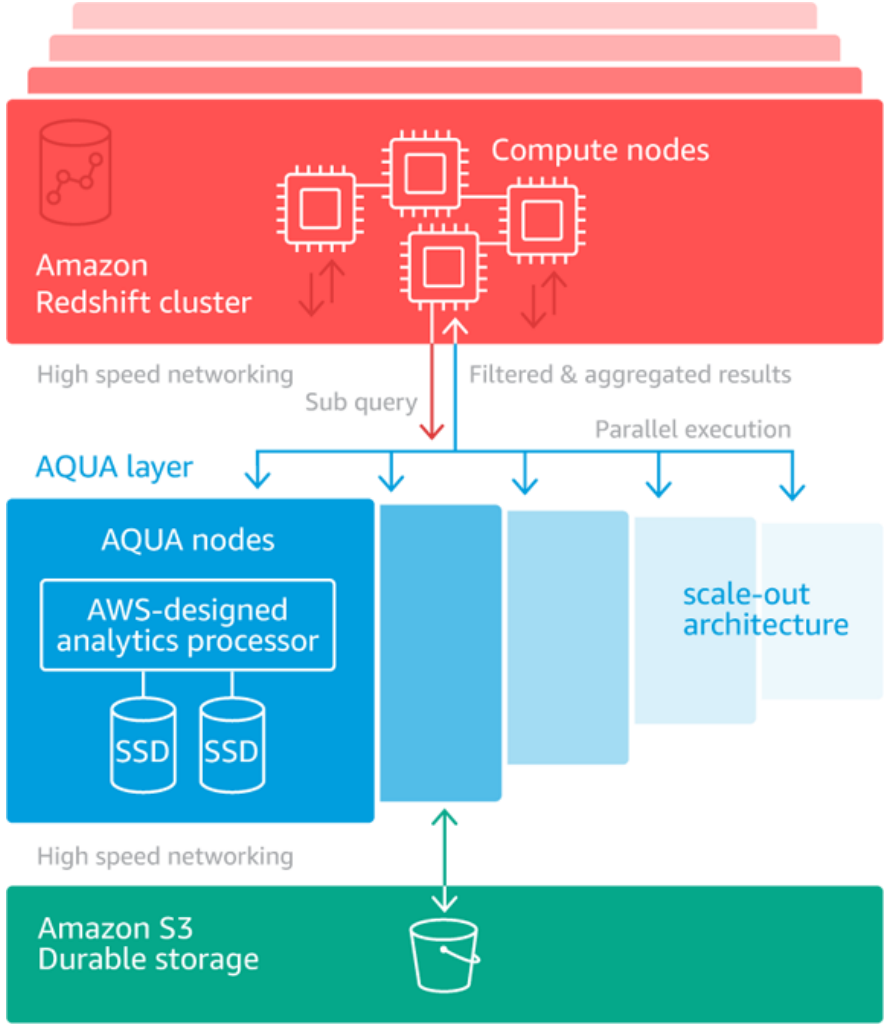
Near-Data Processing: The Case for Smart Storage

- Further restrict data movement
  - Pushing near-data computation to the storage
- Similar to old idea of database machines



Near-Data Processing: The Case for Smart Storage

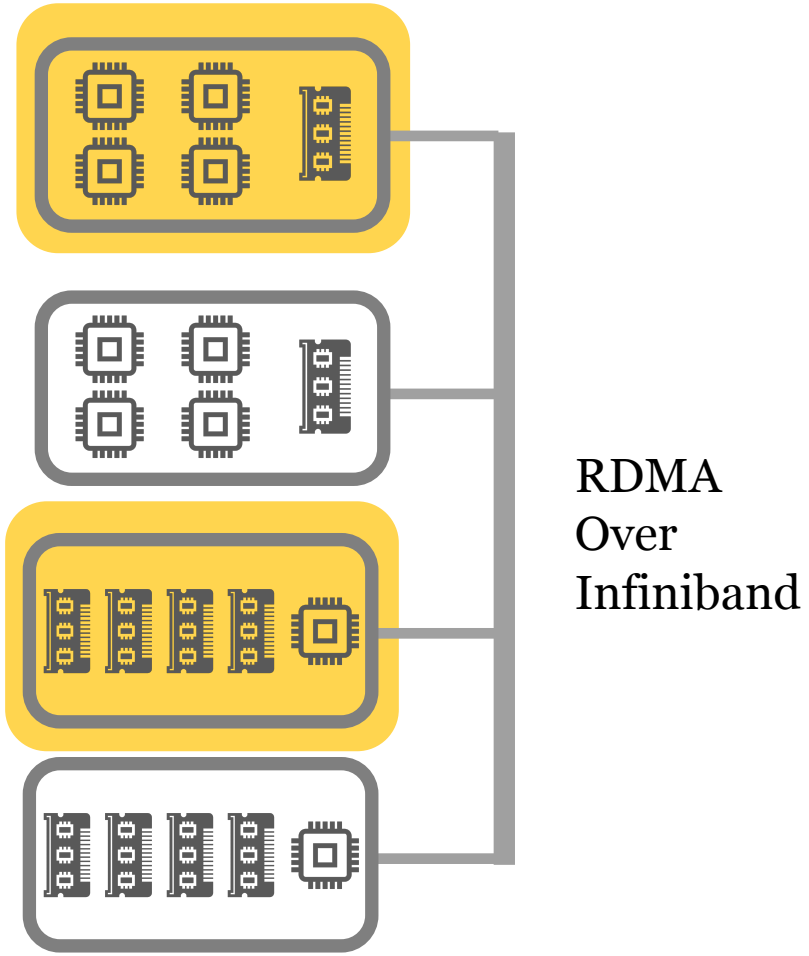
- Further restrict data movement
  - Pushing near-data computation to the storage
- Similar to old idea of database machines
- AWS Aqua nodes



J.Barr. AQUA (Advanced Query Accelerator) – A Speed Boost for Your Amazon Redshift Queries, 2021
https://aws.amazon.com/blogs/aws/new-aqua-advanced-query-accelerator-for-amazon-redshift/

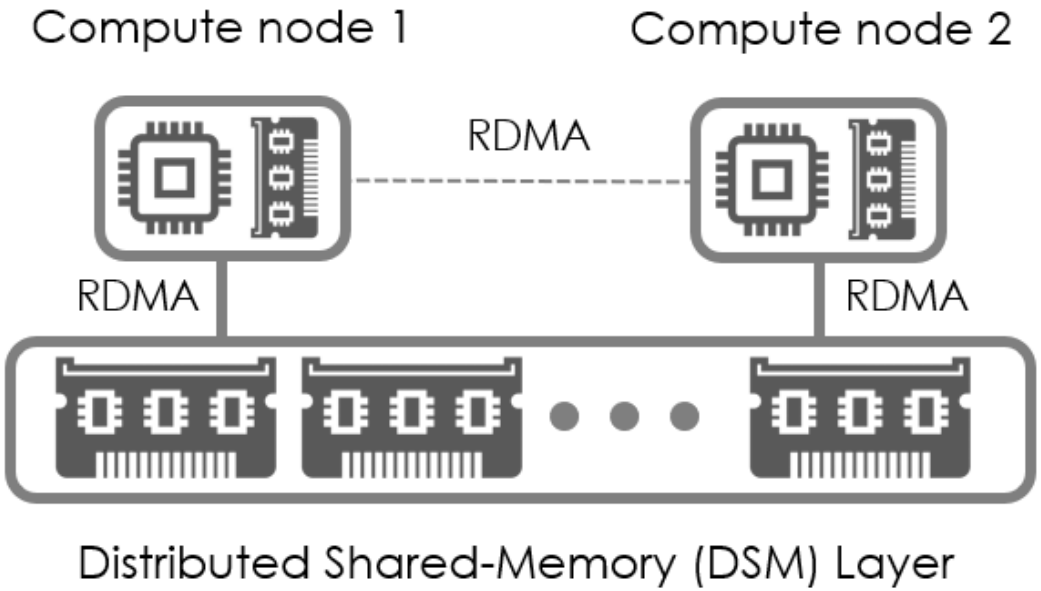
Our Work

- Scope: in-memory DBMS
- Compute nodes
  - Strong computing, but limited local memory
  - SQL parser, optimization, transaction, buffer
  - Each compute can read/write
- Memory nodes
  - Form a DSM layer accessed by all compute nodes



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

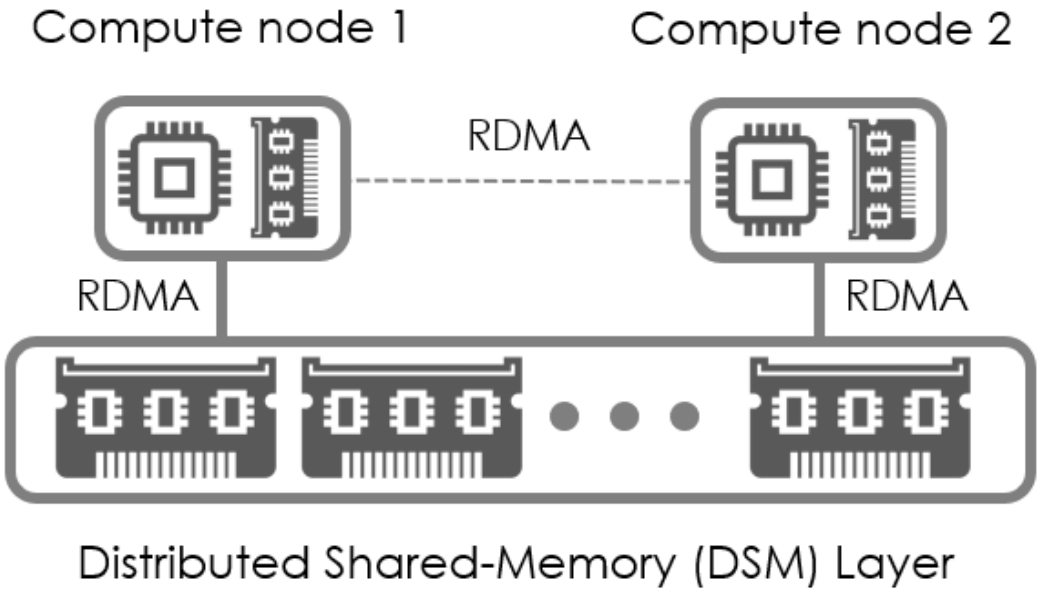
Transaction Execution Design



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Transaction Execution Design

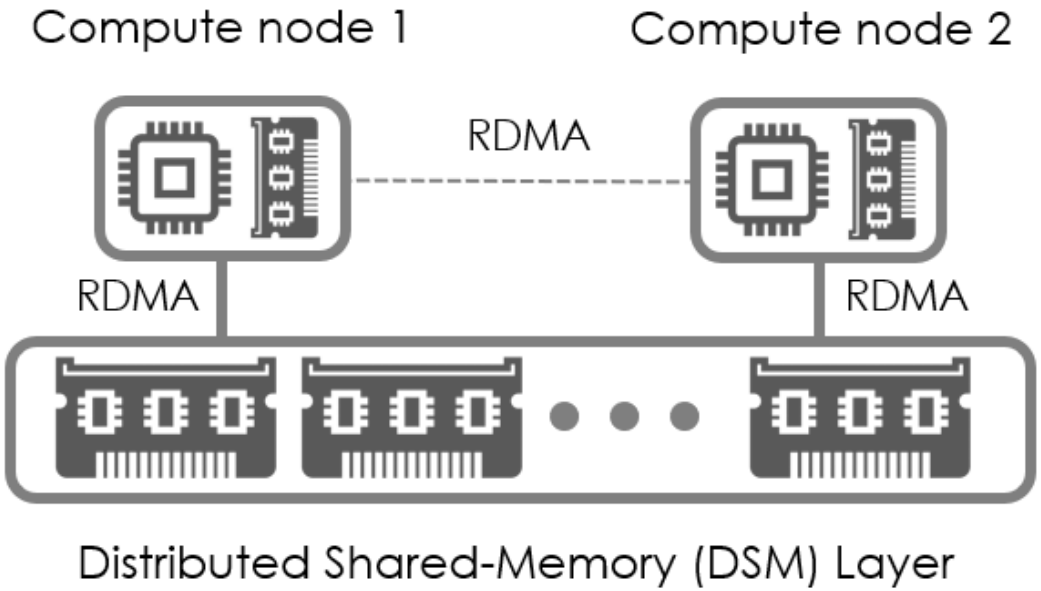
- Challenge: Cache coherence
  - This is not an issue if CXL is used
  - RDMA: No hardware-supported cache coherence between compute nodes



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Transaction Execution Design

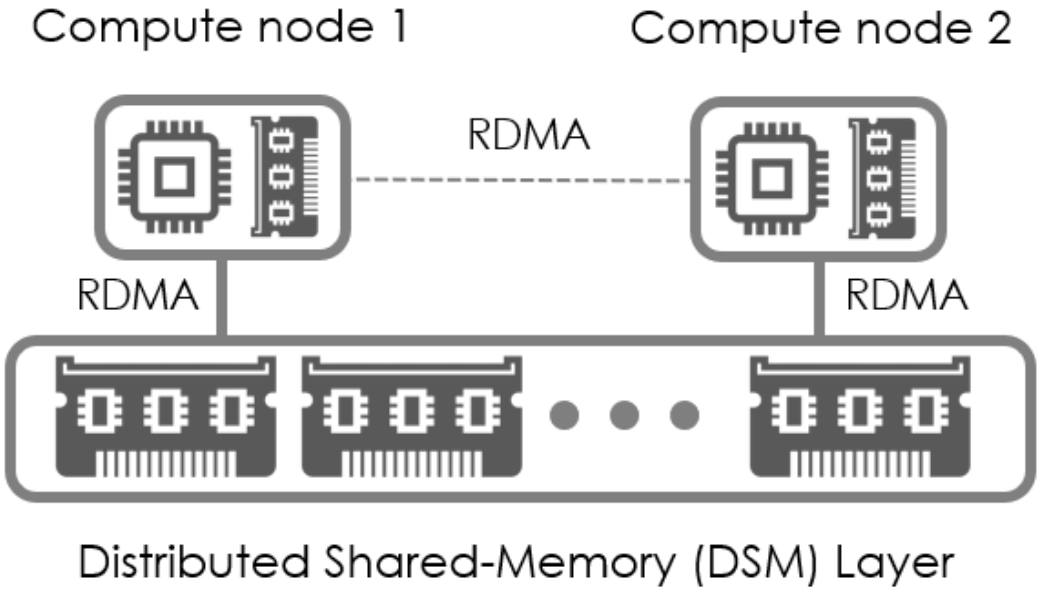
- Challenge: Cache coherence
  - This is not an issue if CXL is used
  - RDMA: No hardware-supported cache coherence between compute nodes
- Challenge: Distributed commit
  - Depends on the model



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Transaction Execution Design

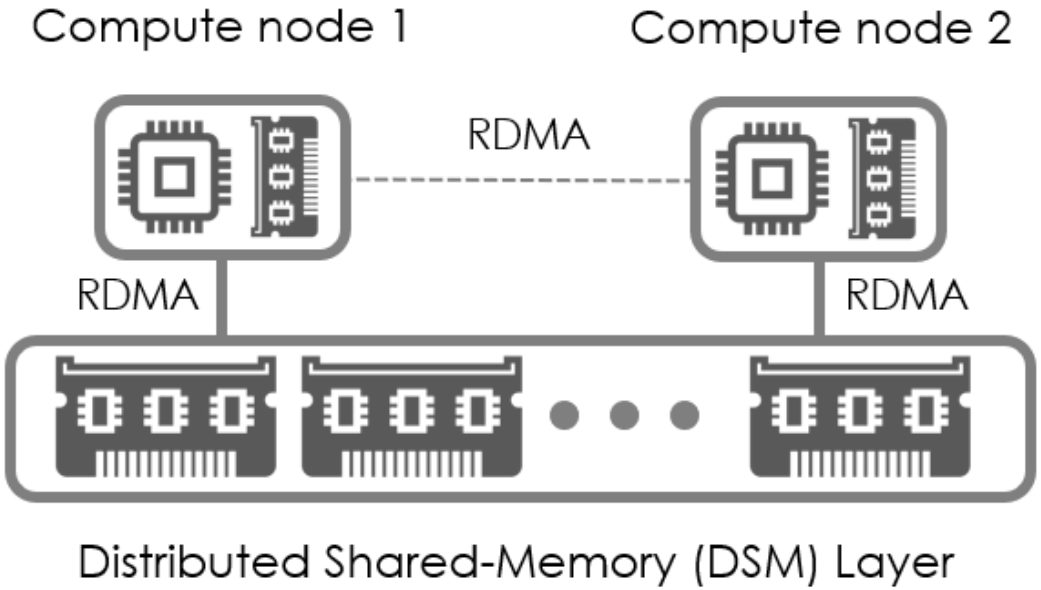
- Challenge: Cache coherence
  - This is not an issue if CXL is used
  - RDMA: No hardware-supported cache coherence between compute nodes
- Challenge: Distributed commit
  - Depends on the model
- Challenge: Concurrency Control Algs
  - Cannot directly use existing CC protocols, e.g., 2PL, MVCC, OCC



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Transaction Execution Design

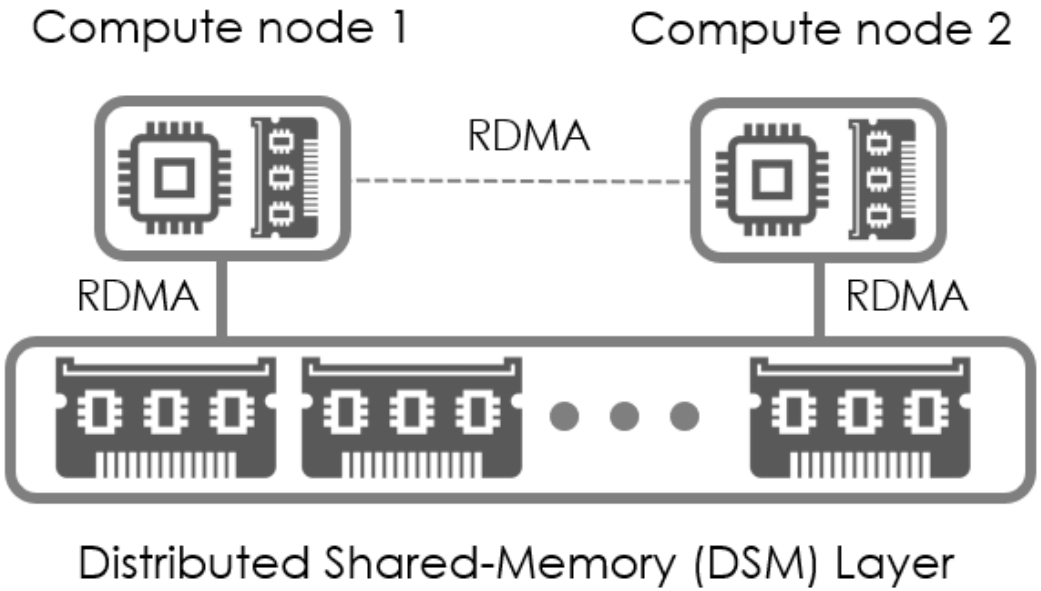
- Challenge: Cache coherence
  - This is not an issue if CXL is used
  - RDMA: No hardware-supported cache coherence between compute nodes
- Challenge: Distributed commit
  - Depends on the model
- Challenge: Concurrency Control Algs
  - Cannot directly use existing CC protocols, e.g., 2PL, MVCC, OCC
- Challenge: Massive concurrency
  - Main difference with multi-core single-node architecture



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Cache Coherence

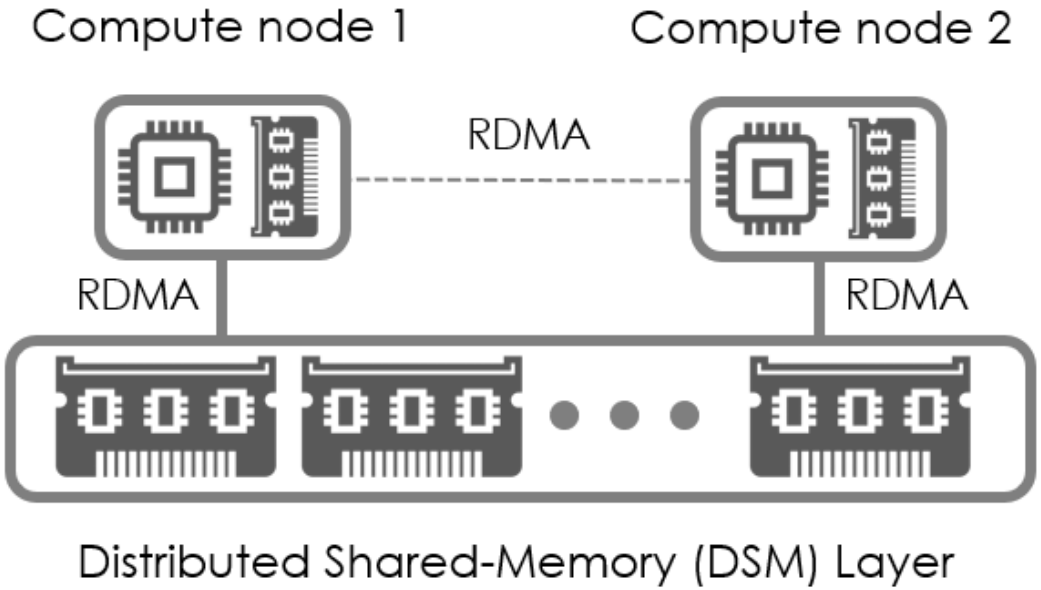
- Do we use the *local buffer* in each compute node?



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Cache Coherence

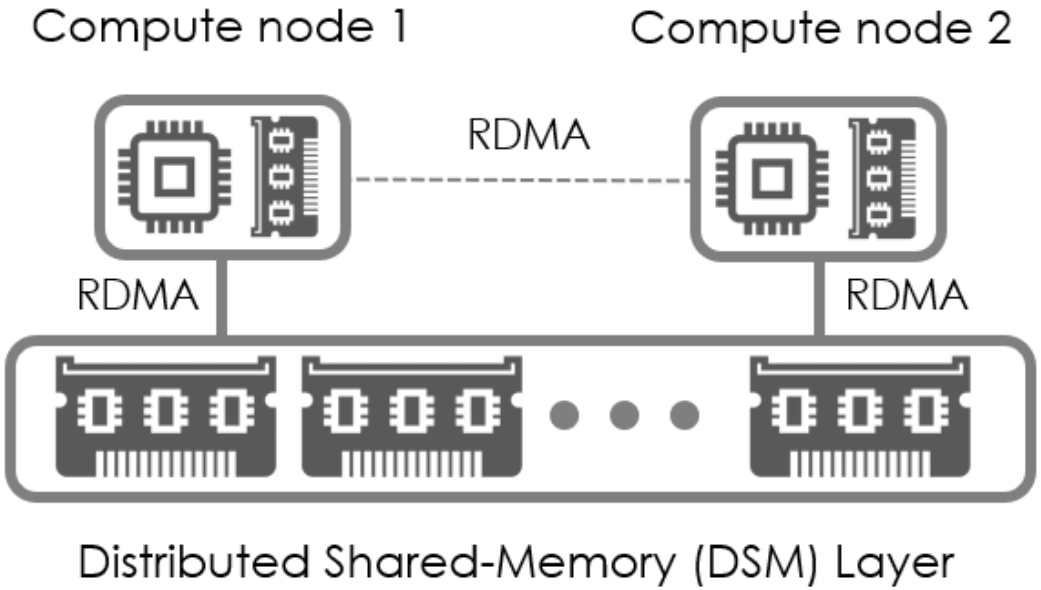
- Do we use the *local buffer* in each compute node?
  - If yes, need to address cache-coherence problem via software-level → overhead
  - If not, performance hit
    - many remote accesses



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Cache Coherence

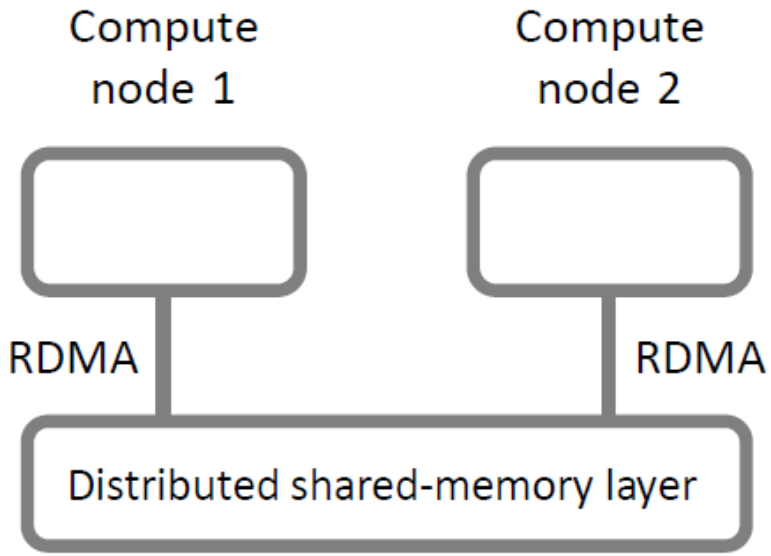
- Do we use the *local buffer* in each compute node?
  - If yes, need to address cache-coherence problem via software-level → overhead
  - If not, performance hit
    - many remote accesses
- Do we use *sharding* between compute nodes?



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Alternative #1: No Sharding, No Cache

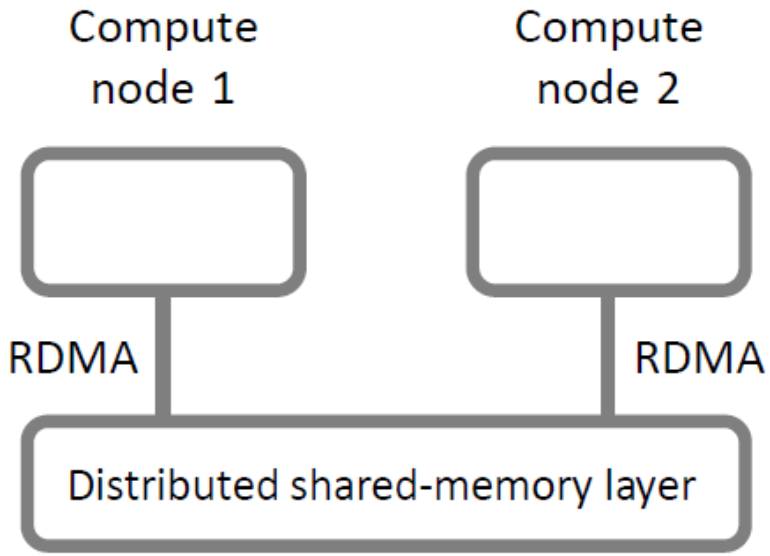
- Each compute node reads/writes all data
- But doesn't store any data in local buffer→ No cache coherence issue
- Data pages are stored in DSM



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Alternative #1: No Sharding, No Cache

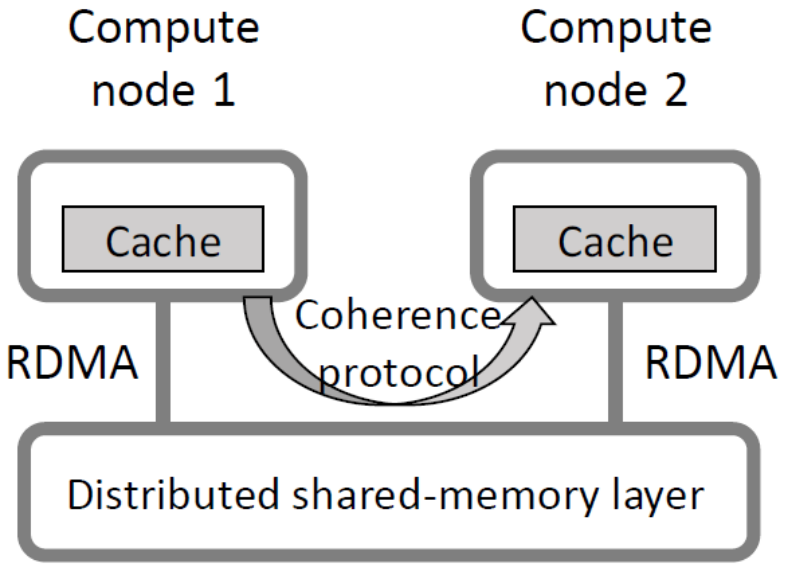
- Each compute node reads/writes all data
- But doesn't store any data in local buffer→ No cache coherence issue
- Data pages are stored in DSM
- All compute nodes use RDMA CAS (compare & swap) to acquire a lock first
- Not realistic → many remote accesses



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Alternative #2: No Sharding, With Cache

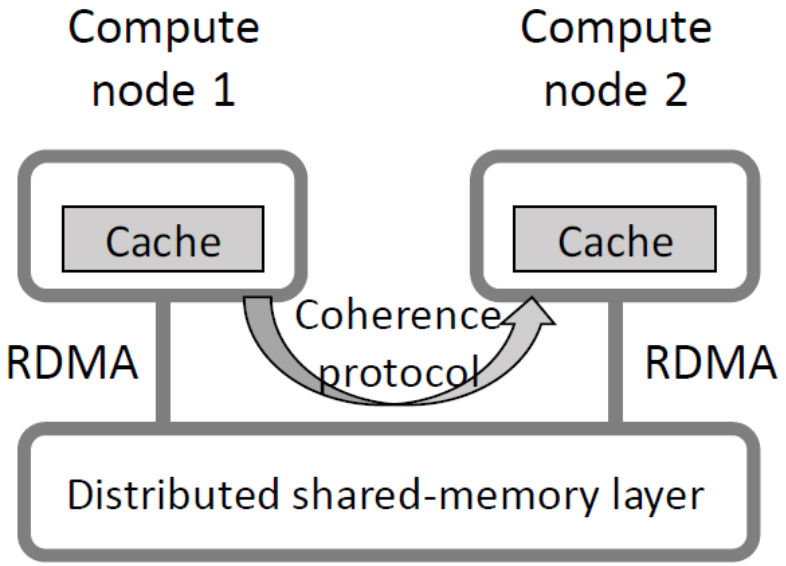
- Each compute node reads/writes all data
- Each compute node caches local data
- How to resolve conflicts?



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Alternative #2: No Sharding, With Cache

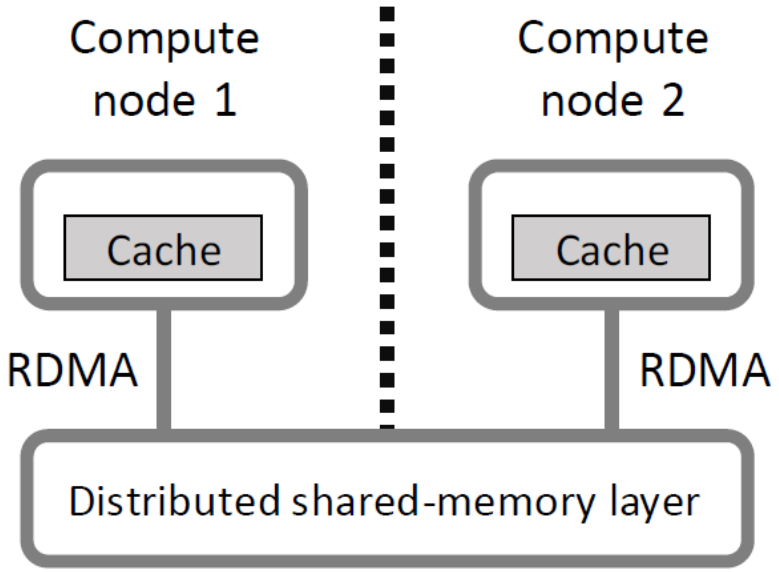
- Each compute node reads/writes all data
- Each compute node caches local data
- How to resolve conflicts?
- Develop a software-level cache coherence protocol
  - Update-based
  - Invalidation-based
- Overhead has to be measured and considered



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Alternative #3: With Sharding, With Cache

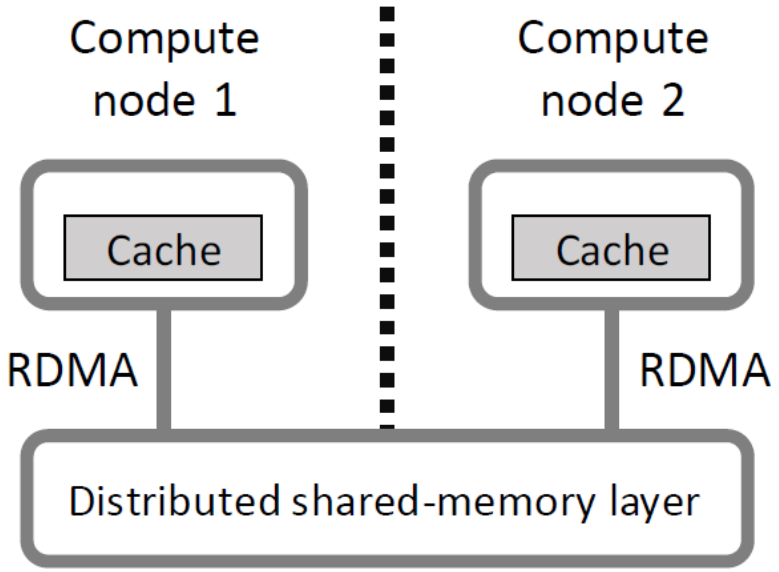
- Logical sharding among compute nodes
- Each compute node accesses a partition
- Use local buffer to cache data
- Bypass cache coherence issue
- But depending on workloads



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Alternative #3: With Sharding, With Cache

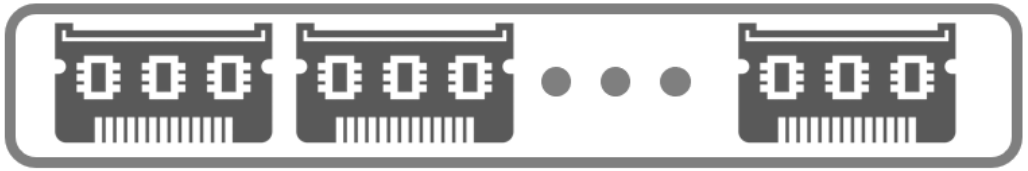
- Logical sharding among compute nodes
- Each compute node accesses a partition
- Use local buffer to cache data
- Bypass cache coherence issue
- But depending on workloads
- Similar to distributed shared-nothing
  - We do logical sharding (not physical sharding)
  - Also, the use of DSM layer can address distributed transaction and data skewness
  - Support elasticity very well



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Distributed Shared-Memory (DSM) Design

- Goal: manage a cluster of memory nodes, and expose necessary APIs to compute nodes
  - Looks like a single unified memory space

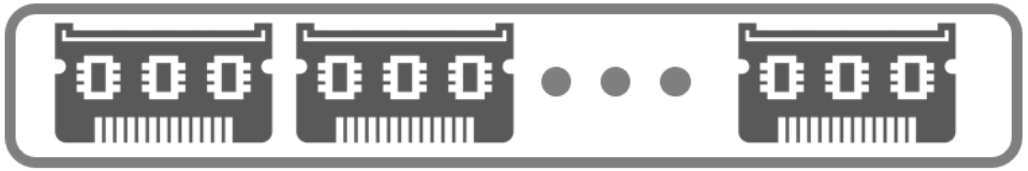


Distributed Shared-Memory (DSM) Layer

R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Distributed Shared-Memory (DSM) Design

- Goal: manage a cluster of memory nodes, and expose necessary APIs to compute nodes
  - Looks like a single unified memory space
- Challenge: Durability & Availability
- Challenge: Abstract APIs
  - Memory access APIs (also memory space representation)
  - Data transmission APIs (for RDMA)
  - Function offloading APIs (for DBMS ops)

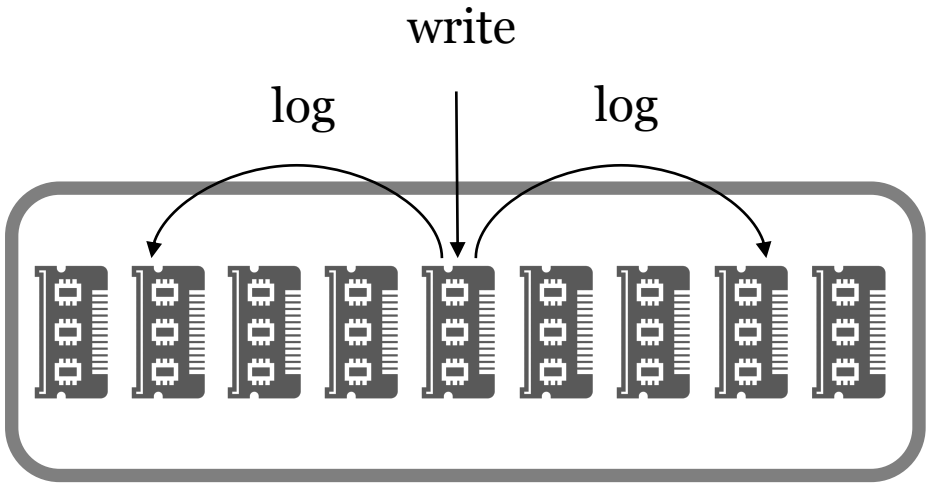


Distributed Shared-Memory (DSM) Layer

R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Durability & Availability

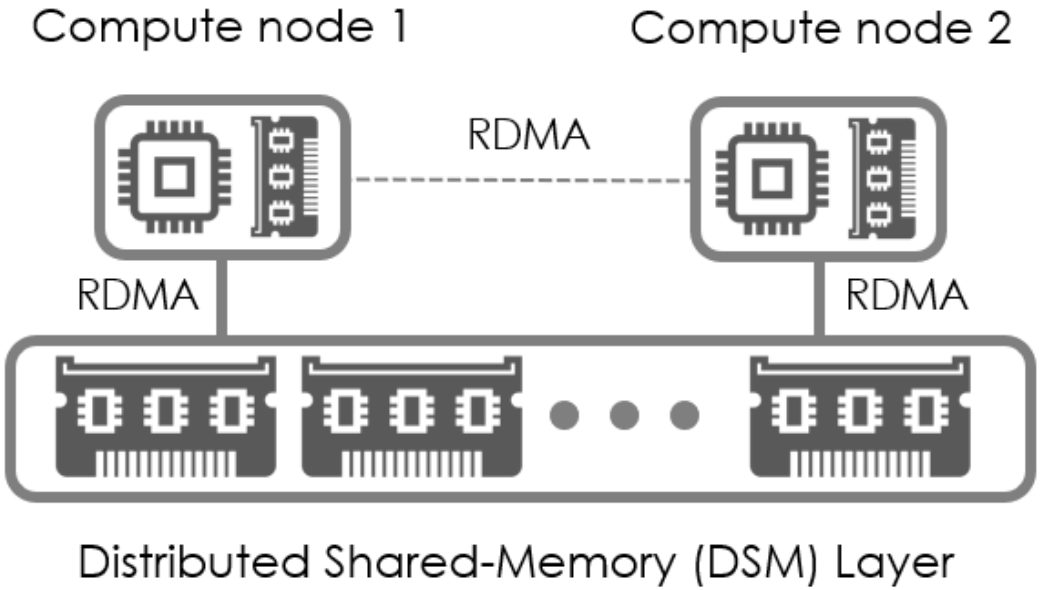
- Memory-based system
- Replication of data
  - Additional memory space + write performance
- Log-based solution (RAMCloud[1])
  - Separate logs from data (log store + data store)
  - Only logs are replicated in three memory nodes
    - A transaction is committed once all the logs are replicated and acknowledged
    - Data pages are asynchronously materialized
  - Data pages are stored only once in memory
  - Periodically checkpoint data pages to shared-storage to improve availability



[1] J. Ousterhout, et al. The Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM, *Proc. SIGOPS*, 2009.

Index Design Challenges

- Challenge: Leveraging RDMA characteristics (e.g., RDMA primitives)
- Challenge: How to use the buffer memory in compute nodes?
- Challenge: Leveraging RDMA byte-addressability
- Challenge: Leveraging near-data computing
- Challenge: How to support multi-node concurrent accesses?



R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

Rethinking Existing Indexes

Disk-based indexes

- E.g., B-tree, LSM-tree
- Not designed for RDMA (e.g., which primitive)
- Not optimized for byte-addressability
- Not leveraging near-data computing
- Multi-node concurrent accesses
  - E.g., use a single lock for both read and write or use read-write locks?

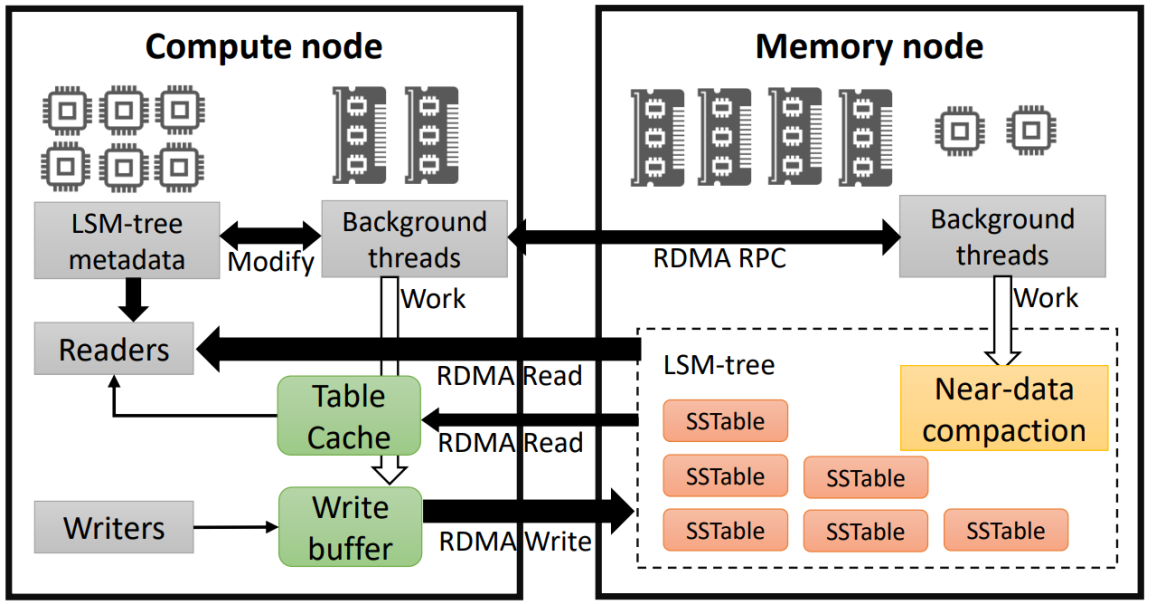
Memory-based indexes

- E.g., Bw-tree, Mass-tree, ART
- Not designed for RDMA (e.g., which primitive)
- Not leveraging local buffer
- Not leveraging near-data computing
- Multi-node concurrent accesses
  - E.g., lock-free vs. lock-based

R. Wang et al. The Case for Distributed Shared Memory Databases with RDMA-Enabled Memory Disaggregation, *Proc. VLDB*, 2022.

dLSM

- Compute node: MemTable, immutable, index blocks, bloom filters
- Memory node: SSTables (memory-optimized)
- One-sided RDMA read/write
- Reduce software overhead
  - Synchronization
  - Flushing
- Near-data compaction
- Optimize for byte-addressability
  - Can access a single KV pair



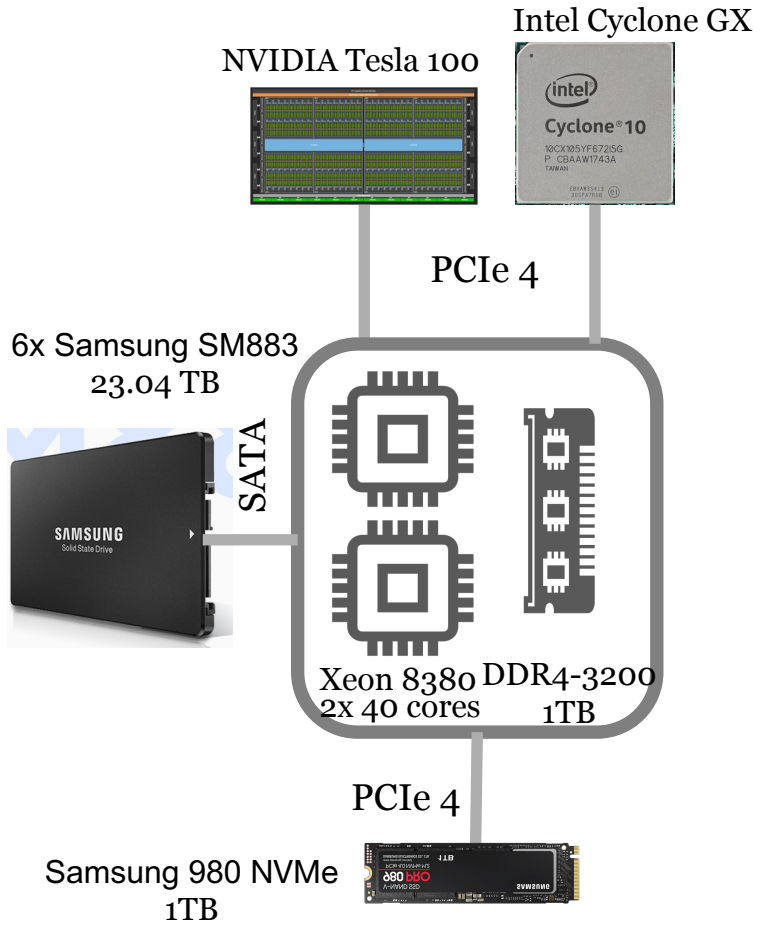
R. Wang et al. dLSM: An LSM-based Index for RDM-Enabled Memory Disaggregation, *Proc. ICDE*, 2023.

DISAGGREGATED HETEROGENEOUS PLATFORM

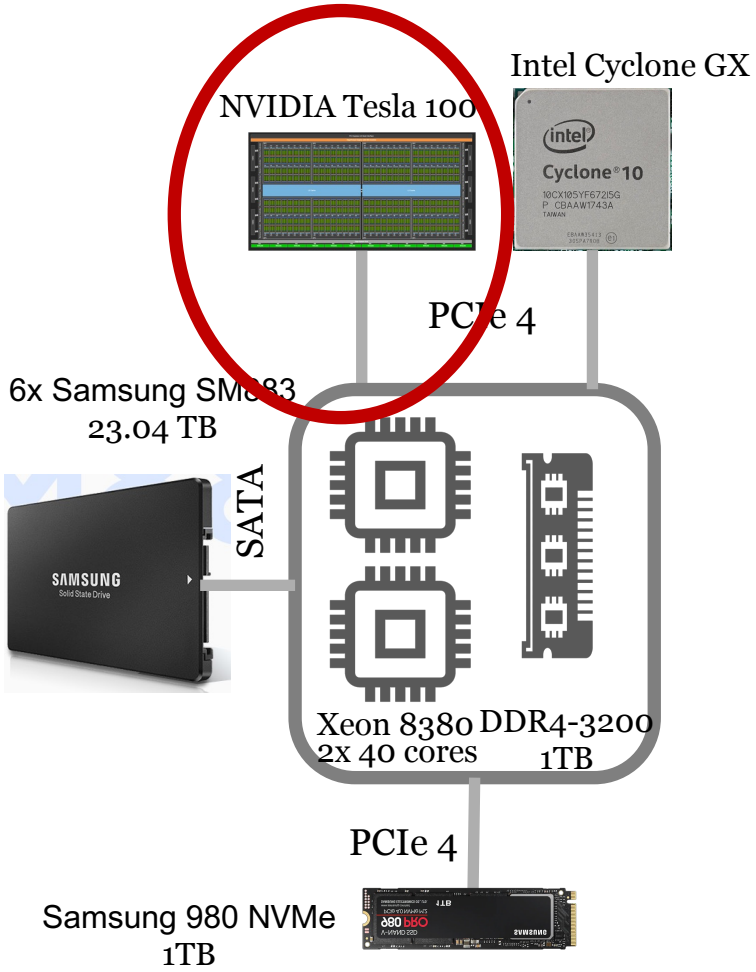
Heterogeneity

Case for Heterogeneous Computing

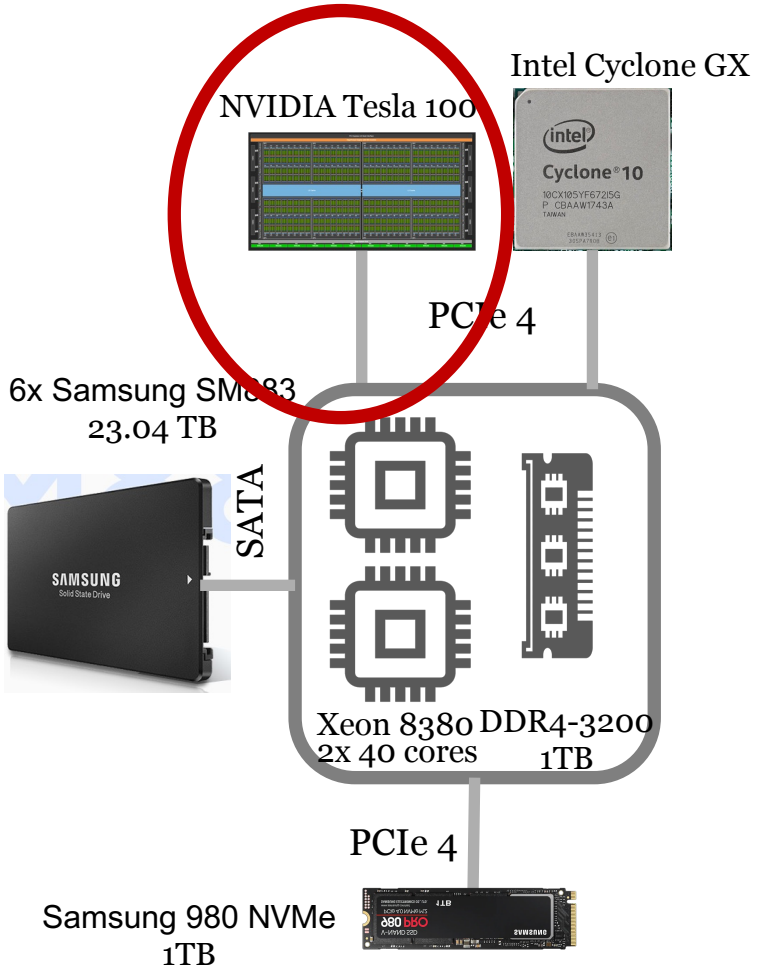
- Computing platforms are already heterogeneous
- Remember that CPU is not keeping up
- Data parallel computations of new workloads perform well on GPUs
- FPGAs are more flexible and can serve to experiment with solutions
- High volume demand for same task might be best met by ASICs



Data Management over GPU/CPU-GPU

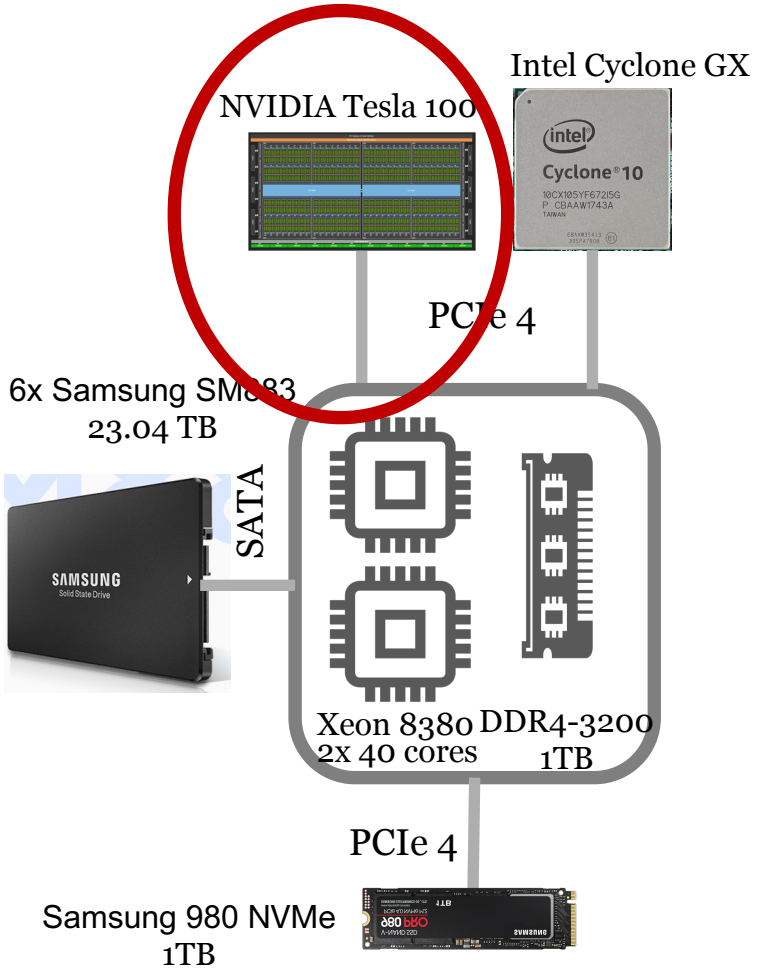


Data Management over GPU/CPU-GPU



Two approaches

Data Management over GPU/CPU-GPU

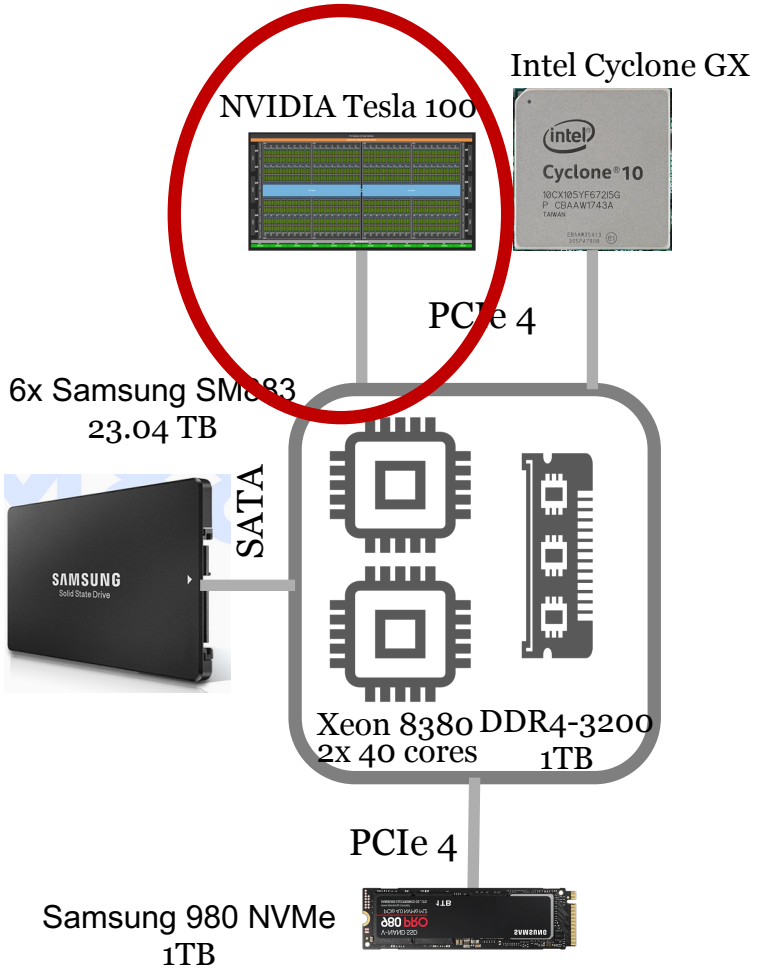


Two approaches

- Transfer data to GPU on-demand[1]
  - PCIe bandwidth is an issue
  - PCIe bottleneck sometimes causes performance < optimized CPU implementations
  - Store working set directly on GPU[2]

[1] J. Li et al. HippogriffDB: Balancing I/O and GPU Bandwidth in Big Data Analytics, *Proc. VLDB*, 2016.
[2] A. Shanbhag, S. Madden and X. Yu. A Study of the Fundamental Performance Characteristics of GPUs and CPUs for Database Analytics, *Proc. SIGMOD*, 2020.

Data Management over GPU/CPU-GPU

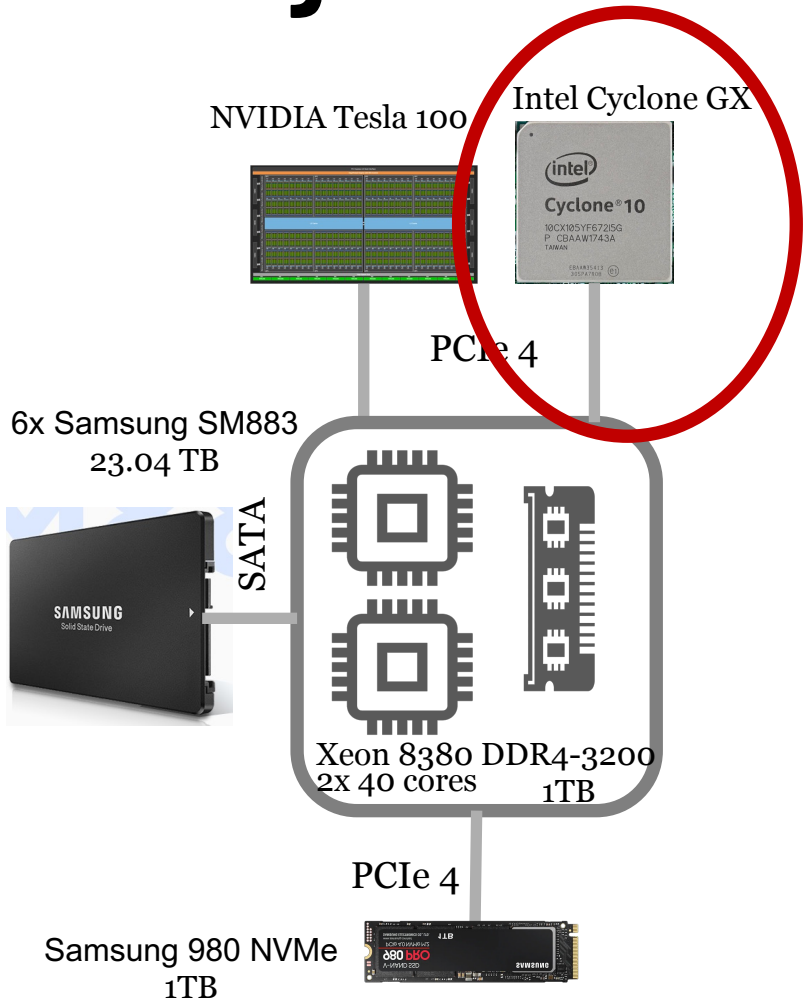


Two approaches

- Transfer data to GPU on-demand[1]
  - PCIe bandwidth is an issue
  - PCIe bottleneck sometimes causes performance < optimized CPU implementations
  - Store working set directly on GPU[2]
- CPU-GPU heterogeneous processing[3,4]
  - Design more complicated
  - Data placement – treat GPU memory as cache[4]
  - Finer granularity cache → more complex query execution
  - Cost model to decide where to execute what

[3] P. Chrysogelos et al. HeatExchange: Encapsulating Heterogeneous CPU-GPU Parallelism in JIT Compiled Engines, *Proc. VLDB*, 2019.
[4] B. W. Yogatama, W. Gong and X. Yu. Orchestrating Data Placement and Query Execution in Heterogeneous CPU-GPU DBMS, *Proc. VLDB*, 2022.

Data Management over FPGA/CPU-FPGA



"FPGAs … configure different parts of the design space offering advantages that other current options do not have:

- line rate processing;
- enabling processing streams of data out of the network, disks, or memory without performance loss;
- architectural flexibility in that they can be inserted in places where the other type of processor cannot be used such as in NICs, in storage, in memory, etc. …; and
- the customizable nature of reconfigurable computing with the FPGA serving equally well to accelerate network function virtualization, data reorganization in a database, or to accelerate joins."

W. Jiang, D. Korolija and G. Alonso. Data Processing with FPGAs on Modern Architectures, *Proc. SIGMOD*, 2023.

FINAL WORDS

Environment is Getting Interesting

- Other h/w developments that can help (but I don't know much about):
  - Tensor Processing Unit (TPU)
  - Data Processing Unit (DPU)
  - Non-Volatile Memory (NVM)
- Design space is varied and large
  - But start somewhere

UNIVERSITY OF WATERLOO | DSg Data Systems Group

Thank you