# Vertically integrated storage systems

Gustavo Alonso

Systems Group

Department of Computer Science

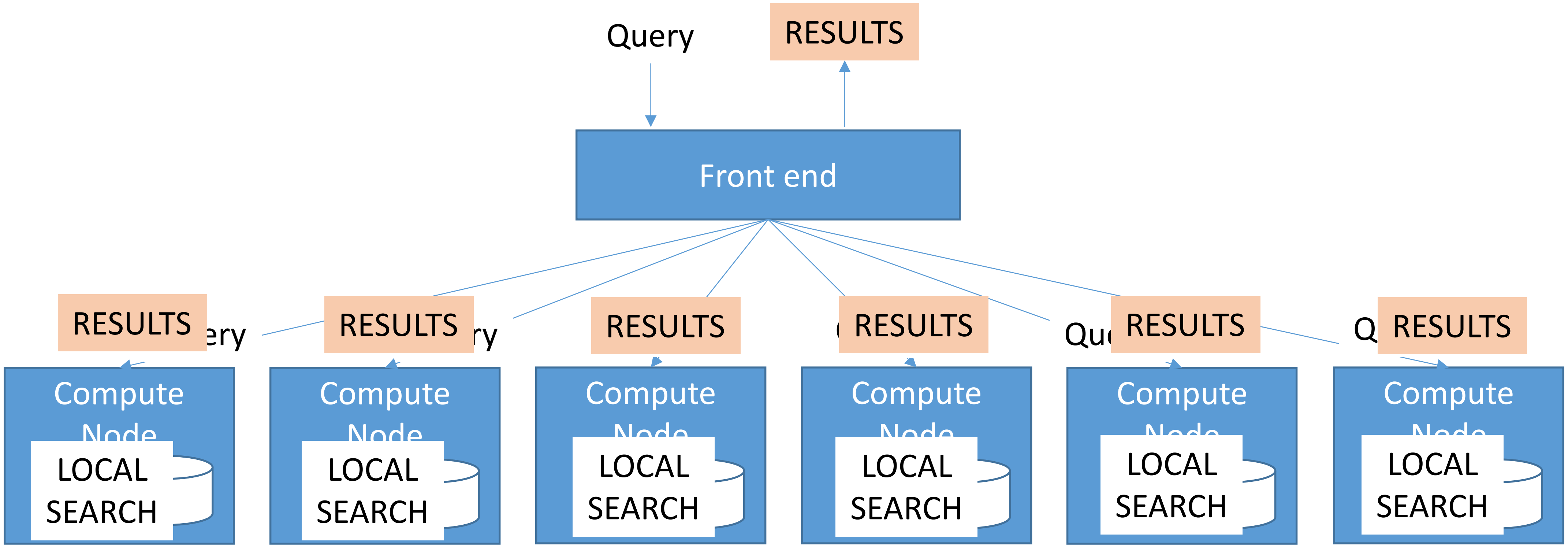
ETH Zurich, Switzerland

# Agenda

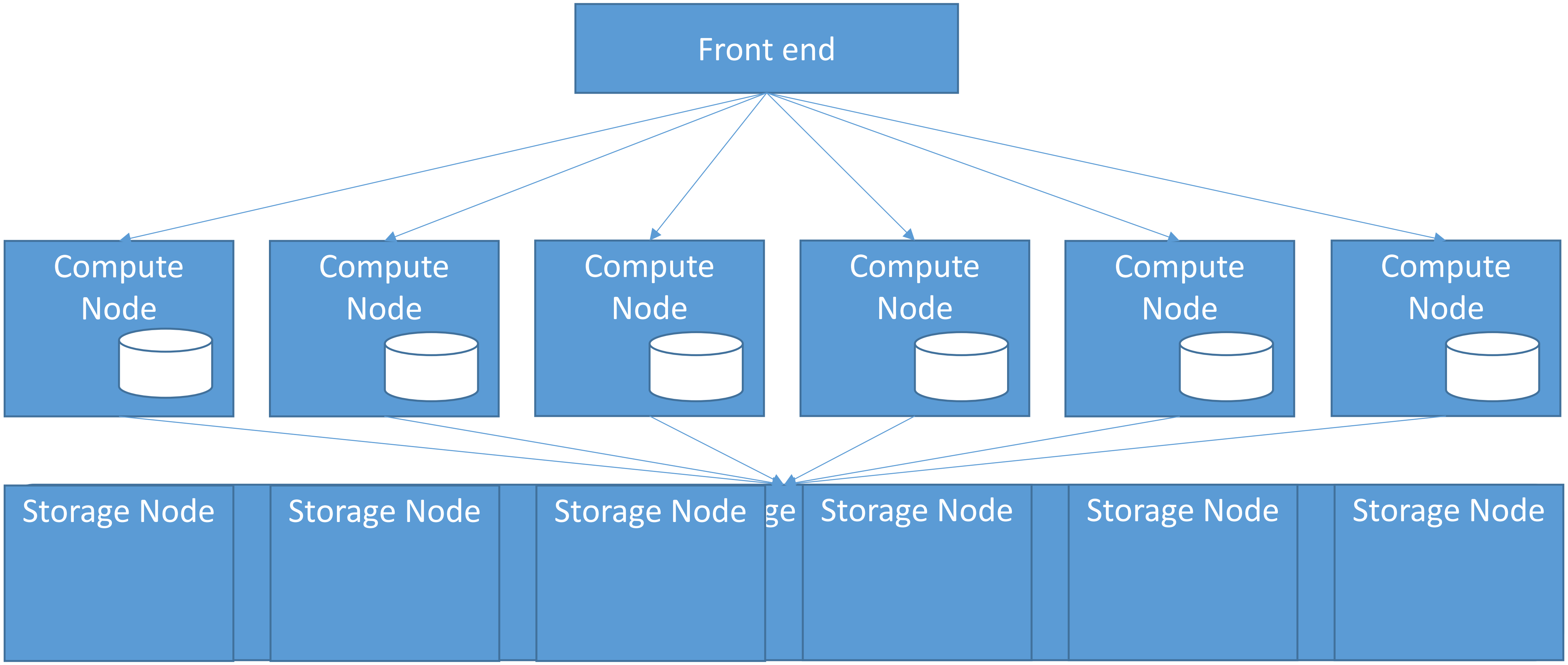
- The status quo:
  - How systems in the cloud operate today: disaggregation
- The grand vision
  - Vertically integrated storage instead of disaggregation
- Reality checks:
  - Does the necessary technology exist?
- The motivation
  - What can be done with vertically integrated storage
- How to get there
  - Building the infrastructure for vertically integrated storage

# The status quo

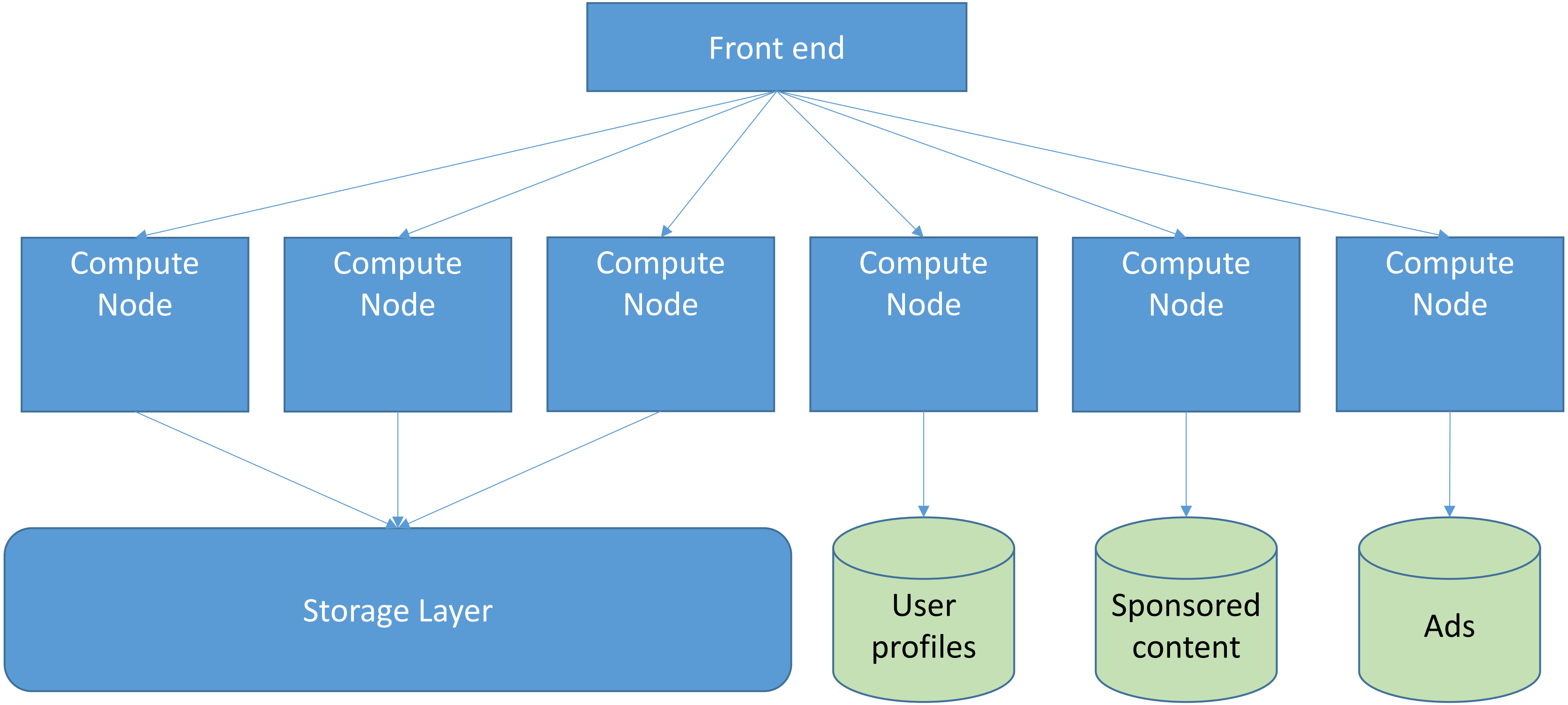
# The cloud is a search engine



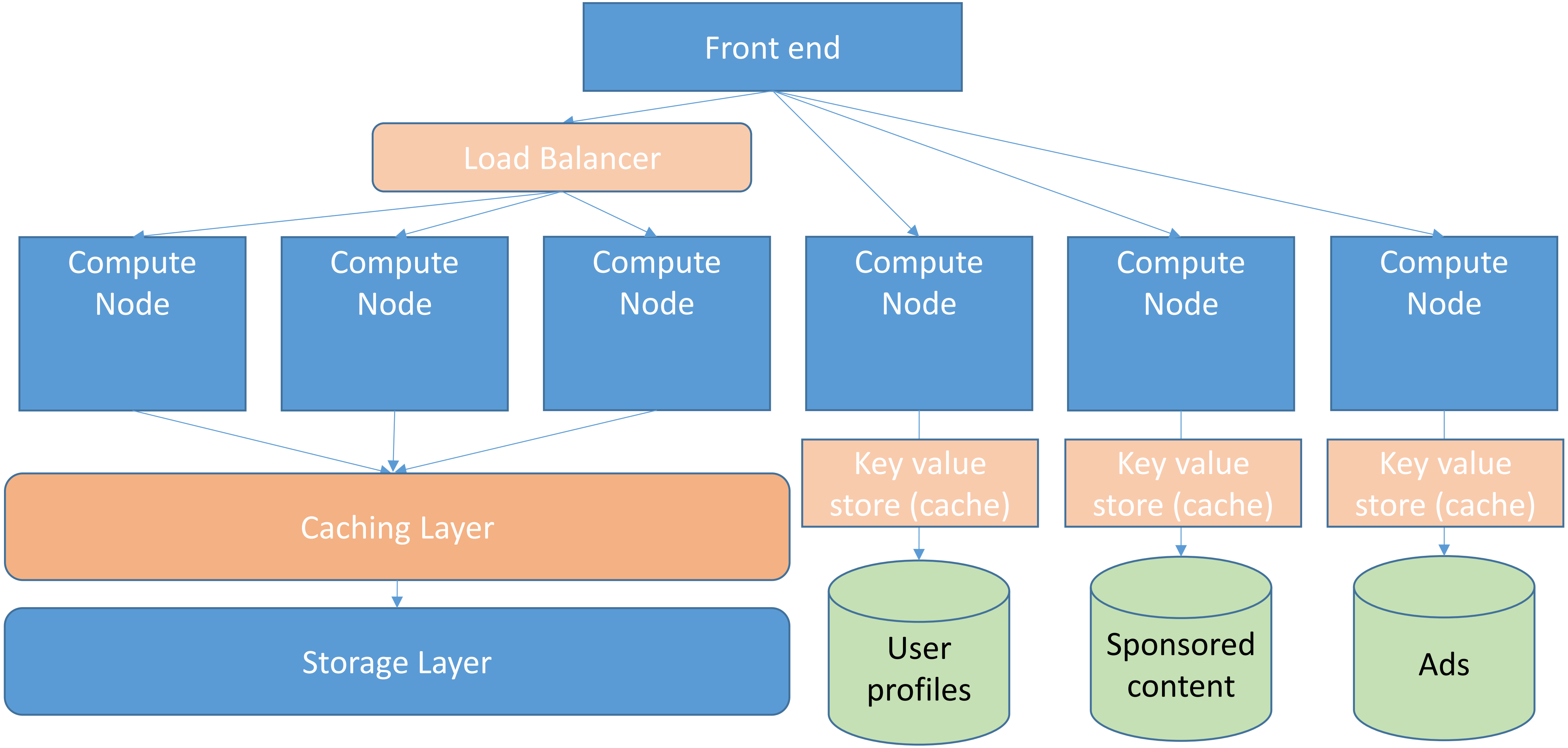
# Separation of compute and storage



# Look at different sources



# Your application is not a search engine

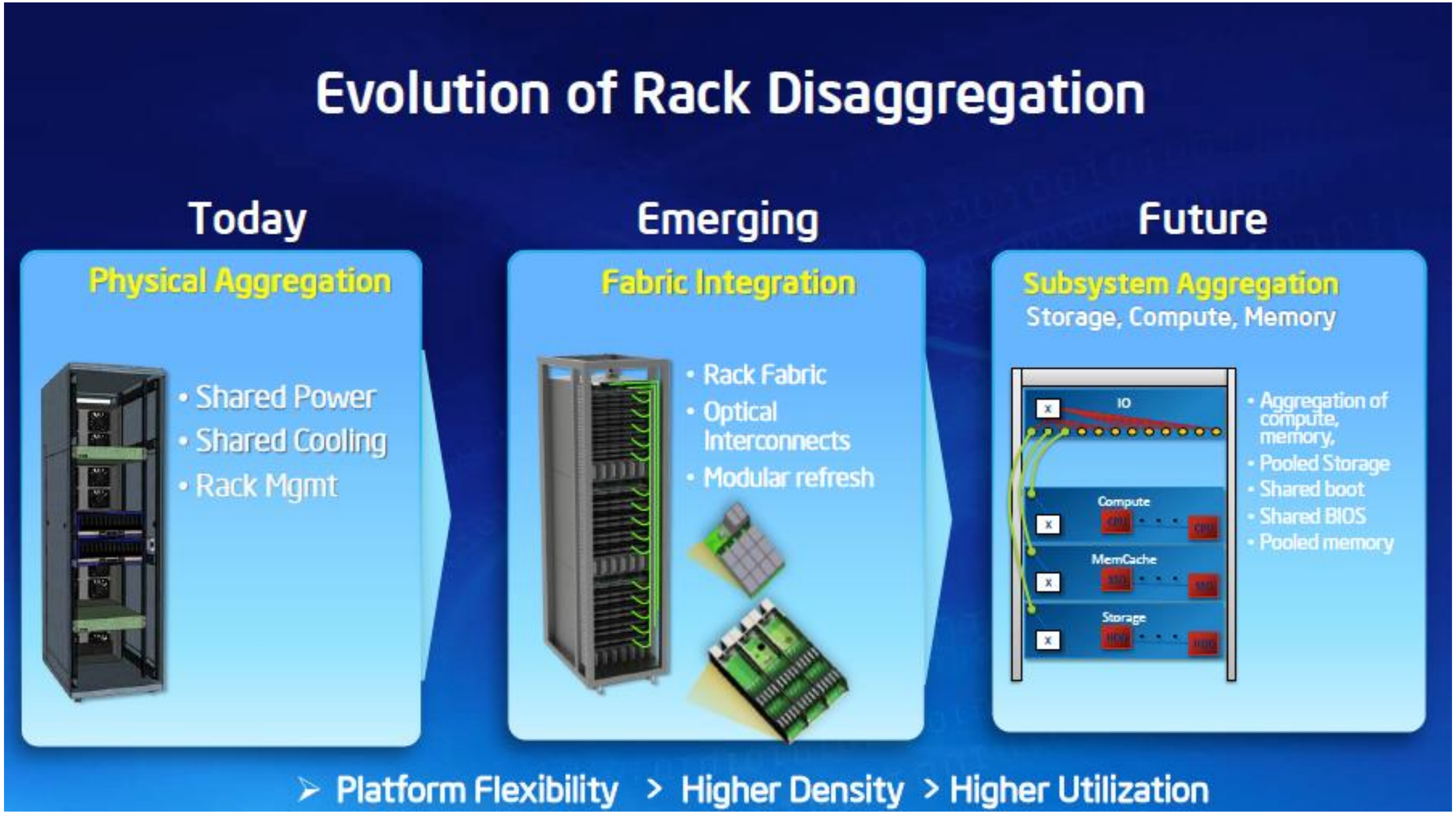


# The dominant architecture

- Cloud architecture has dominated the landscape in the last two decades
- This is changing and changing fast
  - Acceptance that some things do not work in a disaggregated, scale out architecture
  - Recognition that the architecture is highly inefficient and wasteful
  - As architectures become data centric, they tend to focus more on the storage and memory rather than the compute

# The grand vision

# A vision (Intel's) of disaggregation



## Evolution of Rack Disaggregation

**Today**

**Physical Aggregation**

- Shared Power
- Shared Cooling
- Rack Mgmt

**Emerging**

**Fabric Integration**

- Rack Fabric
- Optical Interconnects
- Modular refresh

**Future**

**Subsystem Aggregation**
Storage, Compute, Memory

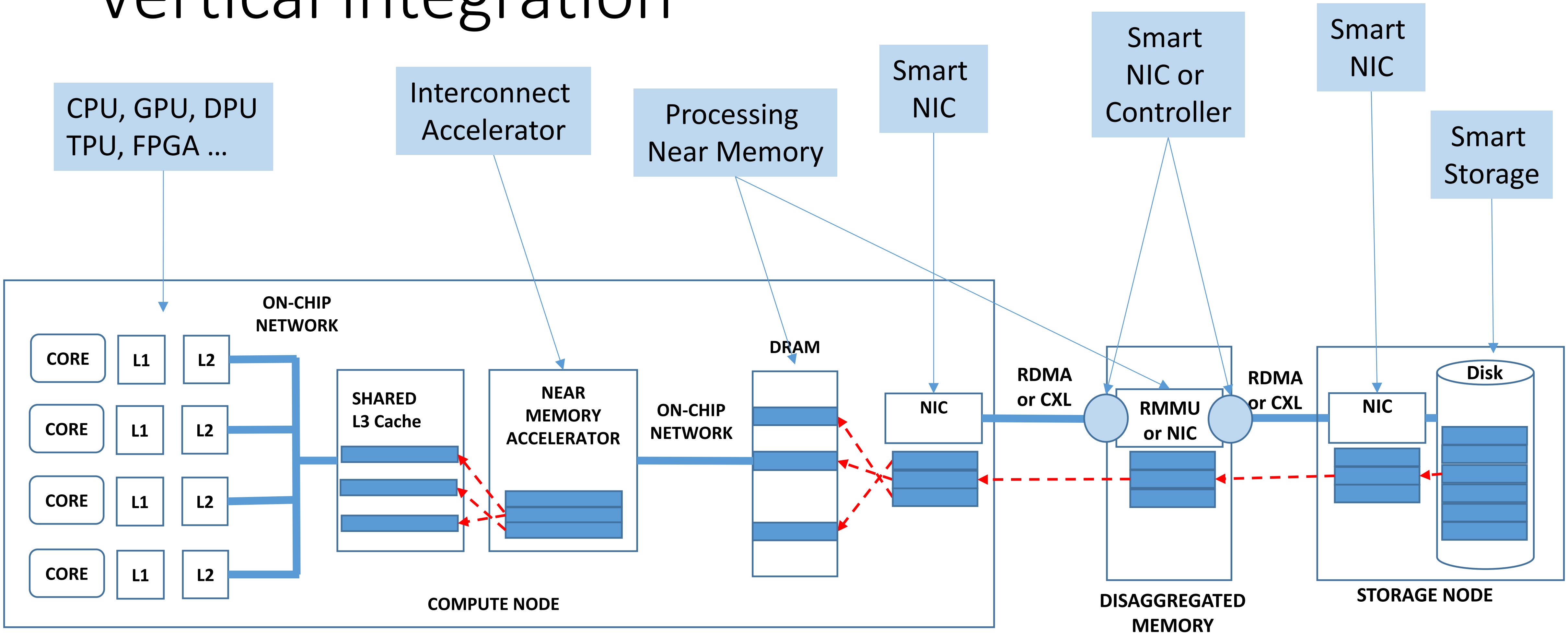
- Aggregation of compute, memory,
- Pooled Storage
- Shared boot
- Shared BIOS
- Pooled memory

➢ Platform Flexibility > Higher Density > Higher Utilization

# Disaggregation

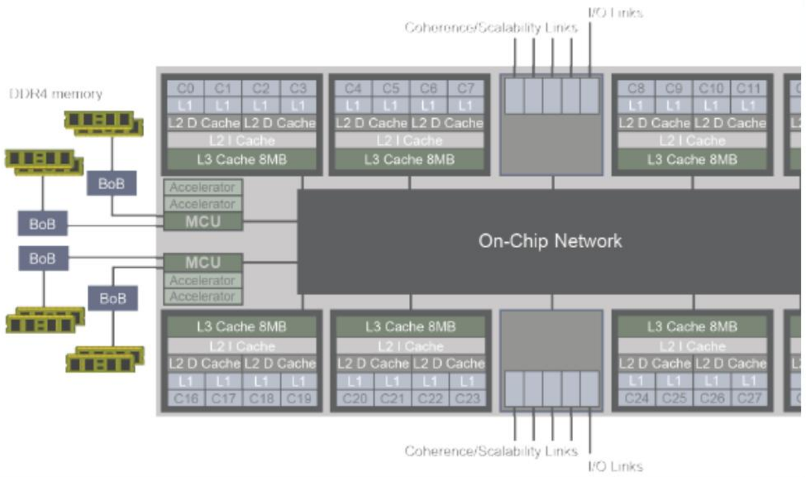
- Disaggregated storage provides elasticity for compute
- But results in a higher price for data movement:
  - Long data paths from storage to compute
  - Many unnecessary data movements
  - A lot of overhead in reading and writing to storage (compression, encryption, data transformations, parallel I/O for performance, replication, etc.)
- There is a way to minimize the price of disaggregation …

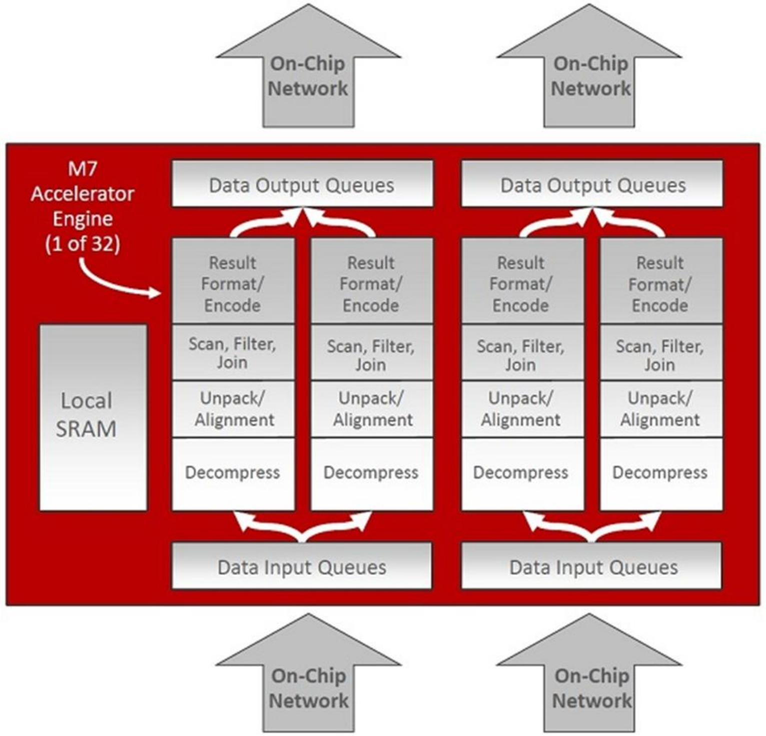
# Vertical integration

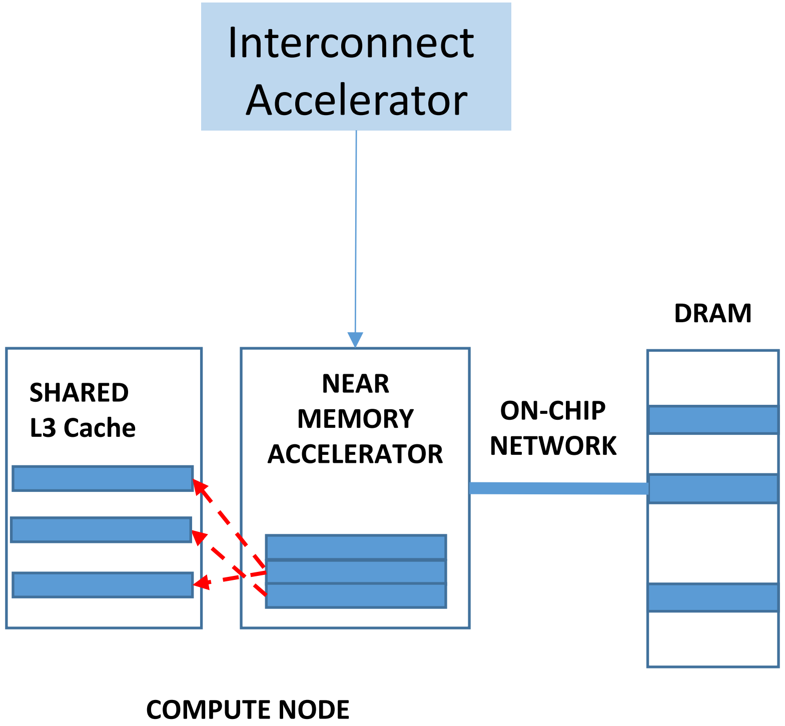


# The reality check

# Near memory accelerator (Oracle Sparc M7)







# Smart NIC (Azure Boost for storage)

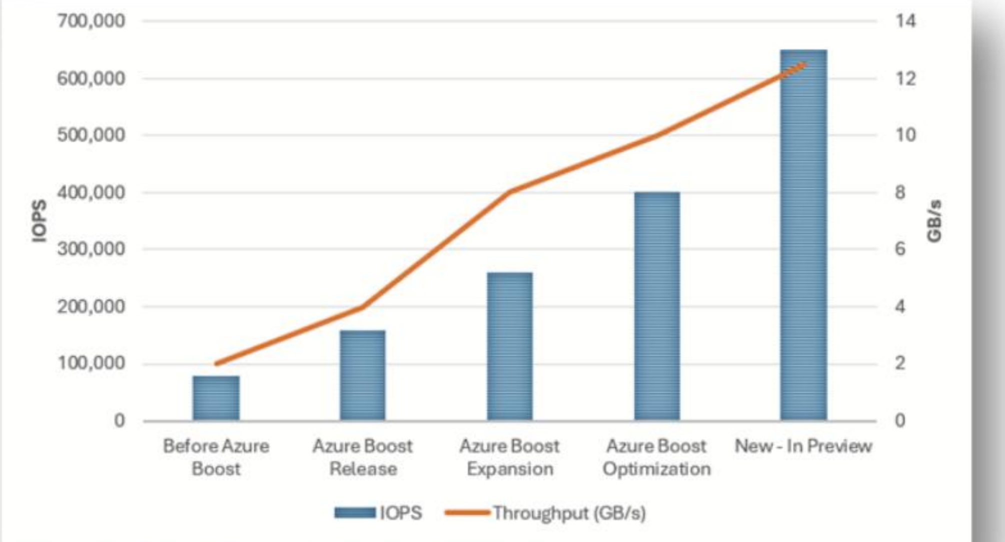
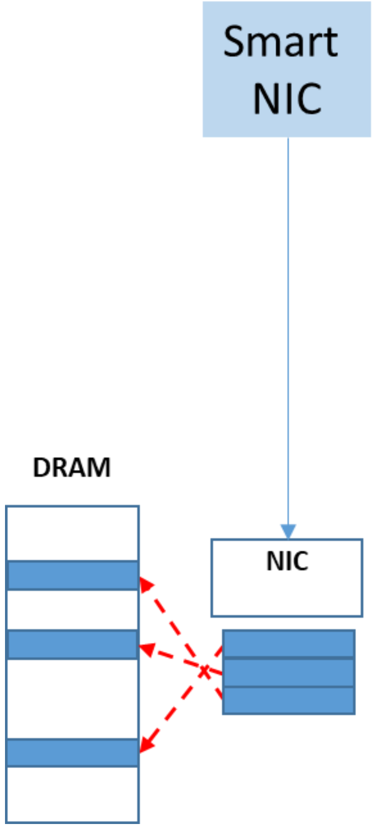


*Figure 1: Azure Managed Disks IOPS & Throughput Enhancements*

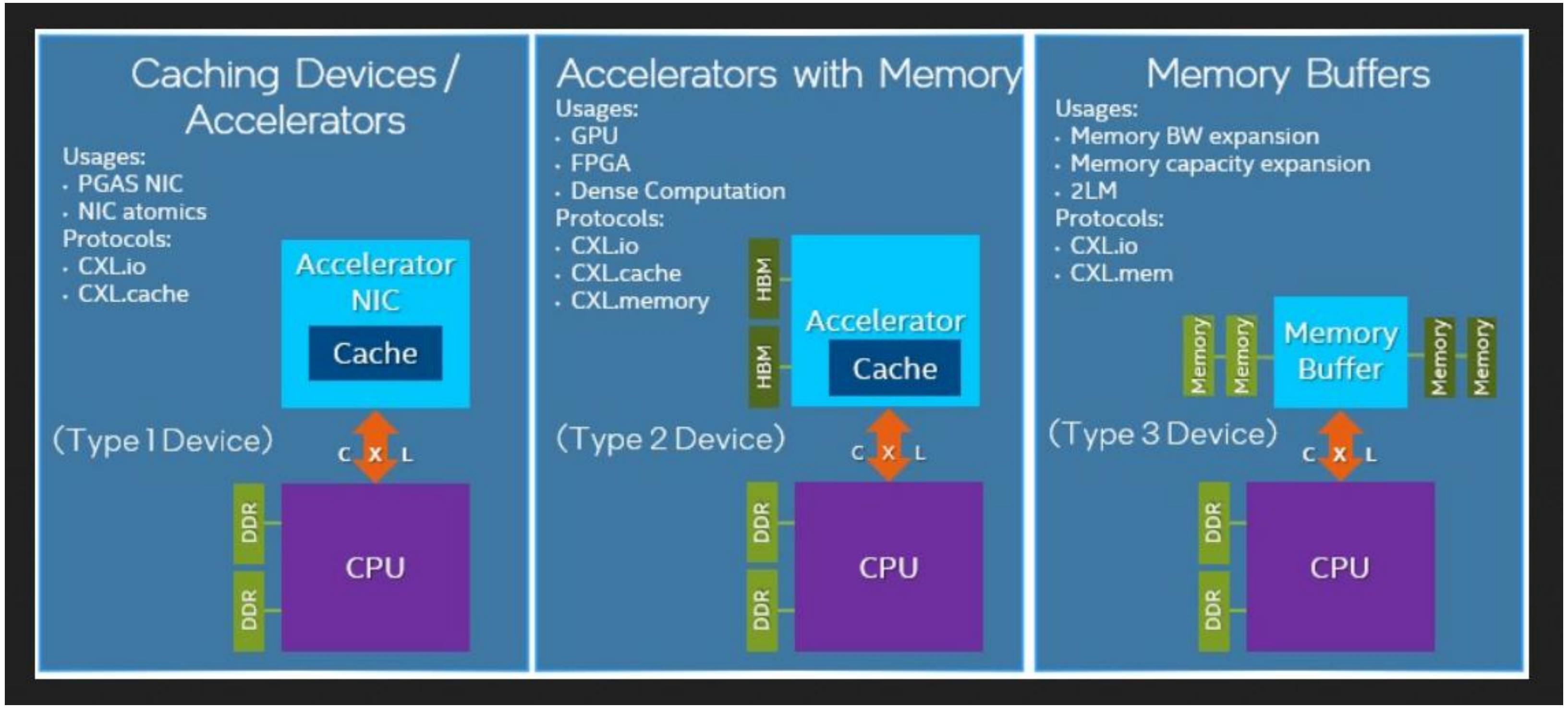


Smart NIC

DRAM

NIC

# Smart CXL disaggregated memory



## Caching Devices / Accelerators

Usages:
- PGAS NIC
- NIC atomics

Protocols:
- CXL.io
- CXL.cache

## Accelerators with Memory

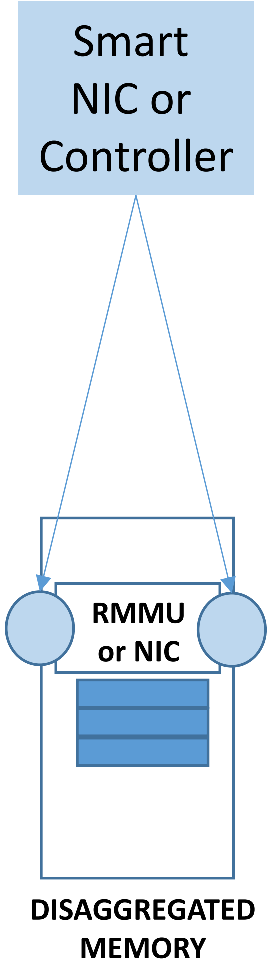
Usages:
- GPU
- FPGA
- Dense Computation

Protocols:
- CXL.io
- CXL.cache
- CXL.memory

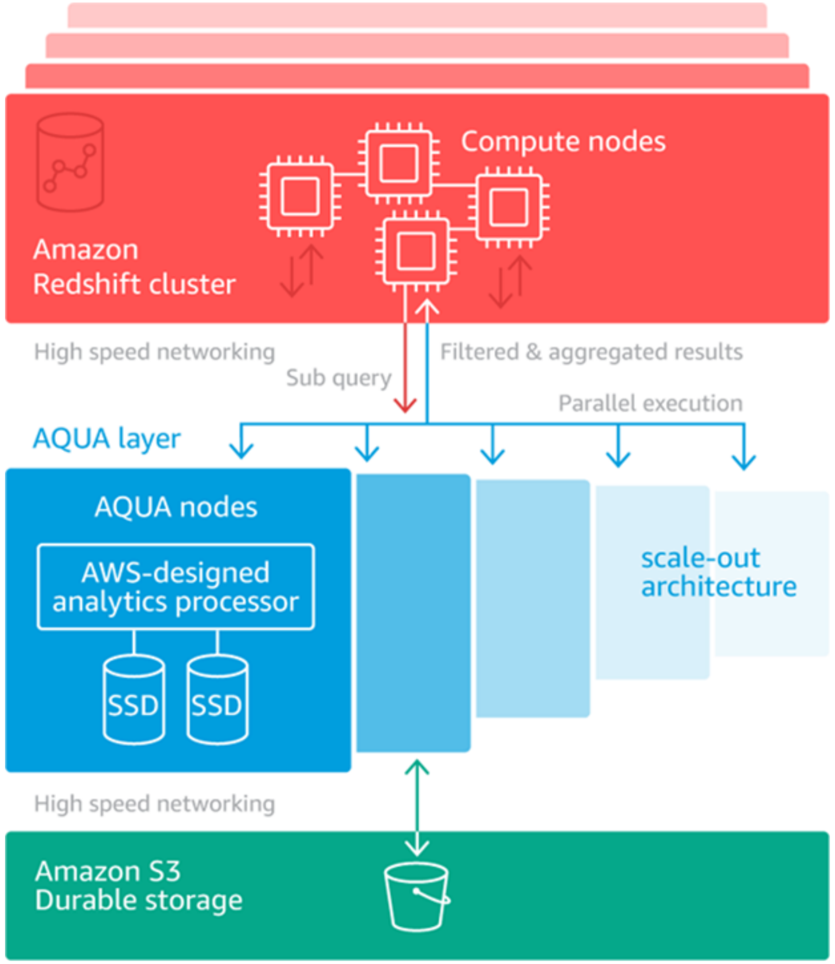
## Memory Buffers

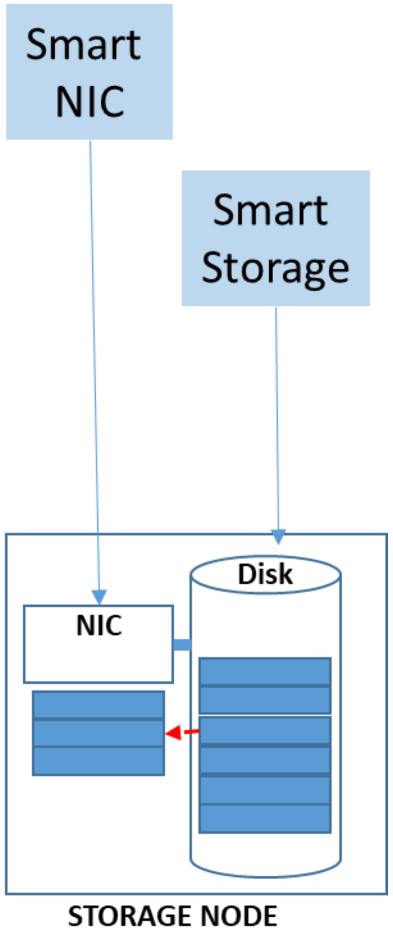
Usages:
- Memory BW expansion
- Memory capacity expansion
- 2LM

Protocols:
- CXL.io
- CXL.mem



# Smart storage (Amazon AQUA)



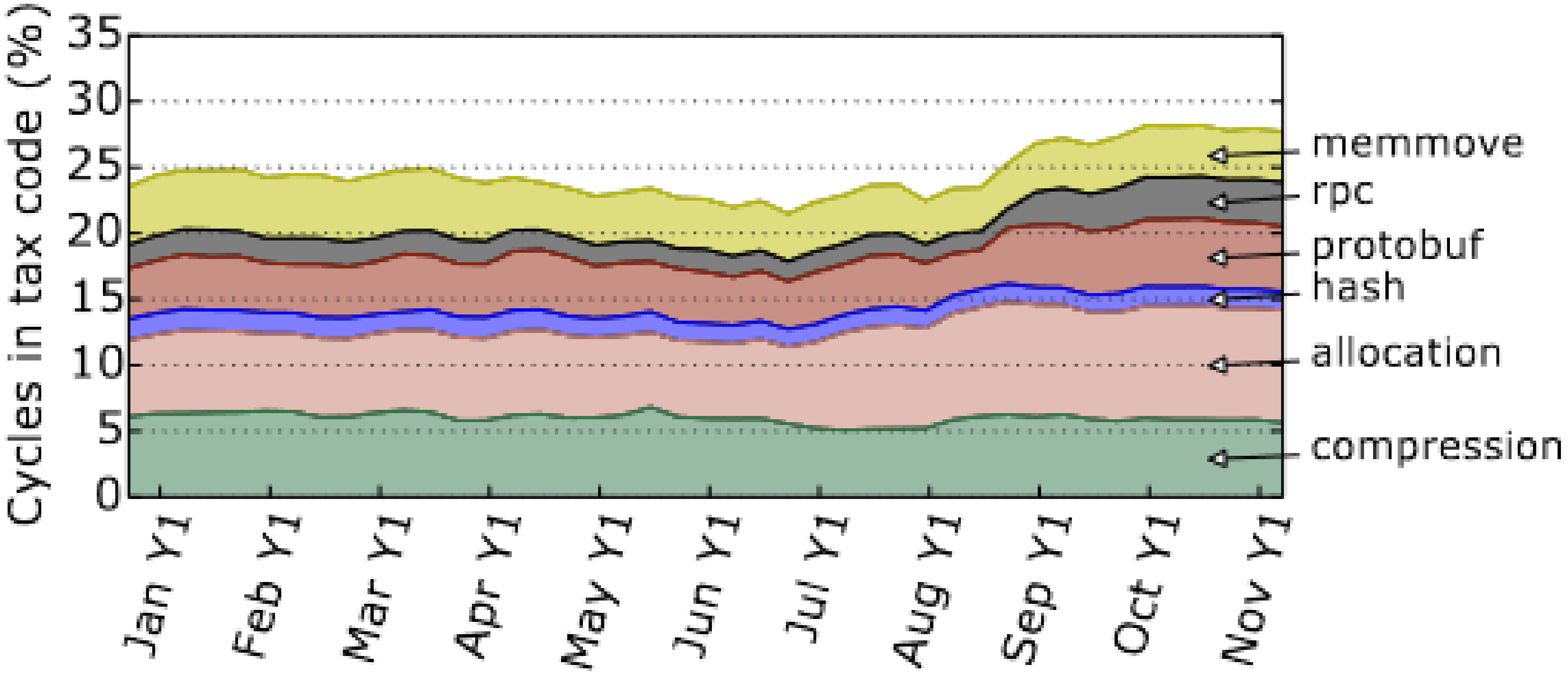


# Smart storage (SSD + FPGA)

https://semiconductor.samsung.com/ssd/smart-ssd/

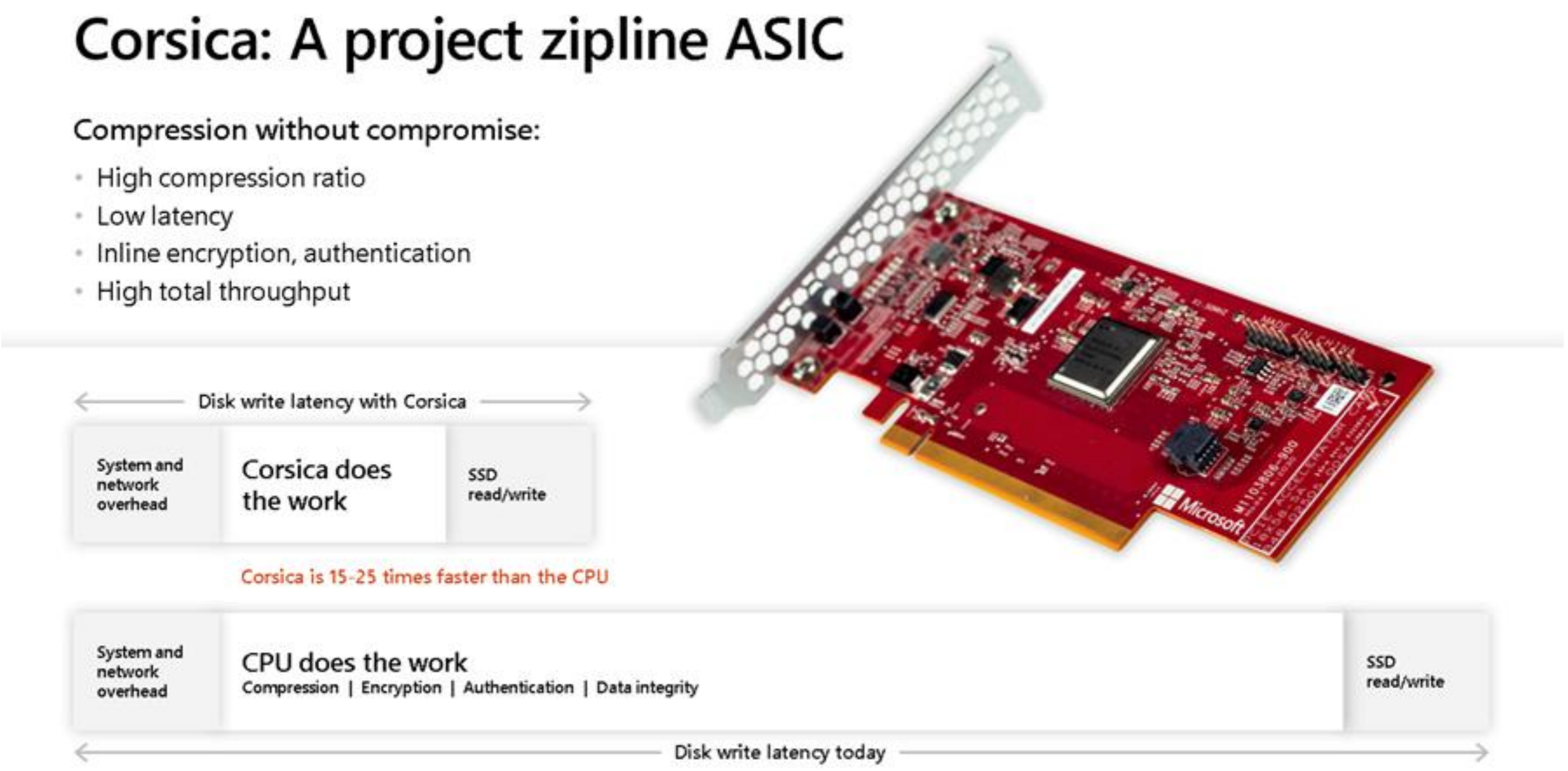
# The motivation

# The Data Center Tax



Profiling a warehouse-scale computer, ISCA 2015

# Data Compression (Microsoft Zipline/Corsica)



## Corsica: A project zipline ASIC

Compression without compromise:

- High compression ratio
- Low latency
- Inline encryption, authentication
- High total throughput

Disk write latency with Corsica

System and network overhead | Corsica does the work | SSD read/write

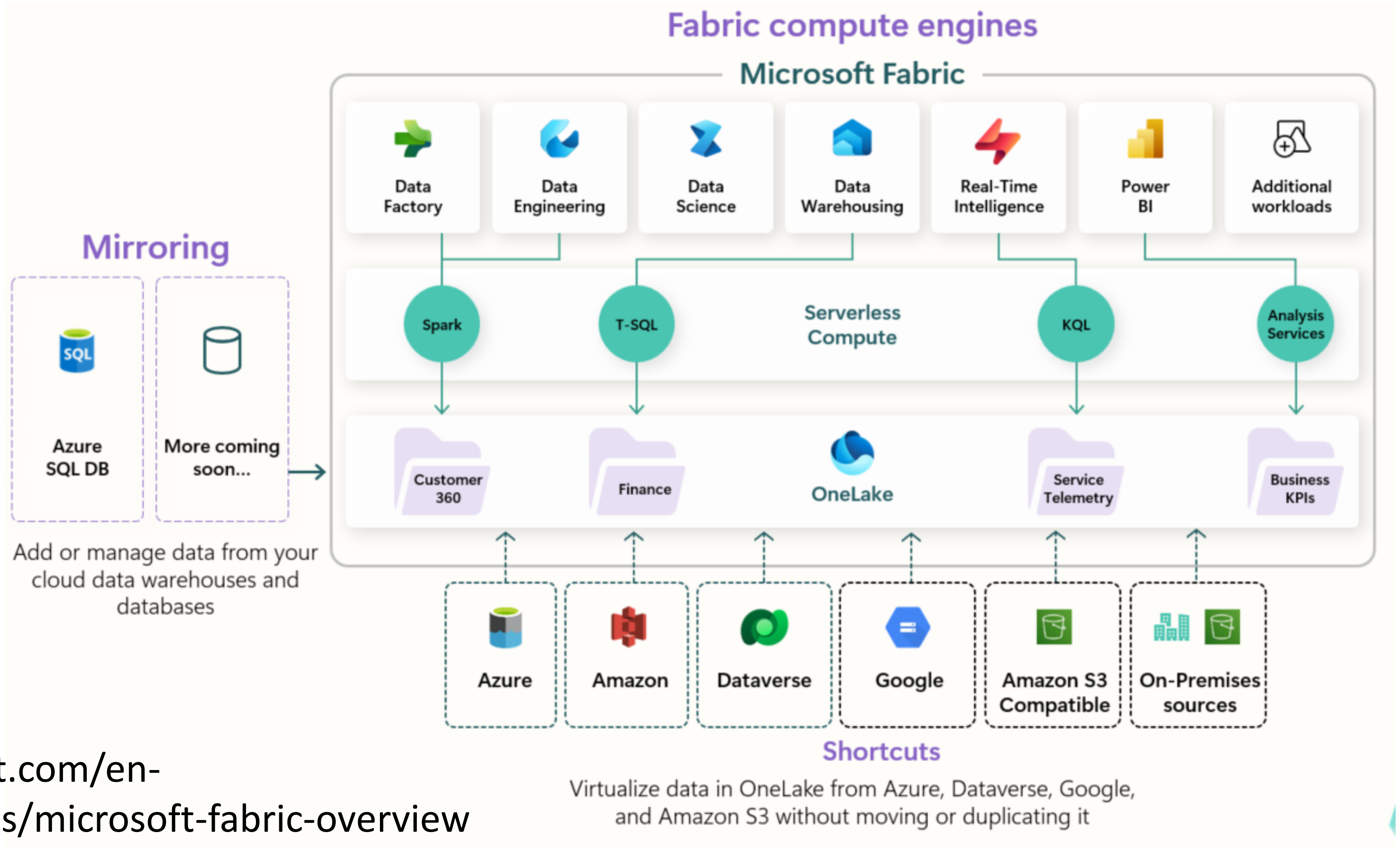
Corsica is 15-25 times faster than the CPU

System and network overhead | CPU does the work<br>Compression | Encryption | Authentication | Data integrity | SSD read/write

Disk write latency today

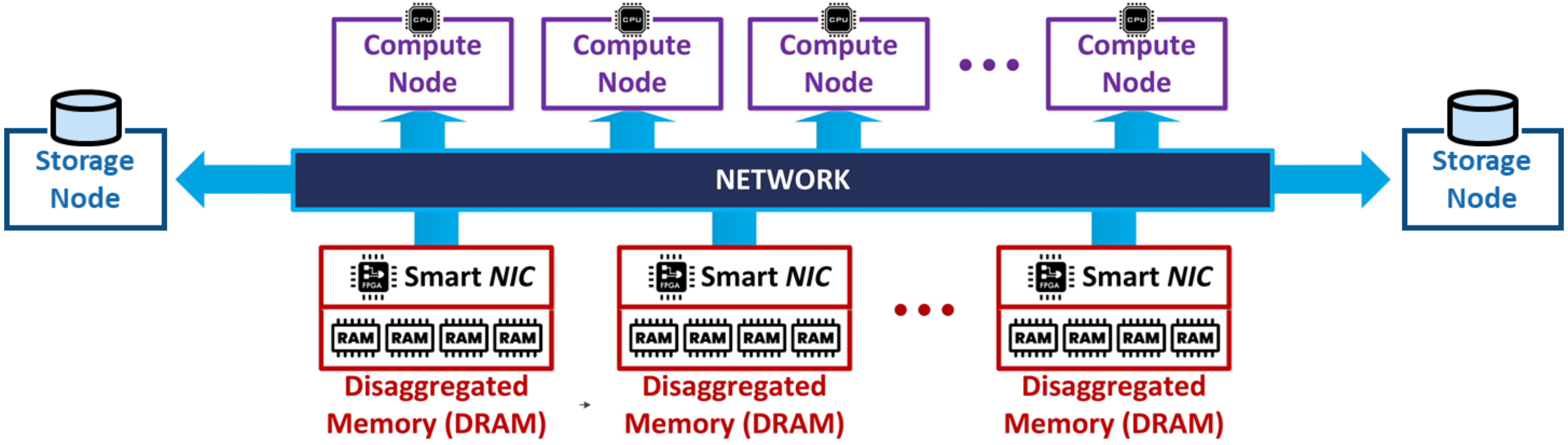
https://azure.microsoft.com/en-us/blog/improved-cloud-service-performance-through-asic-acceleration/

# Very large scale cloud data processing



https://learn.microsoft.com/en-us/fabric/fundamentals/microsoft-fabric-overview

# Reducing data movement (Farview)

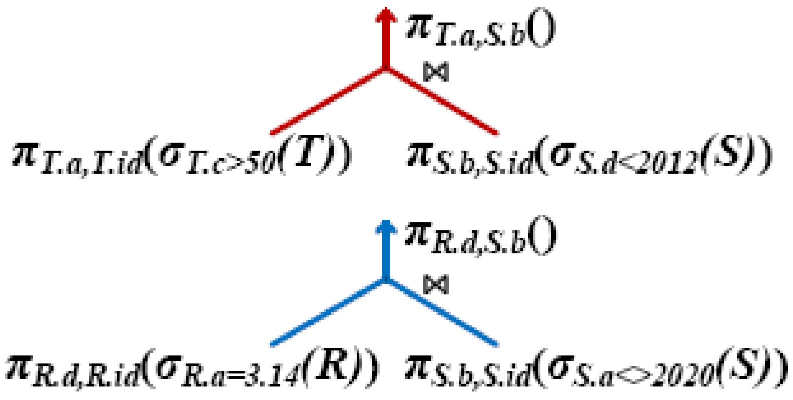


Korolija et al. *Farview: Disaggregated Memory with Operator Off-loading for Database Engines*, CIDR 2022
Work done in collaboration with HPE

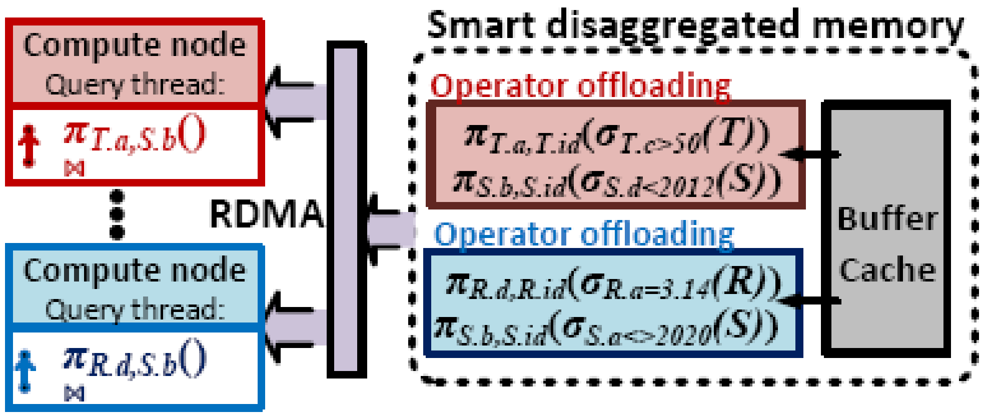
# Smart Disaggregated Memory (Farview)

```
SELECT T.a, S.b
FROM T, S
WHERE T.id = S.id
AND T.c > 50 AND S.d < 2012;

SELECT R.d, S.b
FROM R, S
WHERE R.id = S.id
AND R.a = 3.14 AND S.a <> 2012;
```



$\pi_{T.a,S.b}()$
$\bowtie$
$\pi_{T.a,T.id}(\sigma_{T.c>50}(T))$
$\pi_{S.b,S.id}(\sigma_{S.d<2012}(S))$
$\pi_{R.d,S.b}()$
$\bowtie$
$\pi_{R.d,R.id}(\sigma_{R.a=3.14}(R))$
$\pi_{S.b,S.id}(\sigma_{S.a<>2020}(S))$

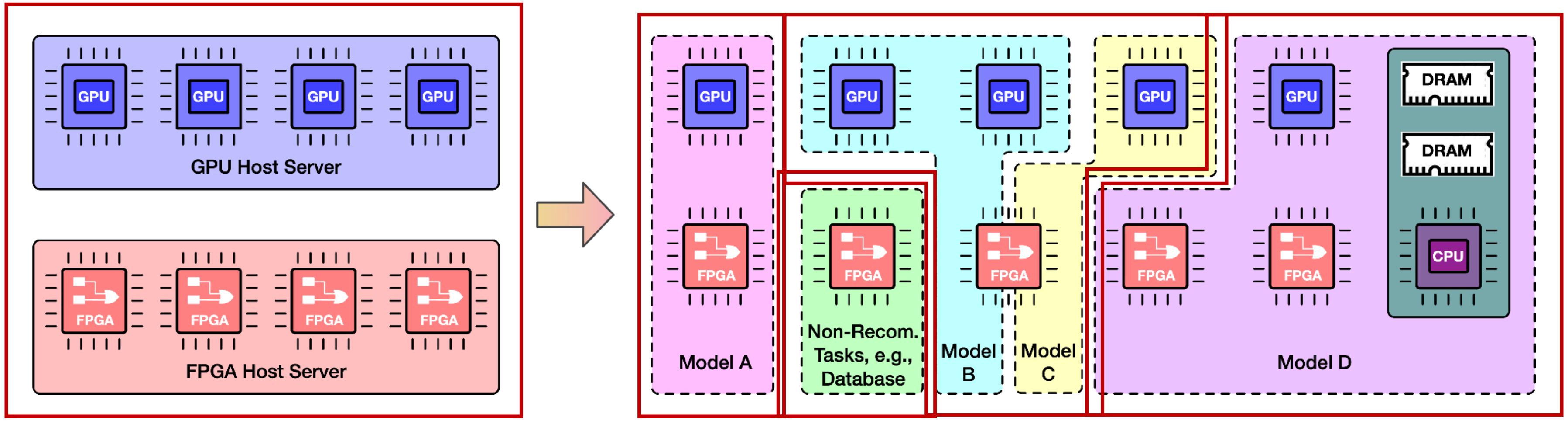


Compute node
Query thread:
$\pi_{T.a,S.b}()$
$\bowtie$
Compute node
Query thread:
$\pi_{R.d,S.b}()$
$\bowtie$
RDMA
Smart disaggregated memory
Operator offloading
$\pi_{T.a,T.id}(\sigma_{T.c>50}(T))$
$\pi_{S.b,S.id}(\sigma_{S.d<2012}(S))$
Operator offloading
$\pi_{R.d,R.id}(\sigma_{R.a=3.14}(R))$
$\pi_{S.b,S.id}(\sigma_{S.a<>2020}(S))$
Buffer Cache

# FleetRec: bridging CPUs, GPUs and FPGAs

- Using existing server

Flexible combination



Interconnect through network

Wenqi Jiang, Zhenhao He, Shuai Zhang, Kai Zeng, Liang Feng, Jiansong Zhang, Tongxuan Liu, Yong Li, Jingren Zhou, Ce Zhang, Gustavo Alonso: FleetRec: Large-Scale Recommendation Inference on Hybrid GPU-FPGA Clusters. KDD 2021

# Vector search acceleration

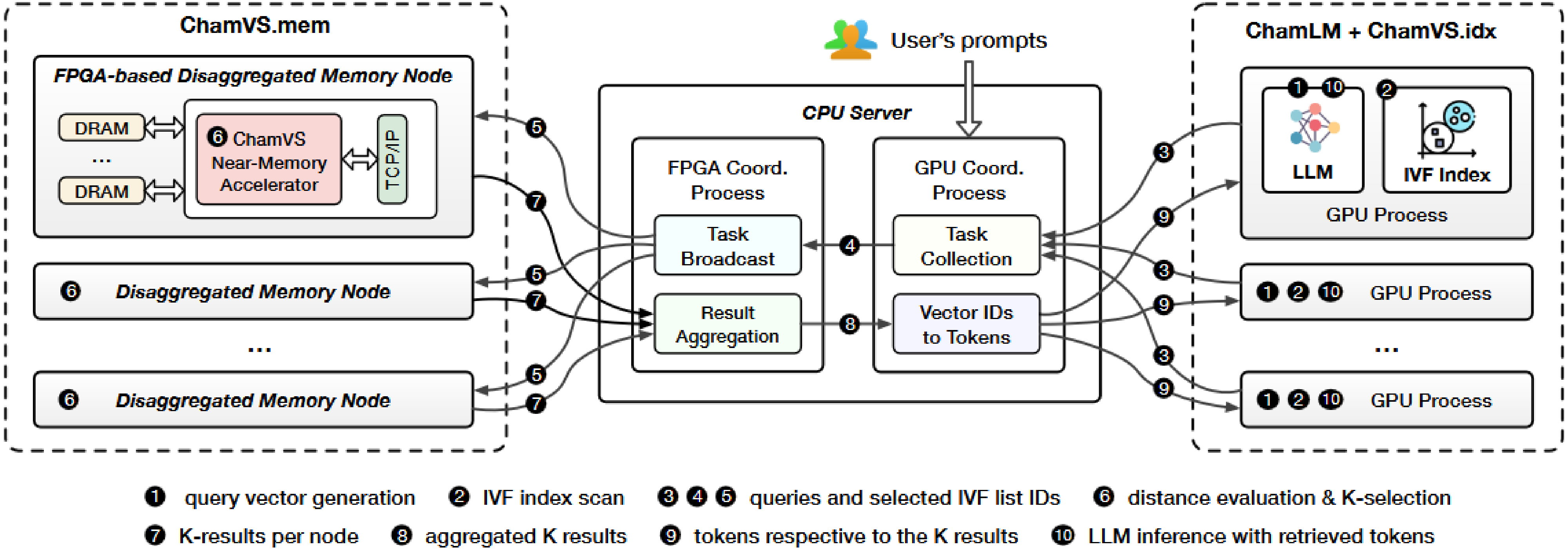


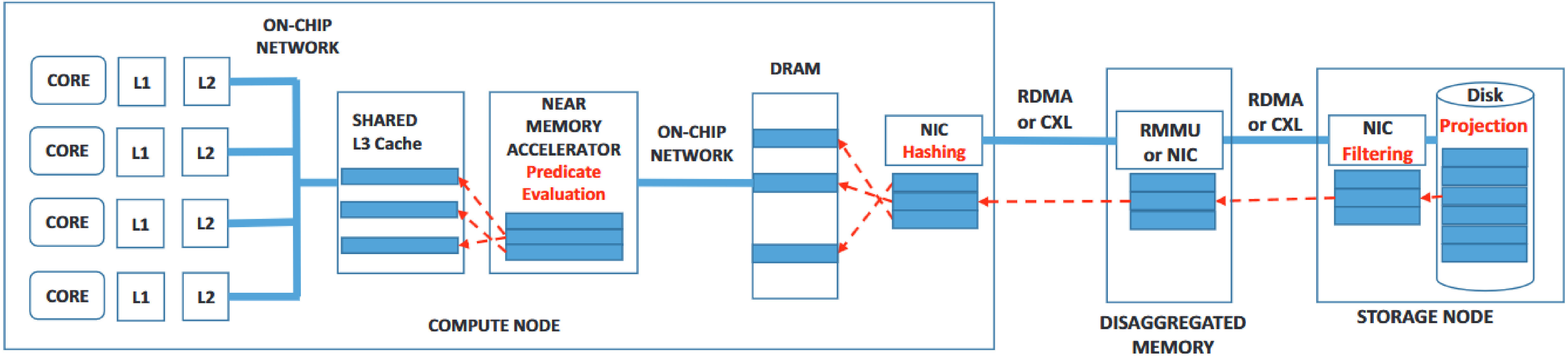
**Figure 3: Chameleon is a heterogeneous and disaggregated accelerator system for efficient RALM inference.**

Chameleon: a Heterogeneous and Disaggregated Accelerator System for Retrieval-Augmented Language Models. Wenqi Jiang et al. VLDB 2025

# Key message

If the data moves, it has to be processed along the data path

# A database example



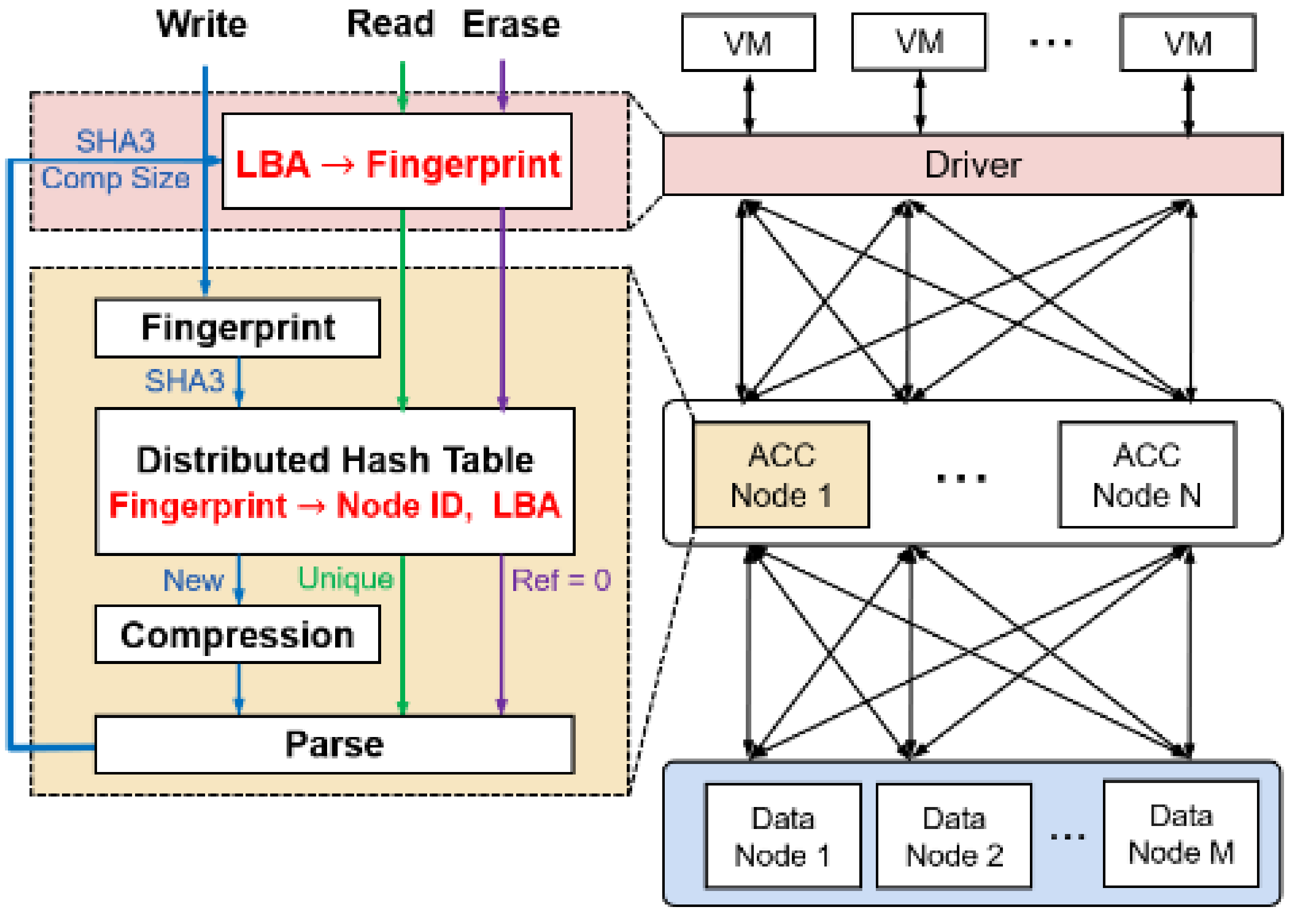
Data Flow Architectures for Data Processing on Modern Hardware. Lerner and Alonso, ICDE 2024

# The research agenda

- What is the most suitable execution model?
  - Streaming?
- What is the interface to computational storage?
- How much compute should move to storage or the data path?
- What processing fits better where?
  - Storage, network, memory, interconnects
- Which operators can be moved to the pipeline?
  - Relational, statistical, sampling, summarization, compression, encryption …
- What are the end-to-end effects and performance?
- How to orchestrate query execution on such a pipeline?

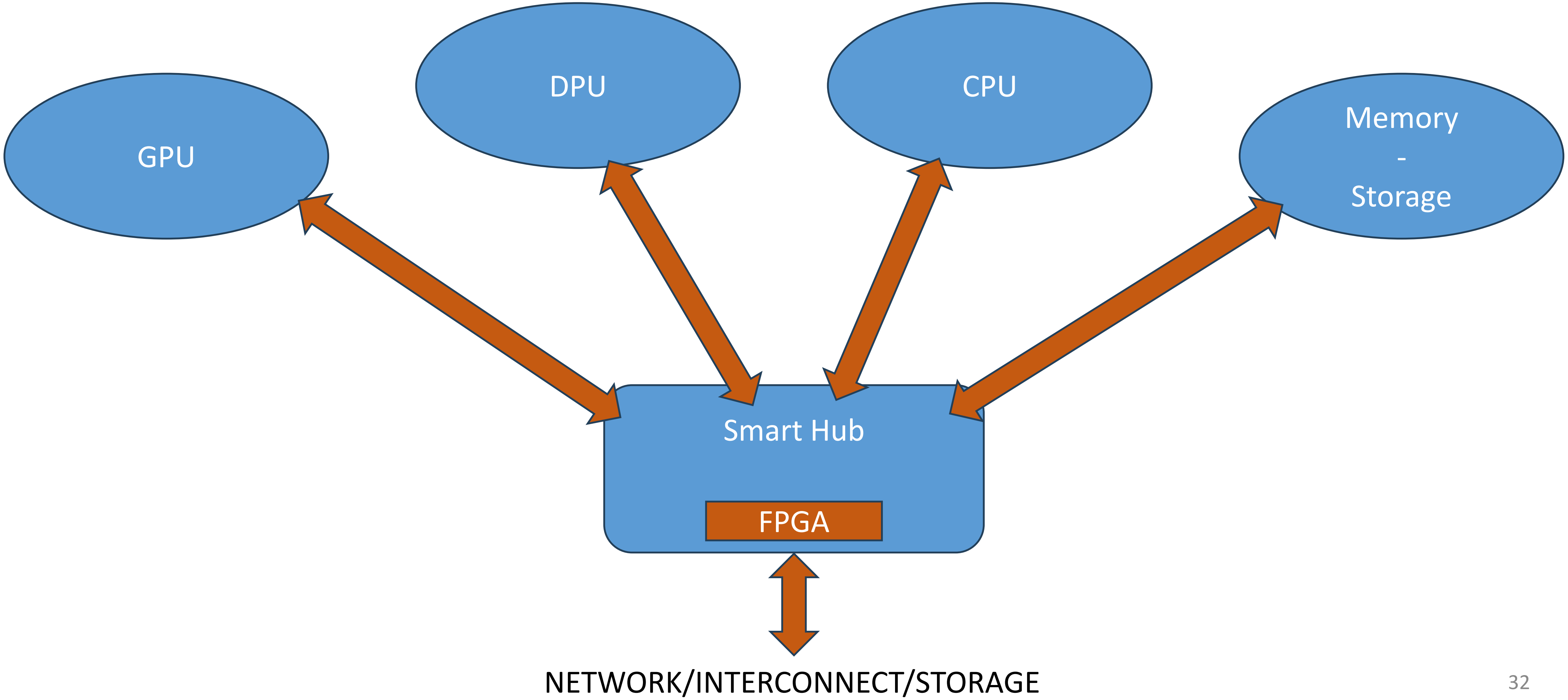
# Also to improve storage

- StreamDeDup
  - Deduplication for the cloud
  - Transparent intermediate layer implemented through in-network FPGAs
  - Deduplicates pages at large scales without involving the CPU or the storage layer



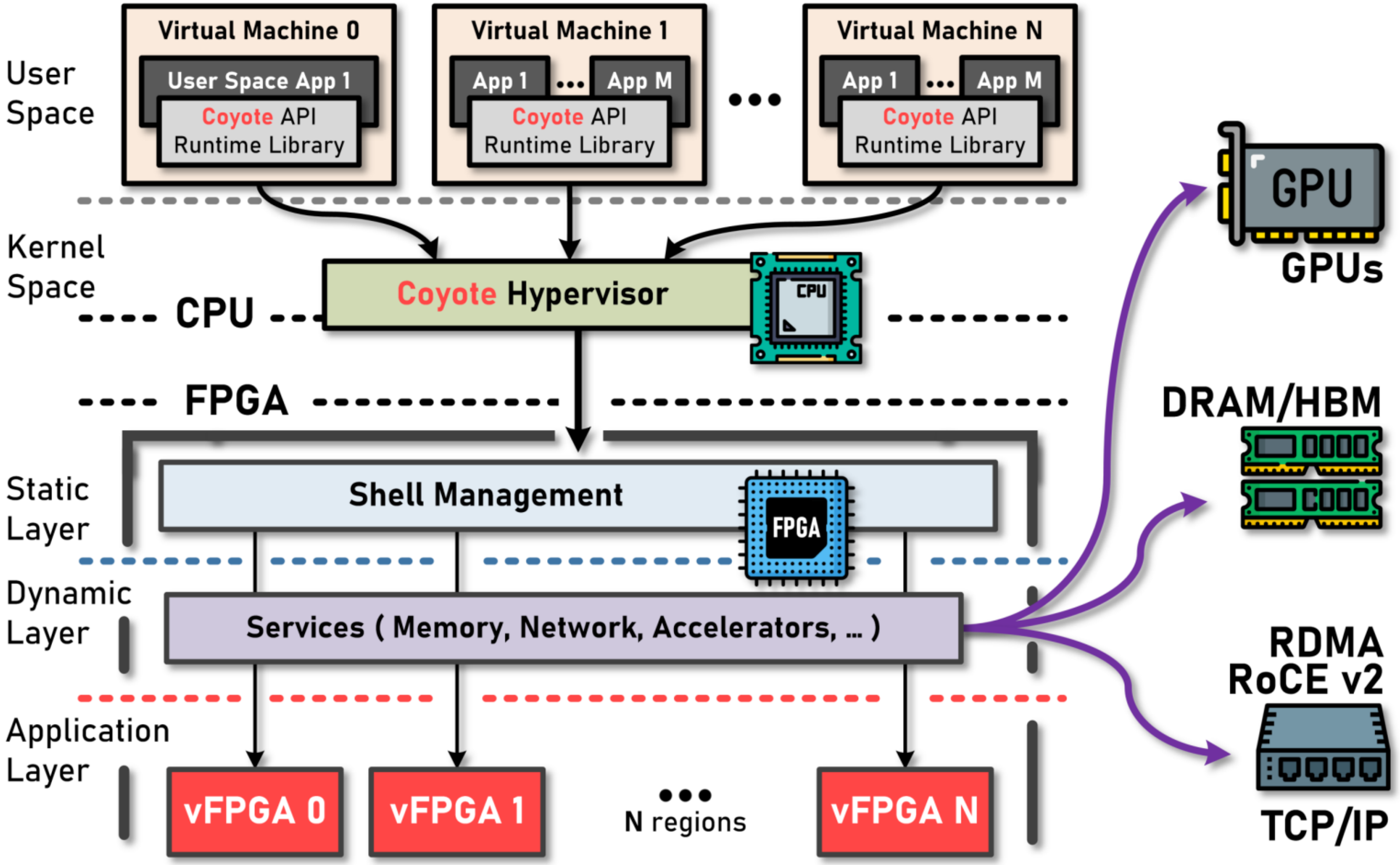
# How to get there

# SLASH (joint work with AMD Research)

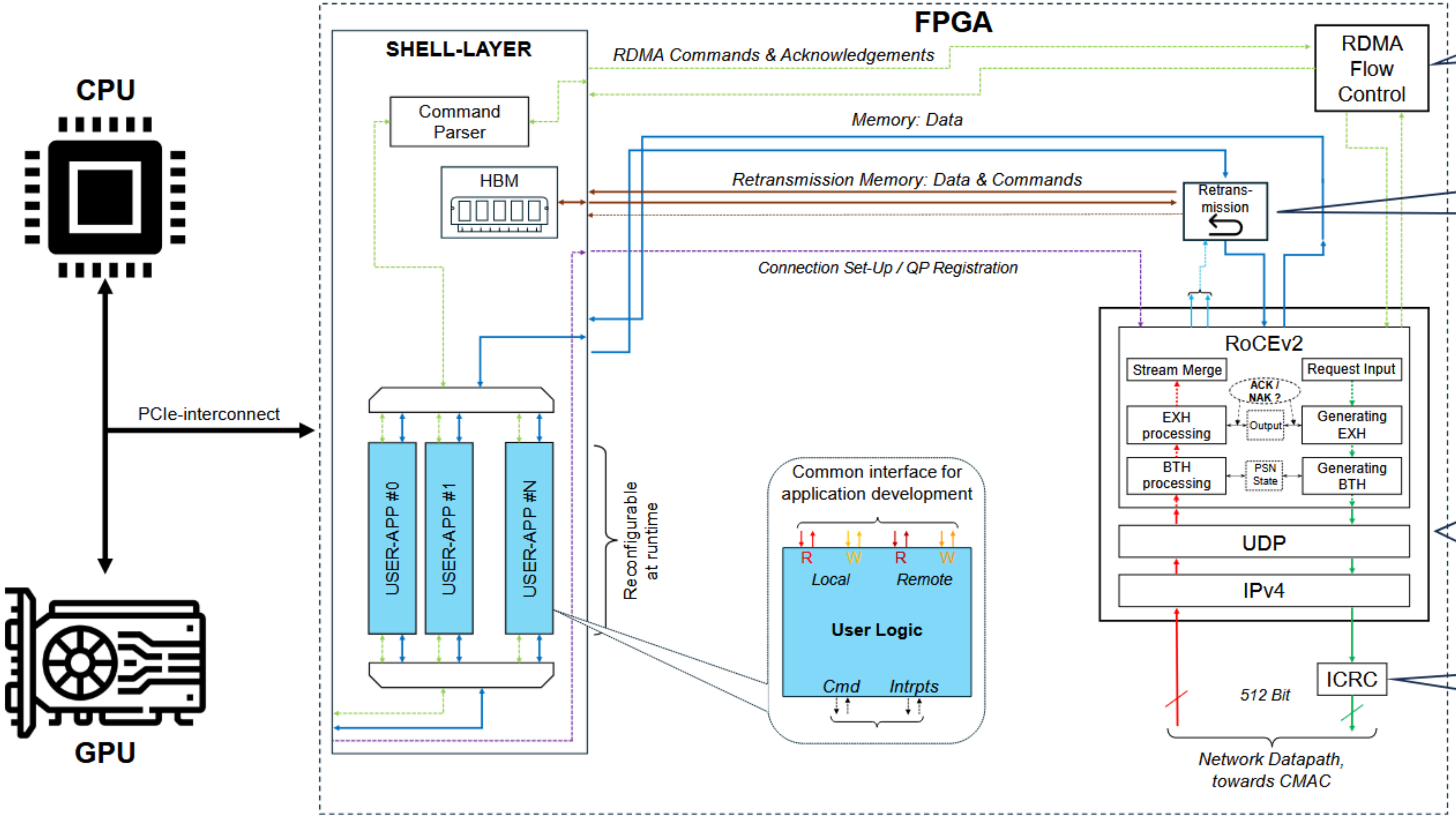


# Prototyping the required hardware

- Coyote v2
- An operating system for FPGAs
    - 2 reconfigurable regions (application and services)
    - Unified memory
    - Access to network and storage



# Network stack (Balboa)



**Flow-control** based on commands + acknowledgements

Buffering of all outgoing payloads in **baremetal HBM-channels**

Main **packet processing pipeline** written in Vitis-HLS

- State-keeping per QPN
- Layered header-processing for IP, UDP and IBV
- Interaction purely via stream-interfaces

Deeply pipelined **checksum calculation**, performance-critical

# Conclusions

- Data shipping is just too expensive
  - Too much data
  - Too much overhead on the system stack
  - Energy and resource inefficient
- Near data processing at all levels of the hardware stack
- Not everything has to/should be done by the CPU
- Starting with Storage
  - Reduce data movement
  - Improve processing efficiency
  - Specialize