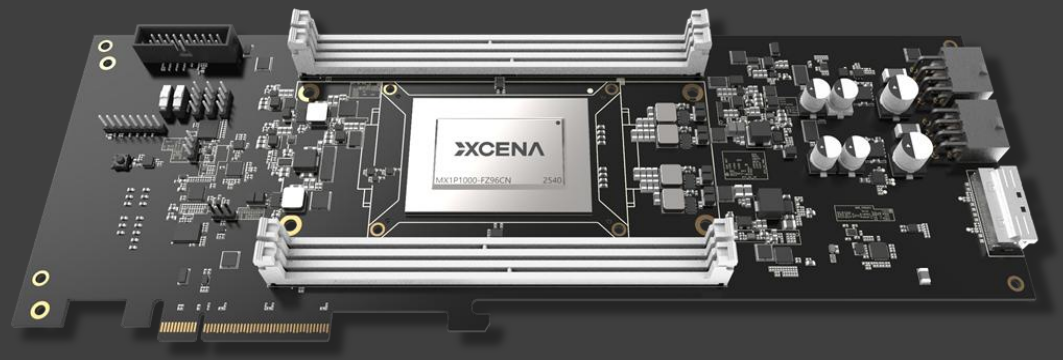
AI+DATA, CXL Memory and Memory Centric Computing

Harry Juhyun Kim

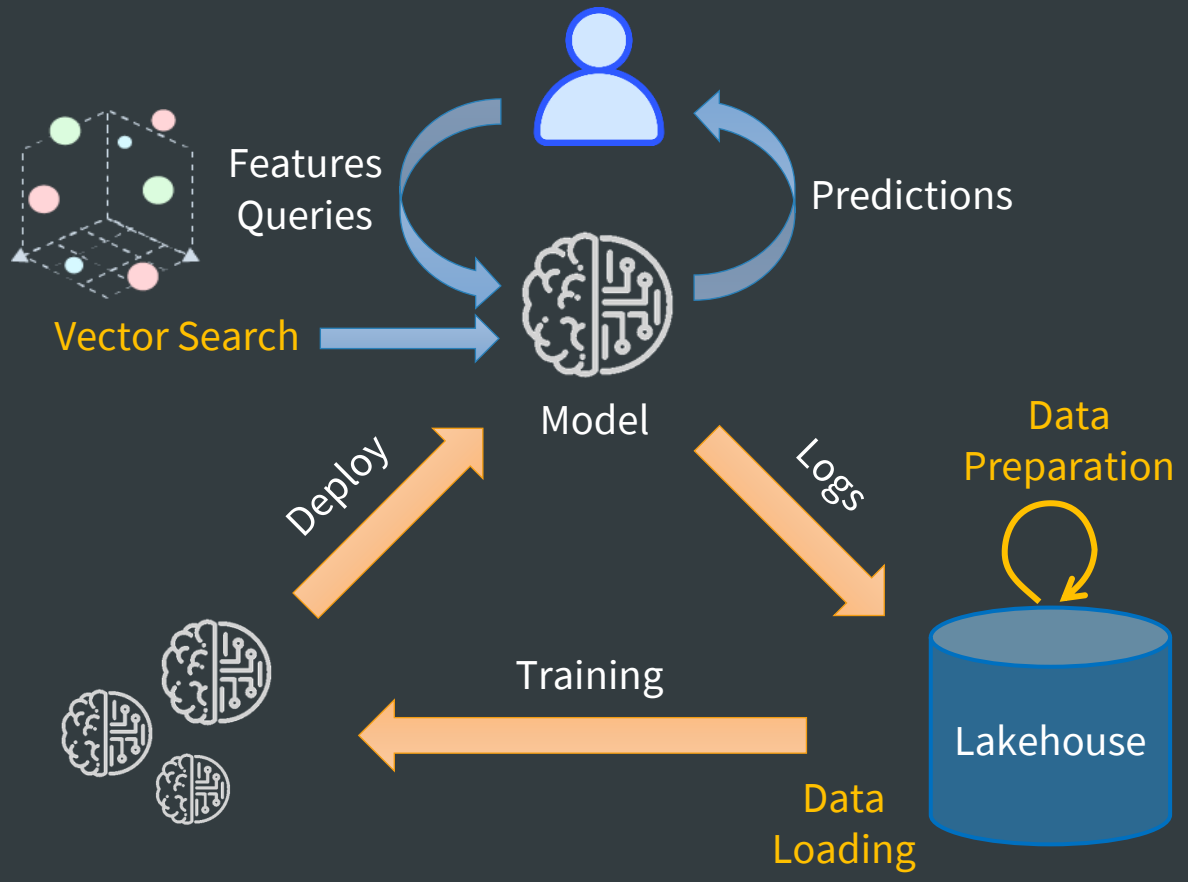
CPO and Co-founder

XCENA



AI+DATA WORLFLOWS

AI/ML IS THE BIGGEST CONSUMER OF DATA TODAY
THE LARGEST TABLES AND LAKEHOUSES ARE BUILT FOR AI



Data Preparation

- Preparing tables
- Embeddings, long sequences of features
- Batch ETL pipelines similar to traditional data transformation
- But extremely inefficient
- Need to understand characteristic of “feature engineering”

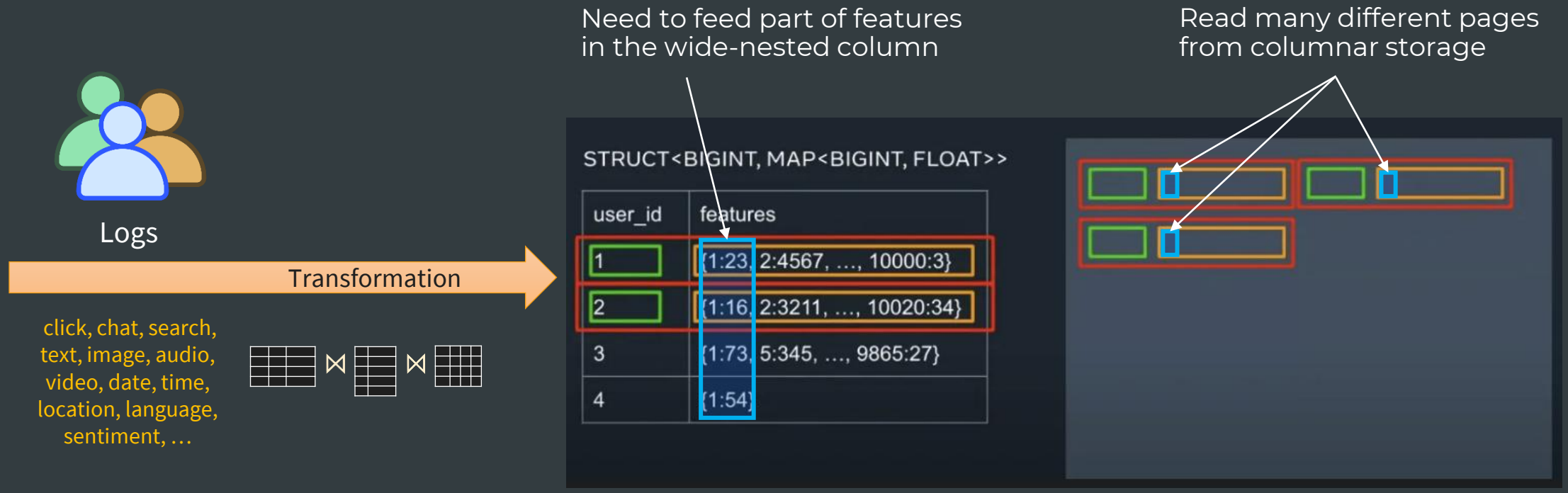
Data Loading

- Feeding GPUs
- Latency sensitive
- Order sensitive (Training)
  → random/chronological/...
- New challenges for data processing

Source: Veloxcon 2025 https://youtu.be/nk9abk0evLk

WHAT AI DATA LOOKS LIKE

Features are frequently added and removed in a wide column with nested schema.
Transformation, feature engineering, storage and retrieval are all challenging.



Source: Nimble File Format

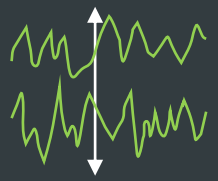
CHALLENGES IN DATA ANALYTICS

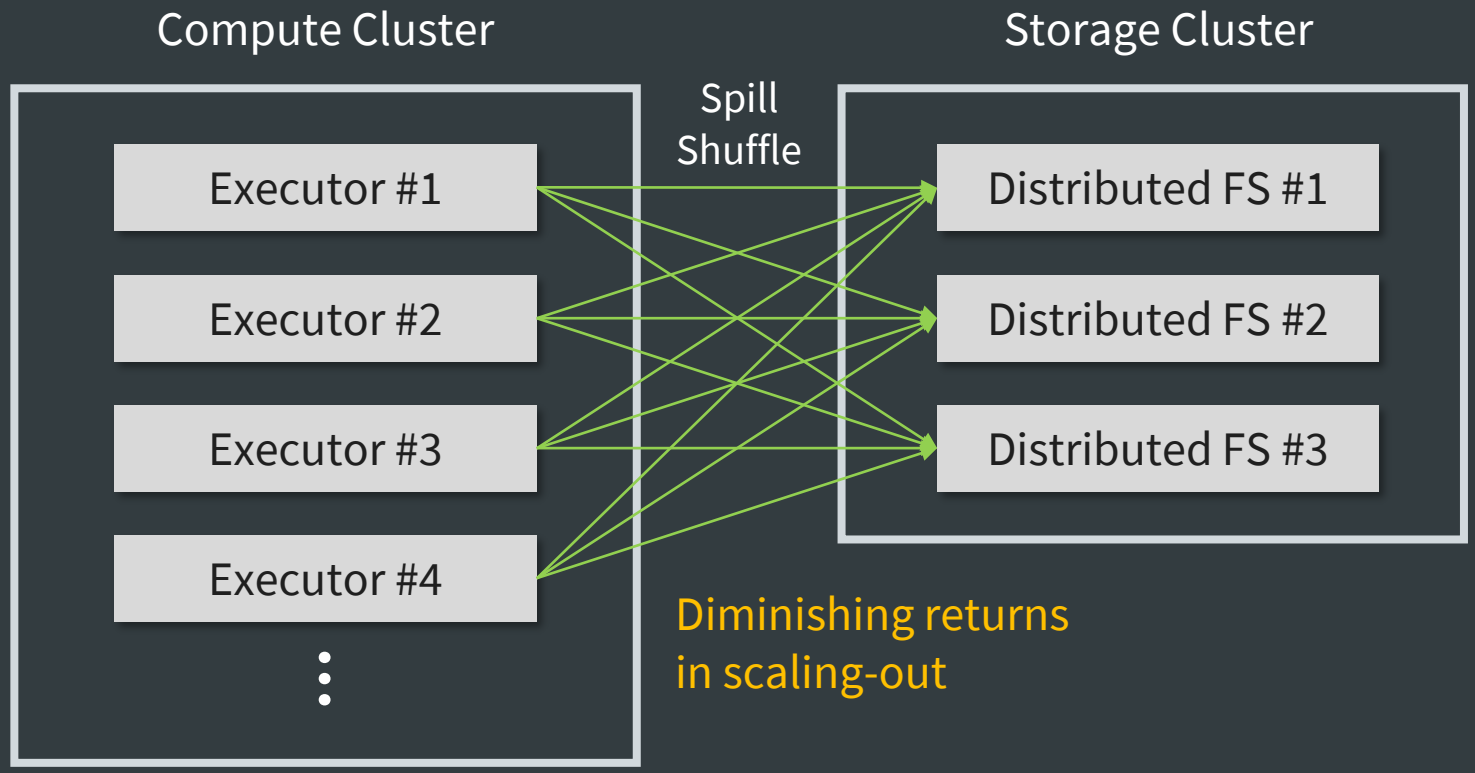
XCENA

MEMORY INEFFICIENCIES AND DIMINISHING RETURNS IN SCALING CALL FOR A NEW MEMORY SOLUTION

Out Of Memory kill
Disk spill

Huge variance in workloads memory requirements leads to over-provisioning

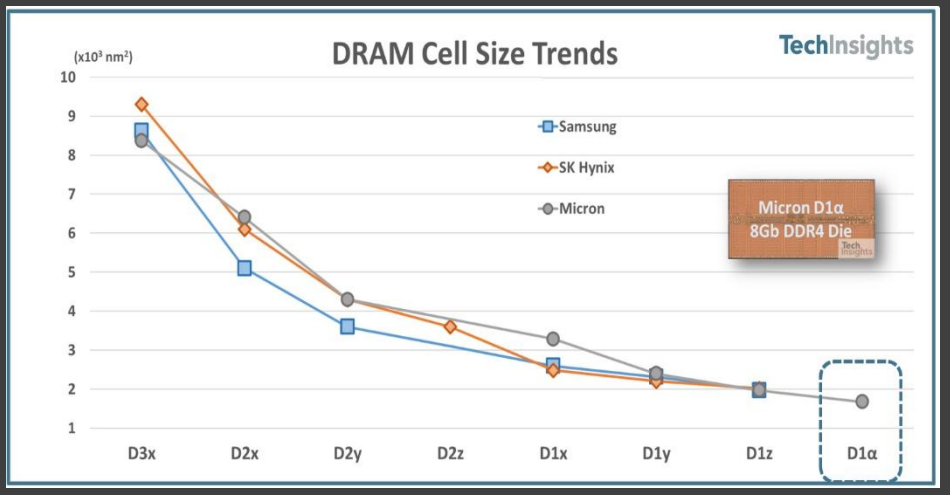




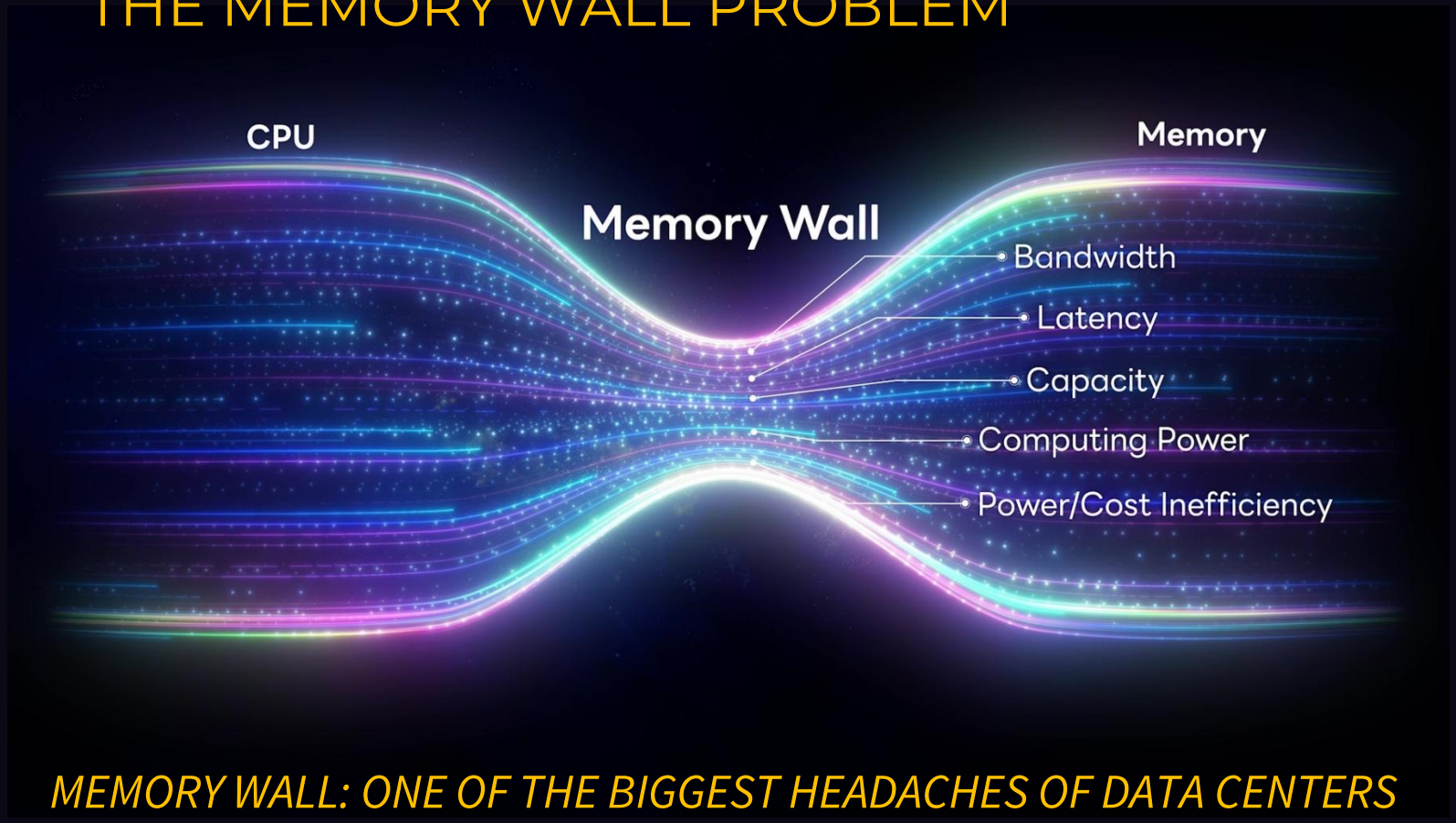
Diminishing returns in scaling-out

PROBLEM

DATA IS RAPIDLY INCREASING,
WHILE
MEMORY TECHNOLOGY
STRUGGLES TO KEEP UP



XCENA AIMS TO BREAK THE MEMORY WALL PROBLEM



MEMORY WALL: ONE OF THE BIGGEST HEADACHES OF DATA CENTERS

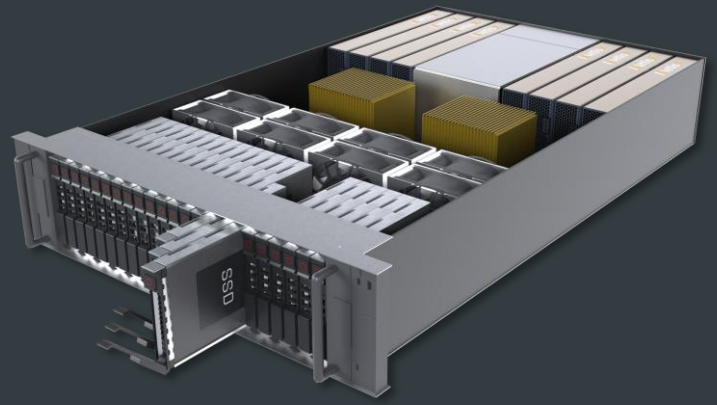
- *SIZE: CHALLENGING TO KEEP UP WITH THE RAPID DATA INCREASE*
- *PERFORMANCE: DATA MOVEMENT IS THE BIGGEST BOTTLENECK*
- *COST: MEMORY IS REALLY EXPENSIVE (HALF OF THE SERVER COSTS)*
- *UTILIZATION: MEMORY UTILIZATION IS VERY LOW (OVER-PROVISIONED)*

SOLUTION?

XCENA

WHAT IF A SINGLE DENSE MEMORY AND COMPUTE NODE COULD REPLACE A CLUSTER OF 10?

Dense Memory and Compute Node



=

Cluster of Nodes

DATA PROCESSING QUADRANT

EMBARRESSINGLY PARALLEL, LOW COMPUTE INTENSITY, MEMORY BOUNDED

Disaggregation, Computational Memory

DDR

CXL MEMORY

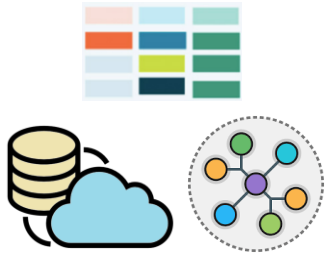
Big Data Processing

- *Vector Databases For Gen AI*
- *Scale-out Databases*
- *Graph Databases*
- *DNA Analysis*

Computational Intensity Per Memory Access

Low → *High*

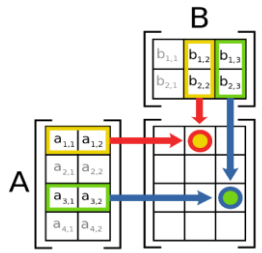
[Large-scale Data Analytics]



[General Computing]

No Interest

[Dense Matrix Multi.]



Operation Diversity Per Memory Access

High

Low

Bandwidth/Capacity Expansion

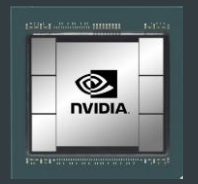
DDR

LPDDR

CXL MEMORY

General Computing

HBM, Explicit Memory



HBM

GDDR

Artificial Intelligence

COMPUTE EXPRESS LINK

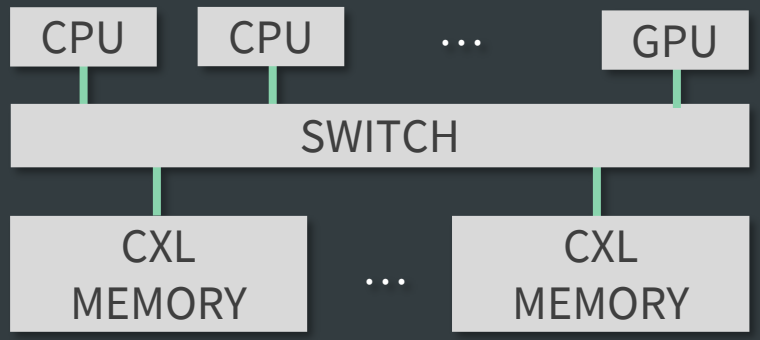
XCENA

THE NEXT GENERATION MEMORY INTERCONNECT PROTOCOL

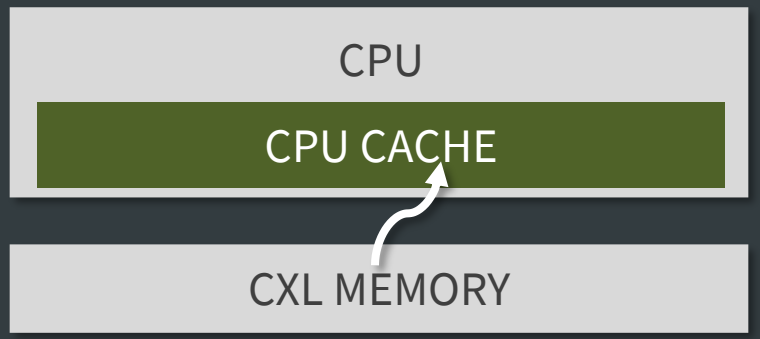
Built on PCIe that lets you plug in memory as the same way you plug in a GPU card

CXL memory looks like regular DRAM with additional latency due to protocol overhead

Memory pooling and sharing through Switch



HIGHLY SCALABLE MEMORY EXPANSION

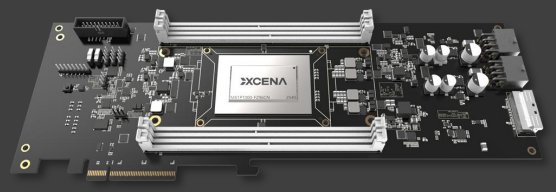


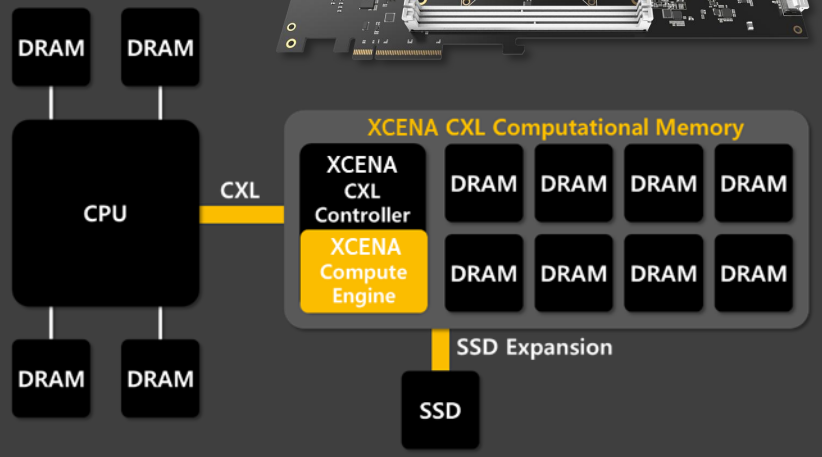
CACHE COHERENCE

HETEROGENEOUS COMPUTING

XCENA

CXL COMPUTATIONAL MEMORY





BEYOND JUST ANOTHER CXL MEMORY EXPANDER

DATA PROCESSING LIBRARY

DATA ANALYTICS ACCELERATION

TCO SAVINGS
SEAMLESS INTEGRATION

COMPUTING HW-SW SOLUTION

NEAR-MEMORY PROCESSING

LESS DATA MOVEMENT
LOWER CPU UTIL
LOWER POWER
BETTER PERFORMANCE

CXL MEMORY

HIGH PERFORMANCE CXL DRAM EXPANSION

GENERAL CXL PLAYERS

INFINITE MEMORY

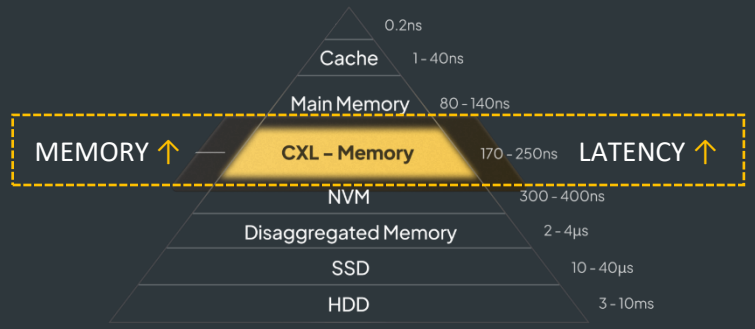
PB-SCALE MEMORY WITH SSD EXPANSION

MORE MEMORY
CHEAPER MEMORY

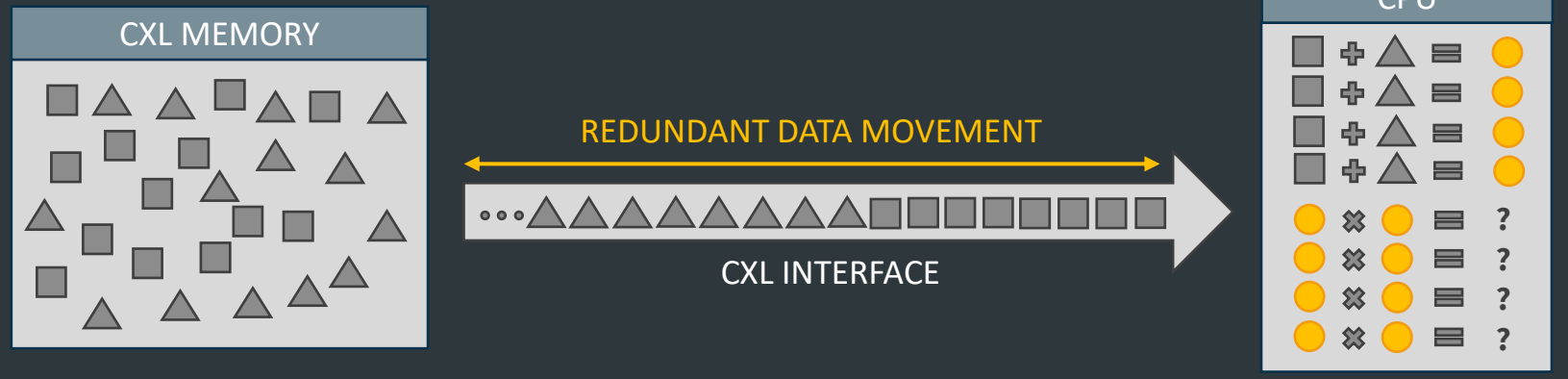
NEAR-MEMORY PROCESSING

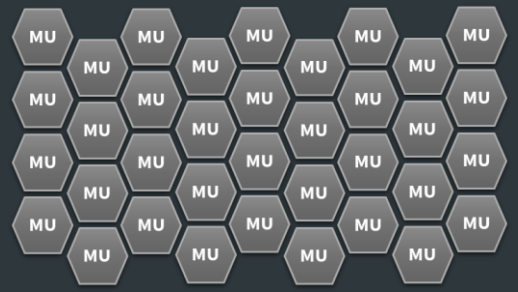
XCENA

More Capacity, Extra Bandwidth, +Latency
Reducing data movement by Near-Memory Processing

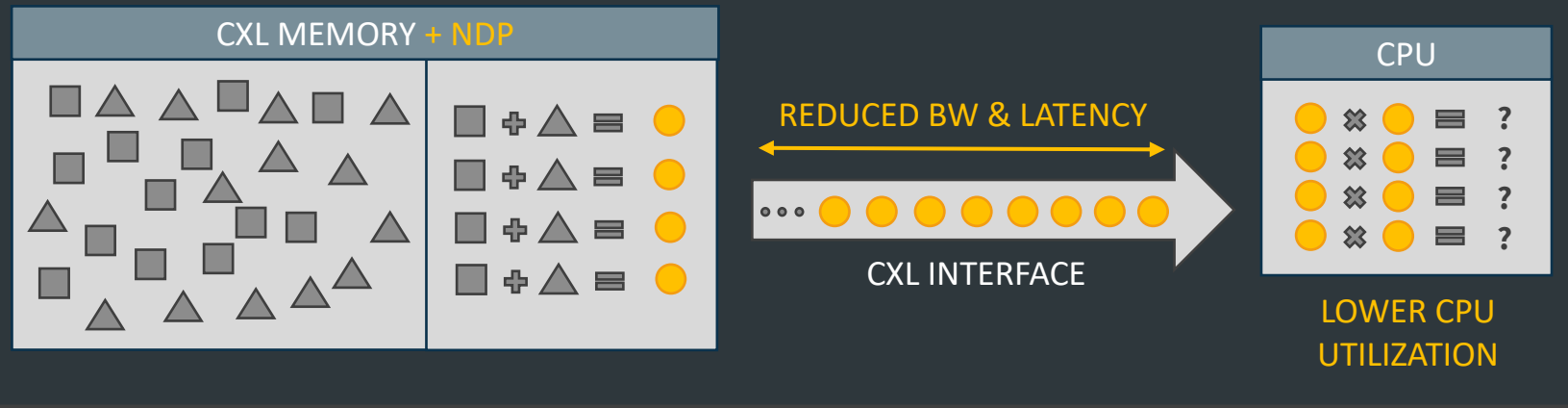


SCALABLE MEMORY
BUT WITH INCREASED LATENCY





1000s RISC-V CORES FOR EFFICIENT DATA PROCESSING

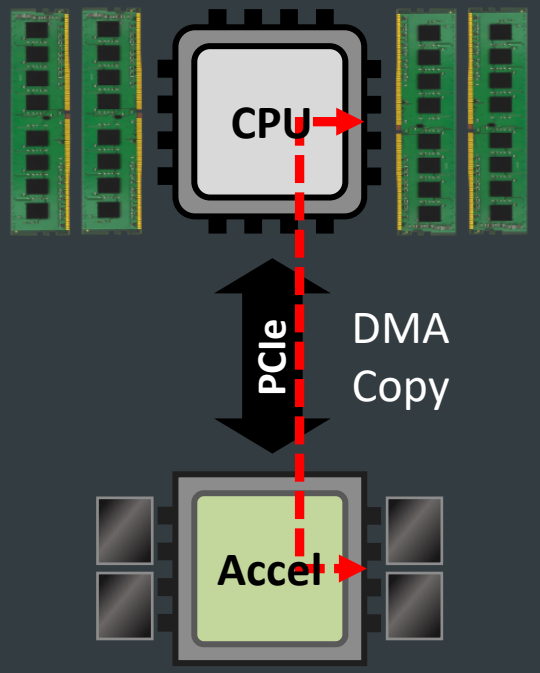


LOWER CPU UTILIZATION

KEY DIFFERENCES
PCIE vs. CXL

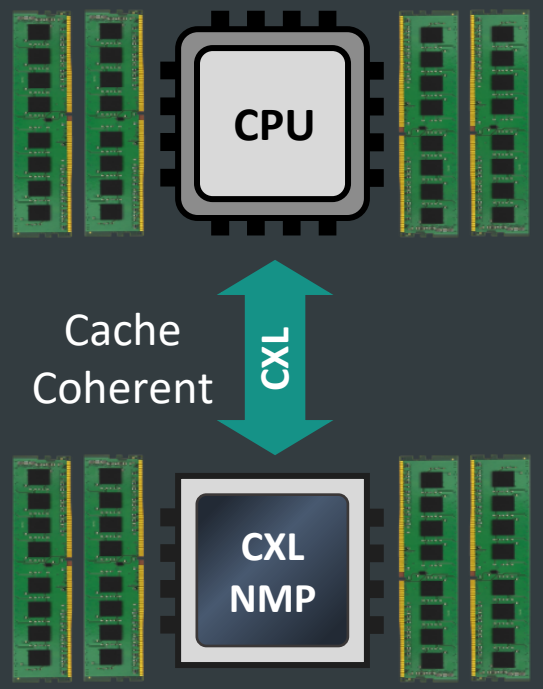
CXL MEMORY IS A PART OF HOST SYSTEM MEMORY,
WHEREAS PCIE DEVICE MEMORY IS NOT VISIBLE TO THE HOST.

Typical PCIe Based Acceleration



- ✓ Write
- ✓ MemCpy (H2D)
- ✓ Compute
- ✓ MemCpy (D2H)
- ✓ Read

CXL Memory Based Acceleration



- ✓ Write (Coherent)
- ✓ Compute
- ✓ Read (Coherent)

← Part of Host Memory regardless of Acceleration

CXL MEMORY + DATA PROCESSING

XCENA

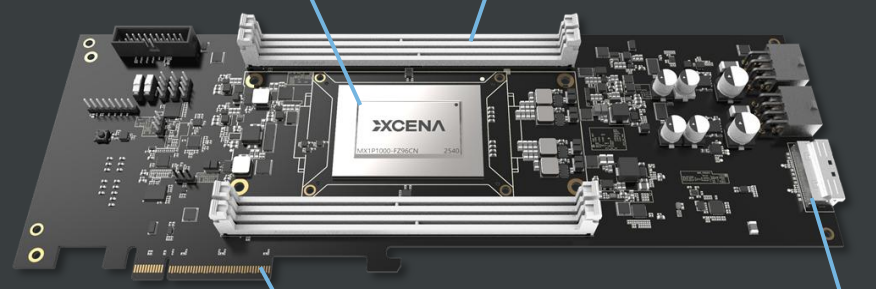
CXL COMPUTATIONAL MEMORY FOR LARGE-SCALE DATA

AVAILABLE SOON

Novel CXL Hardware

1000s of Custom
RISC-V Cores
+
TFLOPS
Vector Engine

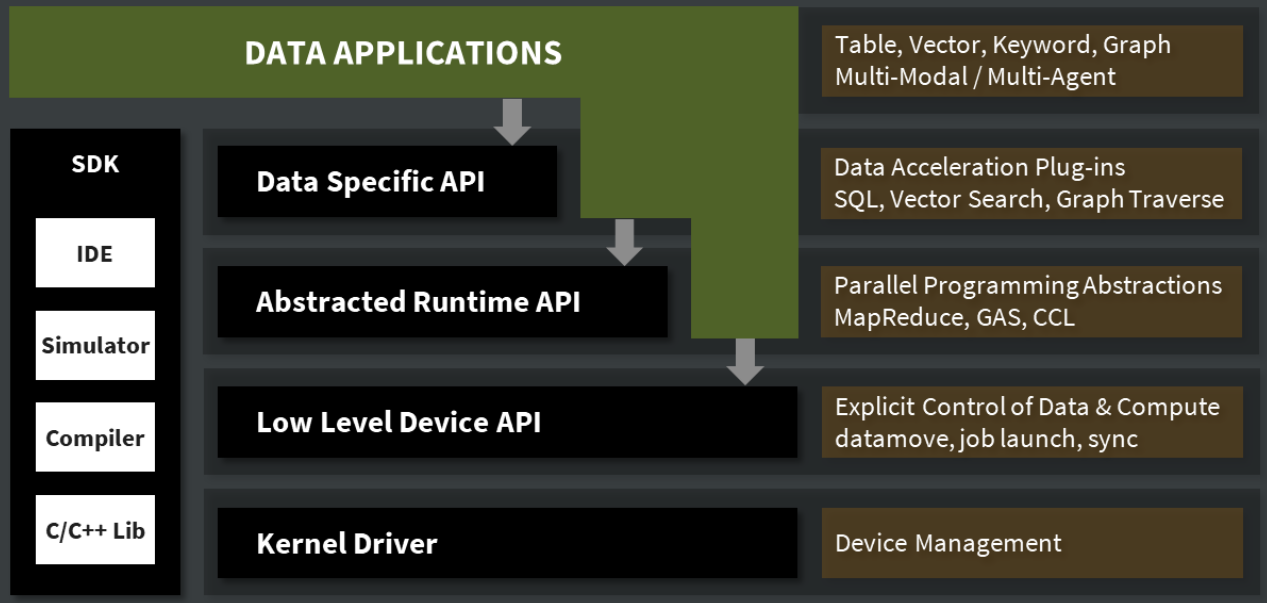
DDR5 x 4Ch
~1TB



CXL 3.0 HDM-DB
with Back-invalidation
Cache Coherence

SSD-backed
CXL Expansion

Rich Software Framework



PARALLEL PROGRAMMING

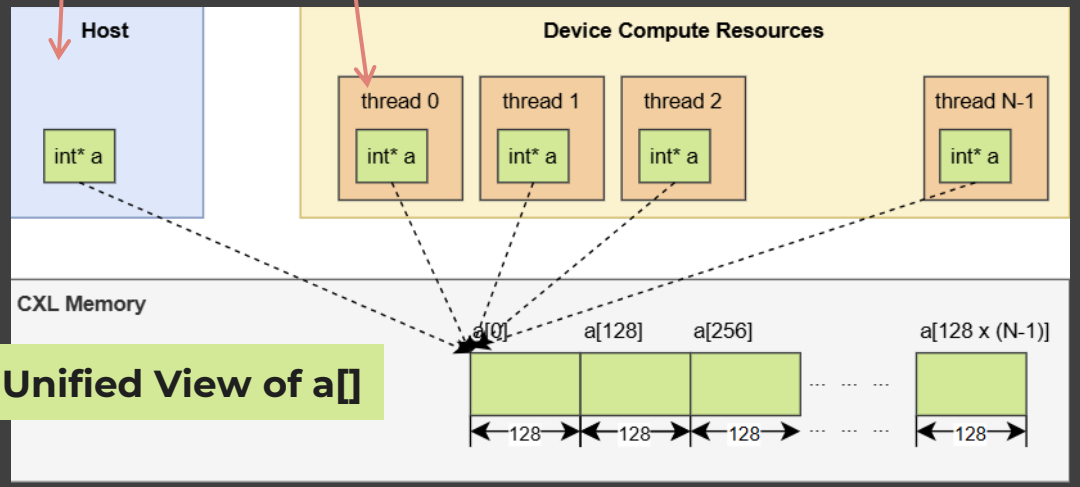
XCENA

Parallel Xceleration Library (PXL)
Map API encapsulates all the complexities of device management and provides a FRAMEWORK of thought for developers.

```cpp
#include "pxl/pxl.hpp"

// setup the device
const char* filename = "mu_kernel/mu_kernel.mubin";
const char* muFuncName = "sort_with_ptr";
auto context = pxl::runtime::createContext(0);
auto module = pxl::createModule(filename);
auto job = context->createJob();
job->load(module);
auto muFunction = module->createFunction(muFuncName);

// allocate cxl memory
int* a = reinterpret_cast<int*>(context->memAlloc(N * 128 * sizeof(int)));
// setup data
for (size_t i = 0; i < N; i++)
{
    for (size_t j = 0; j < 128; j++)
    {
        a[i * 128 + j] = 128 - j;
    }
}

auto map = job->buildMap(muFunction, N);
auto ret = map->execute(a, 128);
map->synchronize();
```

PXL Runtime

```cpp
#include <algorithm>
#include "mu/mu.hpp"

void sort_with_ptr(int* a, int size)
{
    auto taskIdx = mu::getTaskIdx();
    auto curArray = &a[taskIdx * size];
    std::sort(curArray, curArray + size);
}

MU_KERNEL_ADD(sort_with_ptr)
```

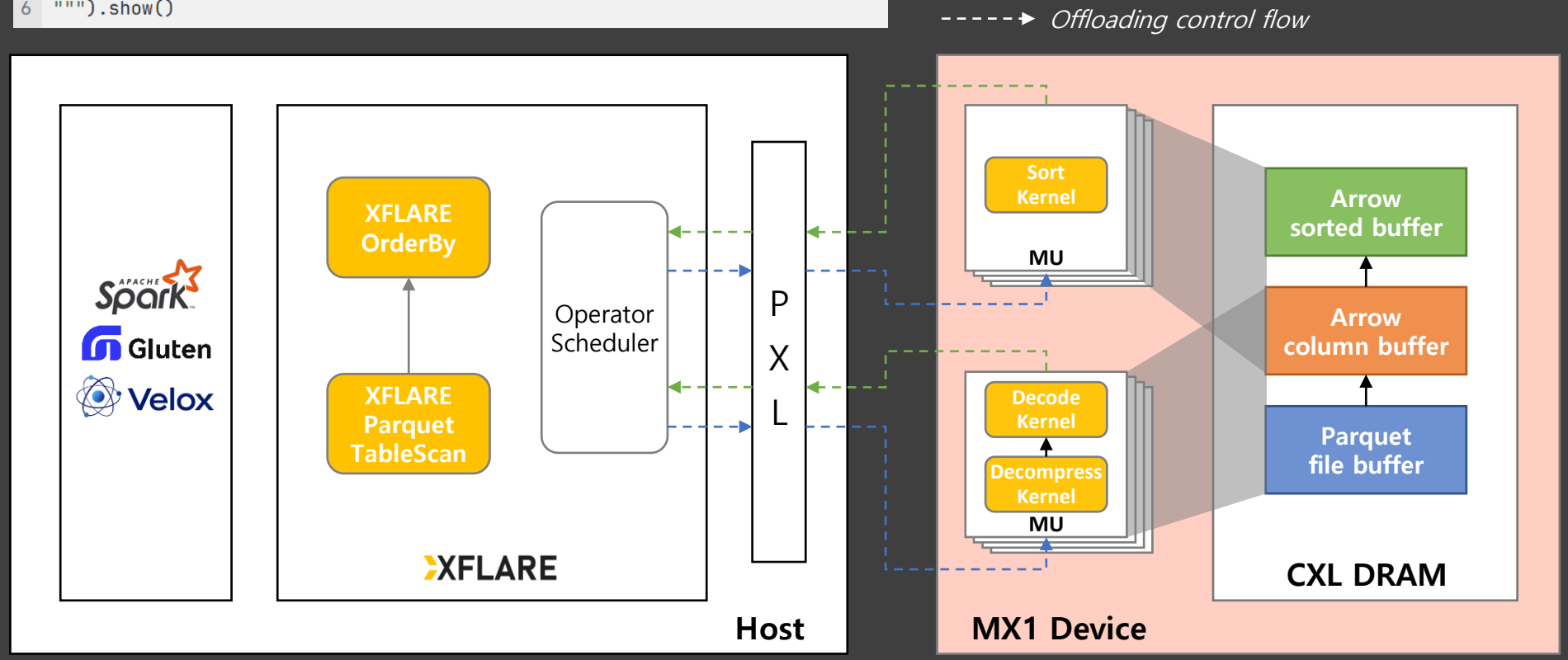


Coherent, Unified View of a[]

DATA SPECIFIC LIBRARY

Spark SQL query is compiled and planned by Spark, translated into Velox plans by Gluten, and executed by XFLARE on MX1.

```
spark.read.parquet("persons.parquet").createOrReplaceTempView("persons")
spark.sql("""
    SELECT   *
    FROM     persons
    ORDER BY age DESC
""").show()
```



DON'T MOVE DATA, MOVE COMPUTATIONS!

CXL LOAD/STORE

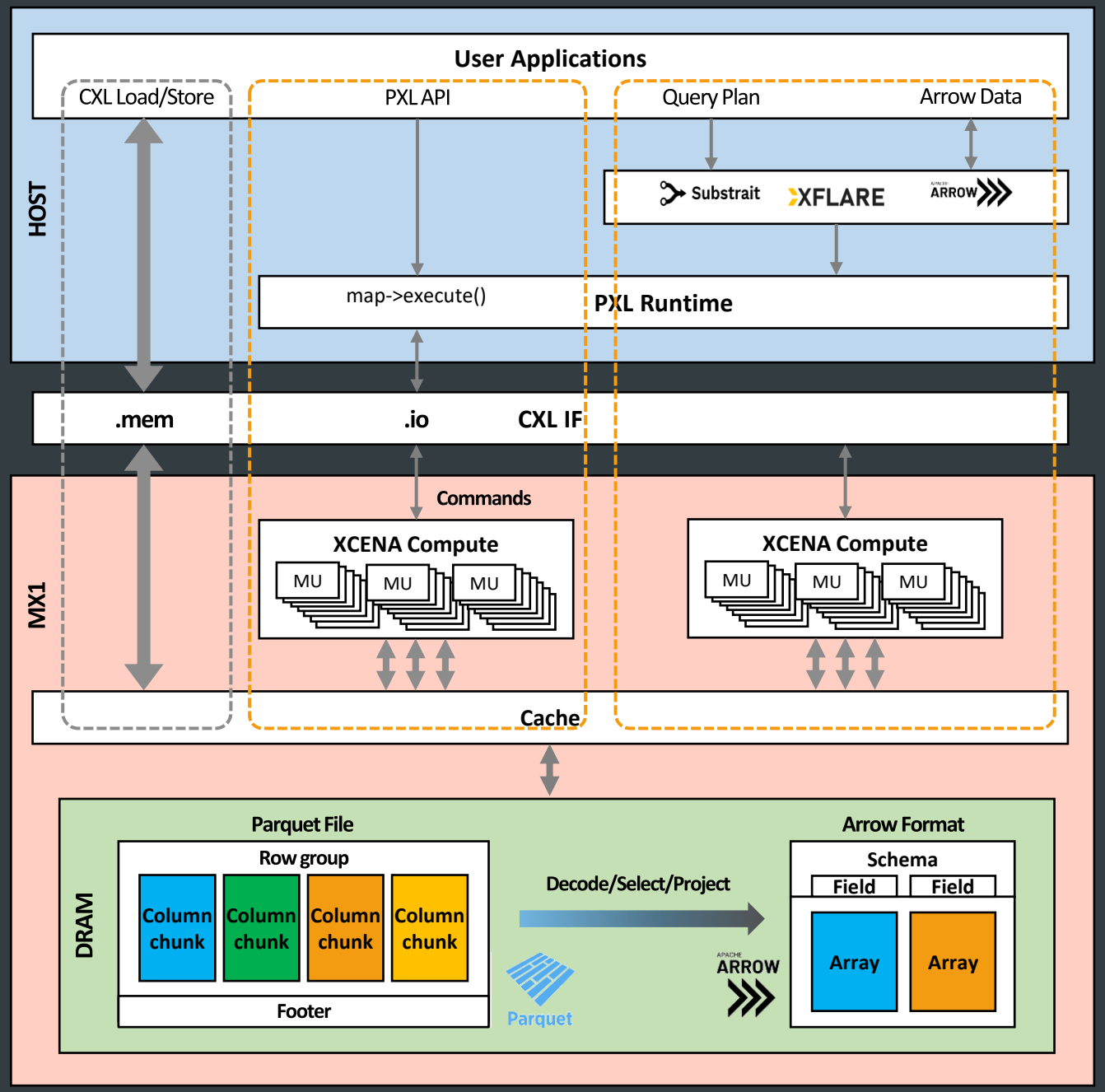
Use CXL memory as just another part of host memory.

PXL API

PXL allows you to manage resources and execute user kernels in parallel.

XFLARE

Common formats (Substrait, Arrow) and high-level APIs simplify query offloading.




SDK
XCENA
xcena.com/SDK
XCENA SDK Docs
Search XCENA SDK Docs
Introduction
Install
Workflow
Key Concepts
Get Started
Tutorials
Documentation
Release Notes
Overview
XCENA Software Development Kit (SDK) is a comprehensive software framework designed to facilitate easy and efficient utilization of XCENA hardware for data-intensive applications such as AI, tools, the XCENA SDK allows for seamless integration with CXL computational memory, enabling
Features
Seamless integration with computational CXL memory, specialized for data-intensive applications
Multi-level APIs for flexible, efficient development
Support for emulation and simulation with general tools like QEMU and our proprietary simulators, ensuring smooth development and testing environments
Software Ecosystem
Applications: Main applications running on the host system that leverage XCENA hardware for
XCENA.COM/SDK
QEMU CXL 3.0 Host + Computational Memory
XCENA SDK & Runtime Library
Example Codes and Tutorials
→ ALL IN A DOCKER IMAGE

THANK YOU

HARRY KIM, CPO

harry.kim@xcena.com

http://xcena.com

https://www.linkedin.com/company/xcena/