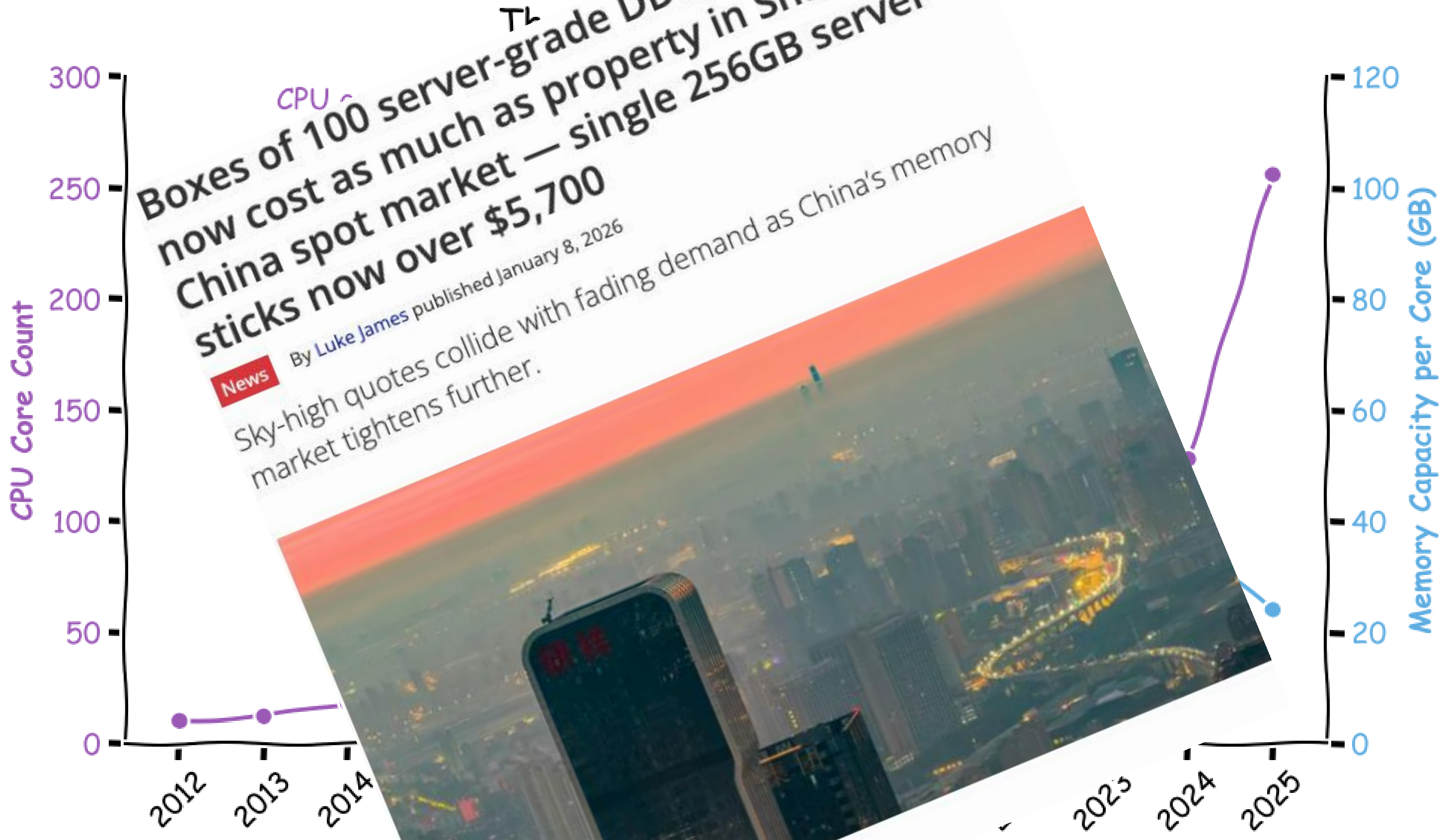
CIDR'26

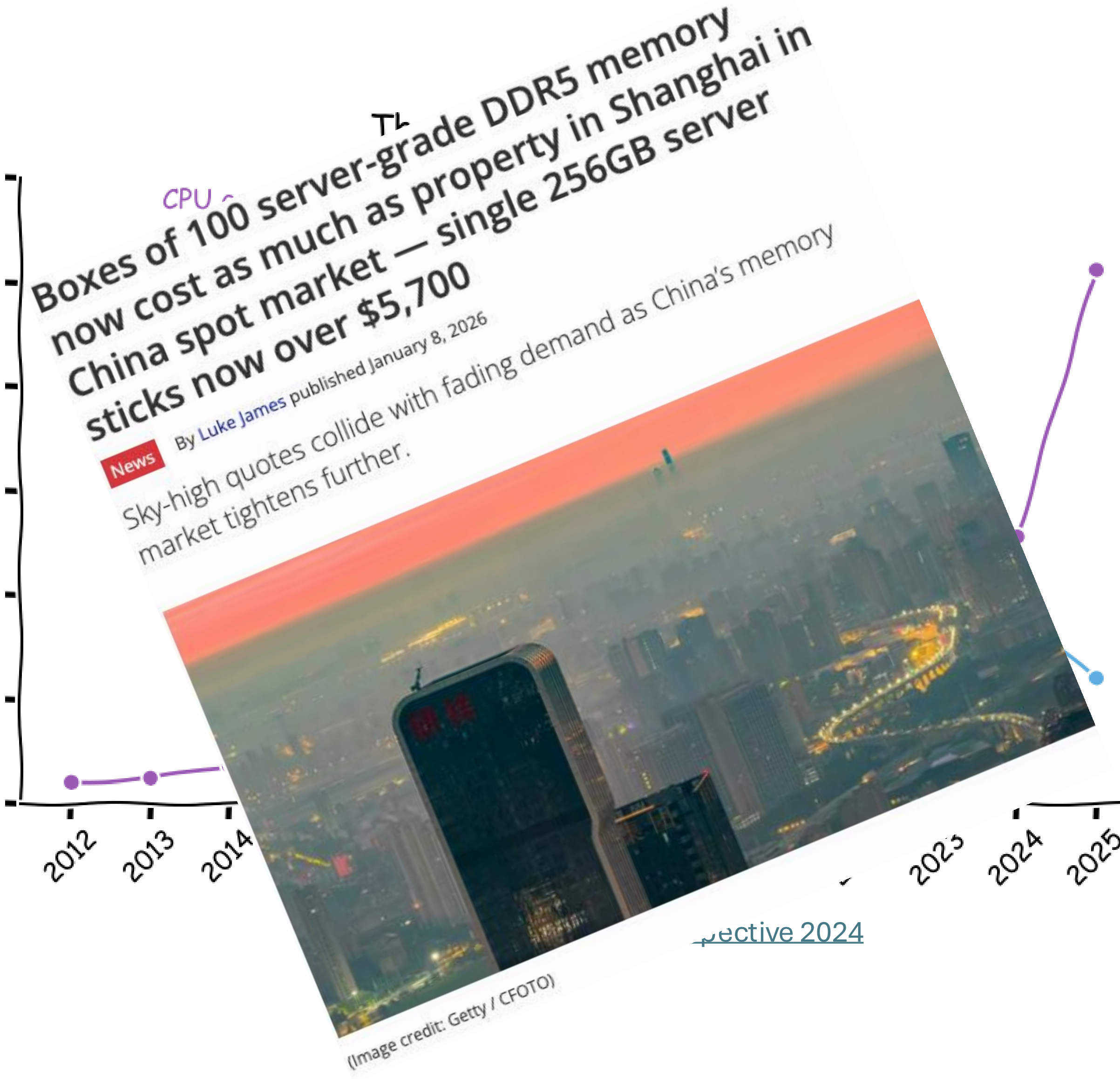
# Hash Joins Meet CXL: A Fresh Look

**Wentao Huang**[†] Mian Lu[‡] Kian-Lee Tan[†]

[†]National University of Singapore [‡]4Paradigm Inc.

# The Memory Scaling Wall





**Boxes of 100 server-grade DDR5 memory now cost as much as property in Shanghai in China spot market — single 256GB server sticks now over $5,700**

News By Luke James published January 8, 2026

Sky-high quotes collide with fading demand as China's memory market tightens further.

(Image credit: Getty / CFOTO)

pective 2024

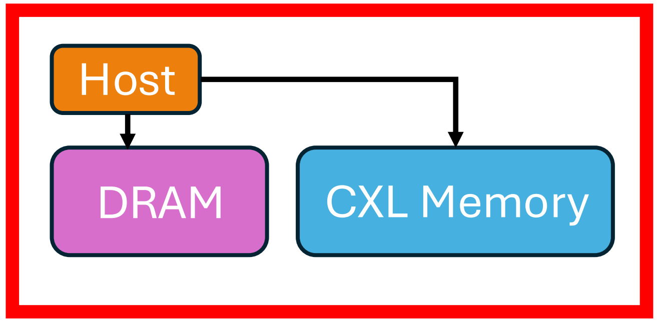
## Current Dilemma

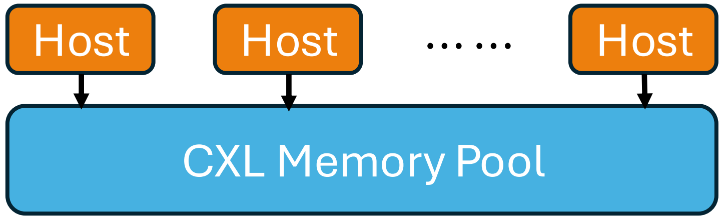
- ● Limited capacity-per-core
- ● Declining bandwidth-per-core
- ● Rising memory provisioning cost

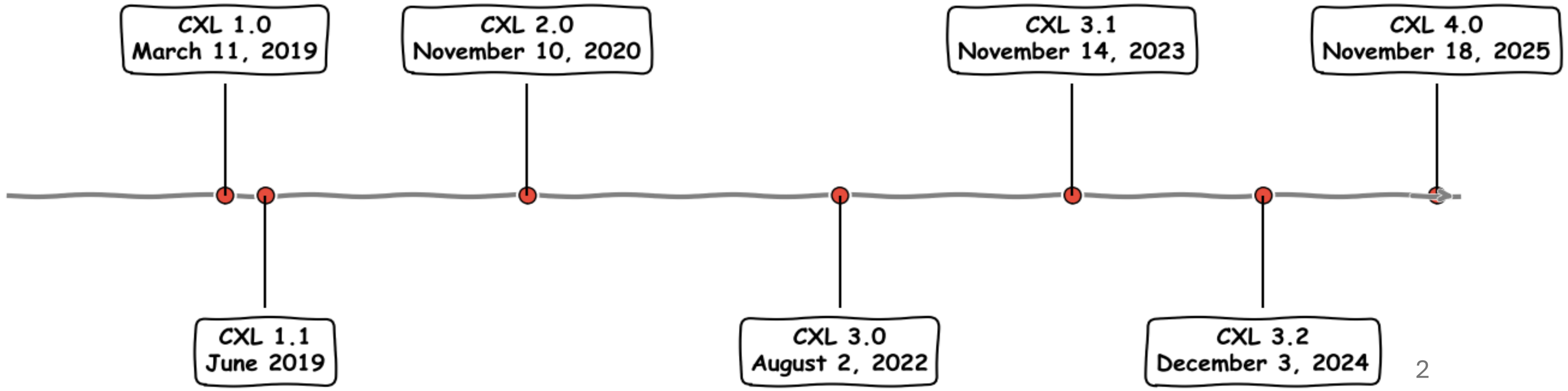
# The CXL Memory Tehcnology

CXL memory enables

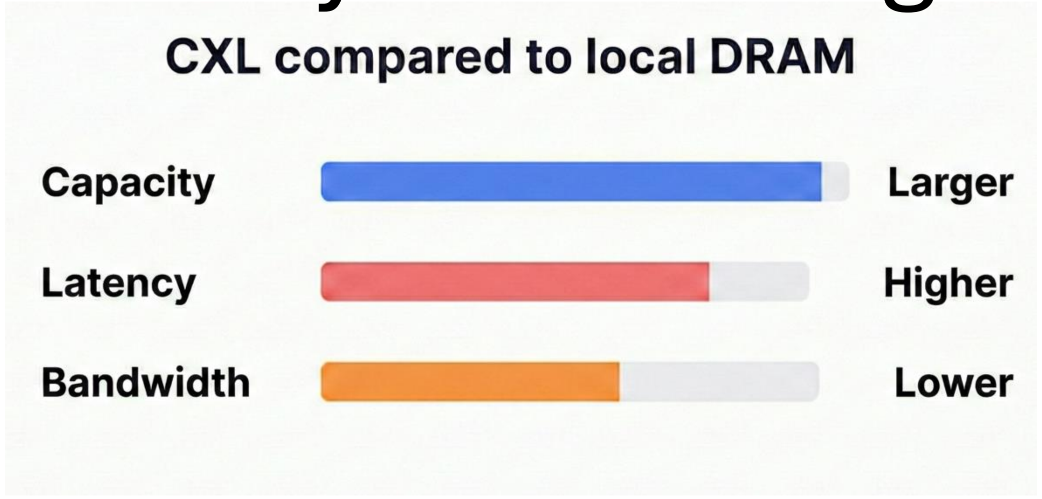
- Cache-coherent PCIe access
- Capacity scaling
- Bandwidth expansion
- Memory pooling & sharing
- Reuse of previous-gen RAM
- Low cost-per-byte
- ……

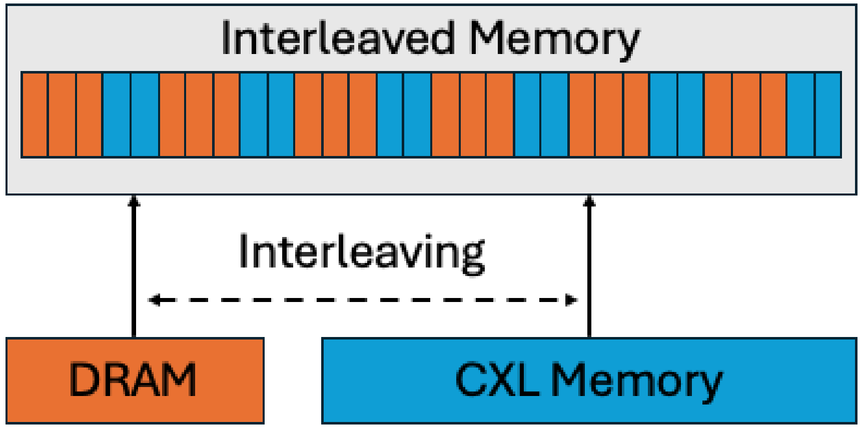


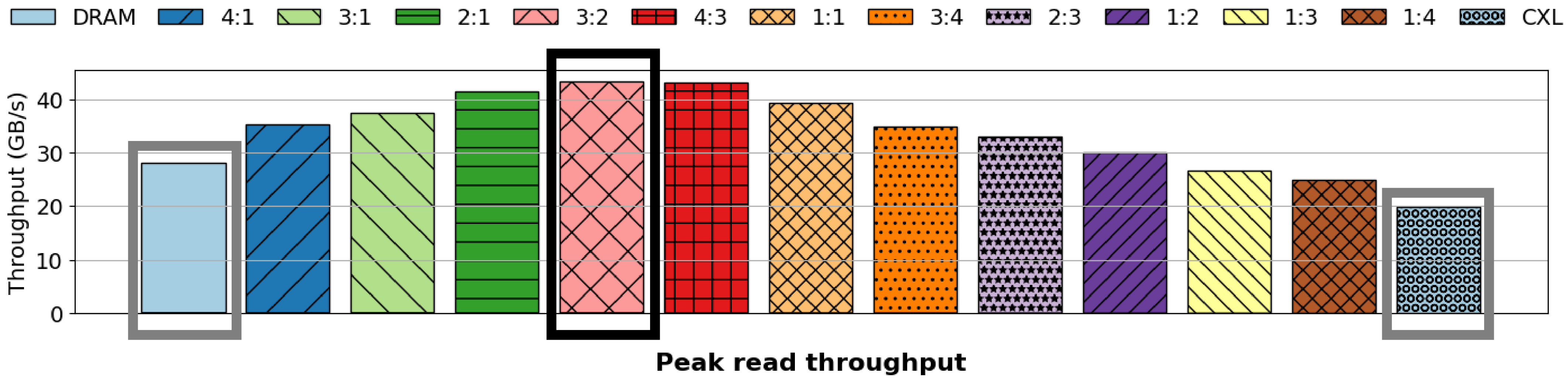




# Memory Interleaving



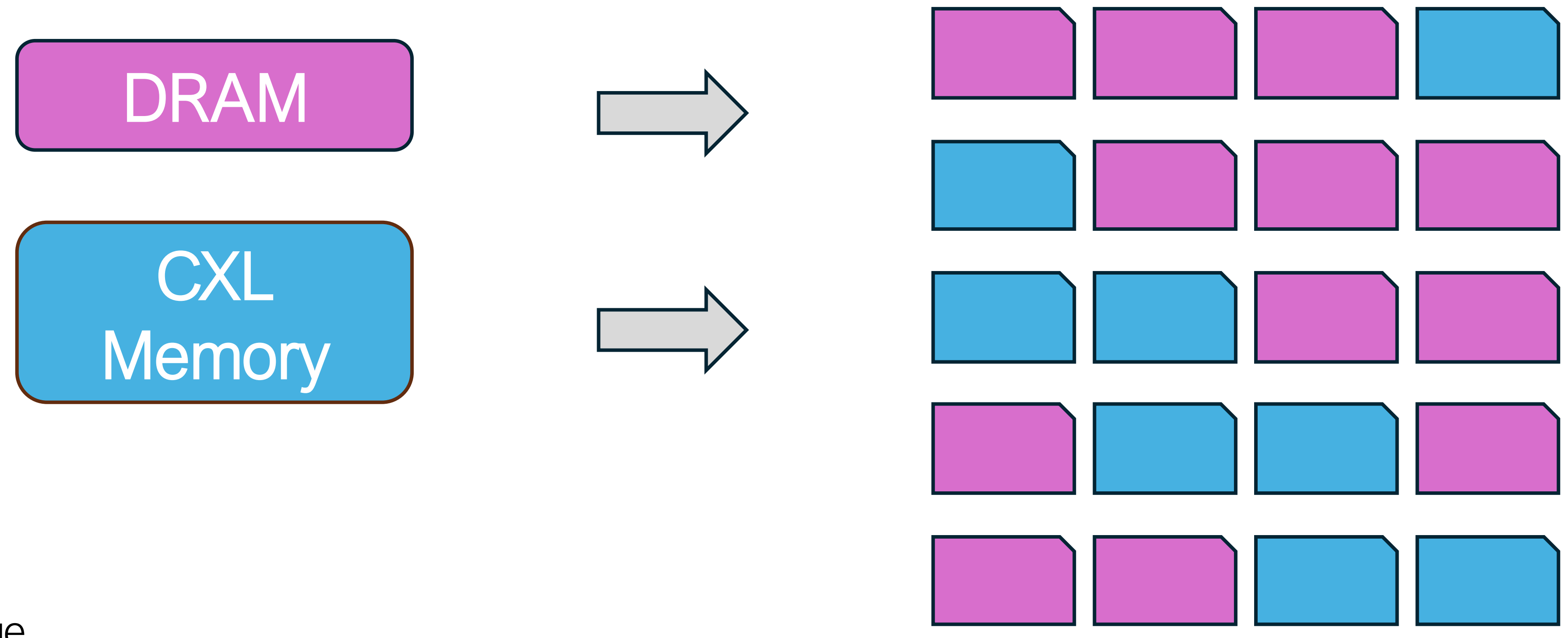




Source: Huang et al. 2024

## The Status Quo Approach

Step 1 — create an interleaved memory tier

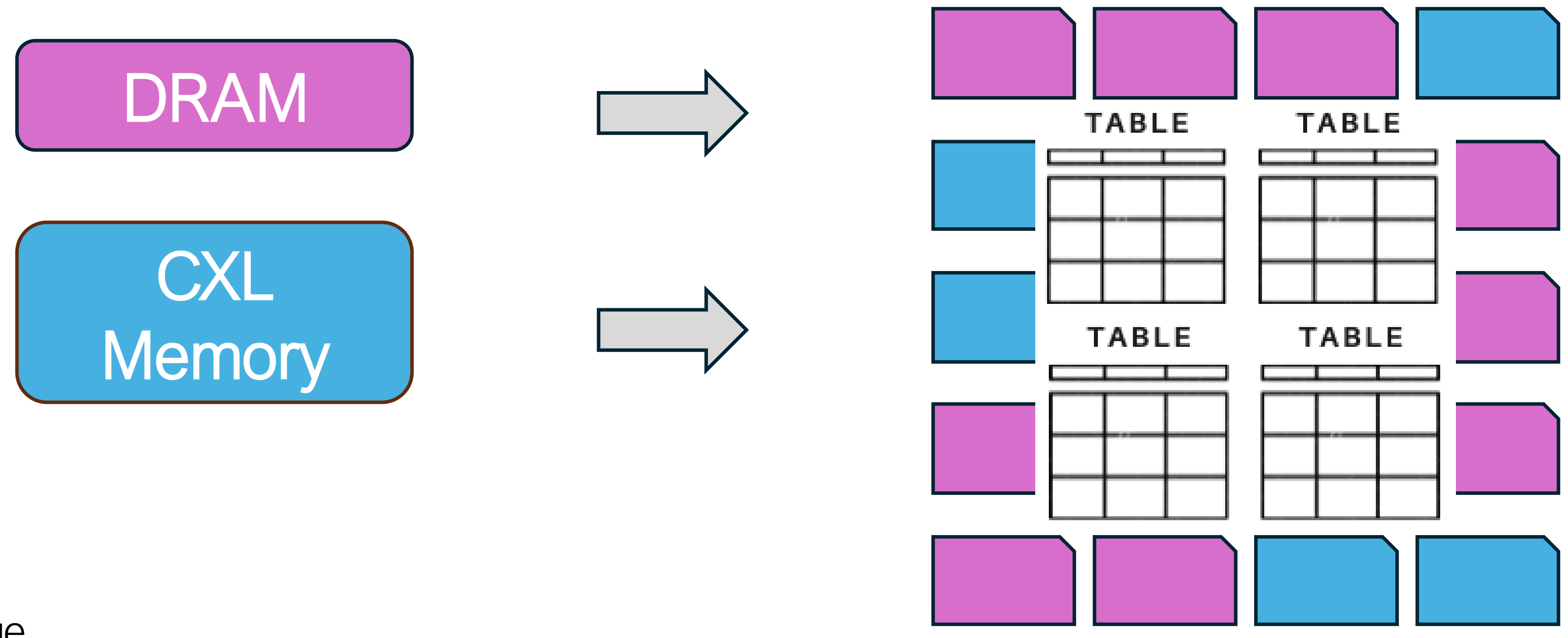


🟪 DRAM page

🟦 CXL memory page

# The Status Quo Approach

Step 2 — relocate the entire workload to this tier

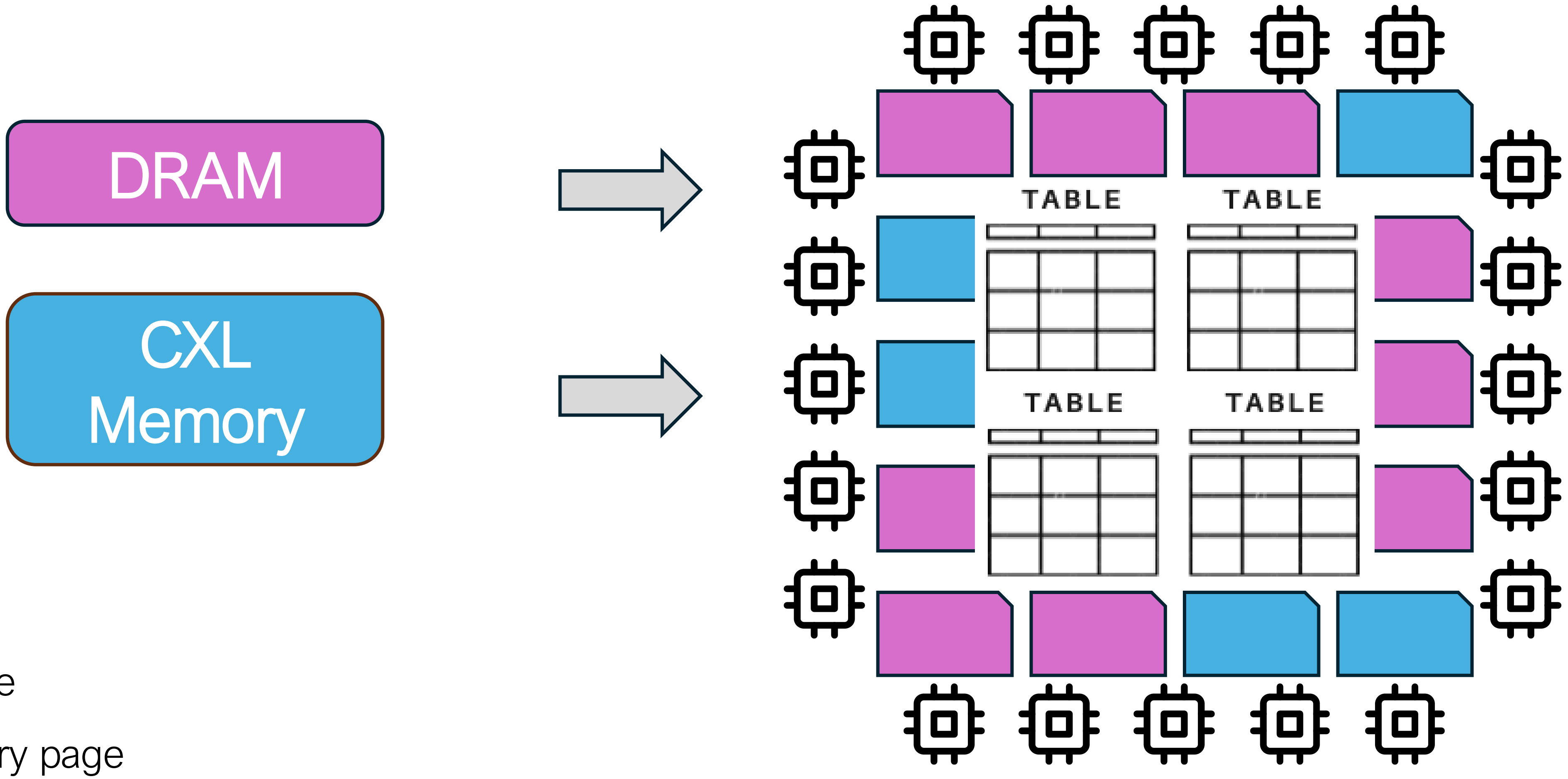


🟪 DRAM page

🟦 CXL memory page

# The Status Quo Approach

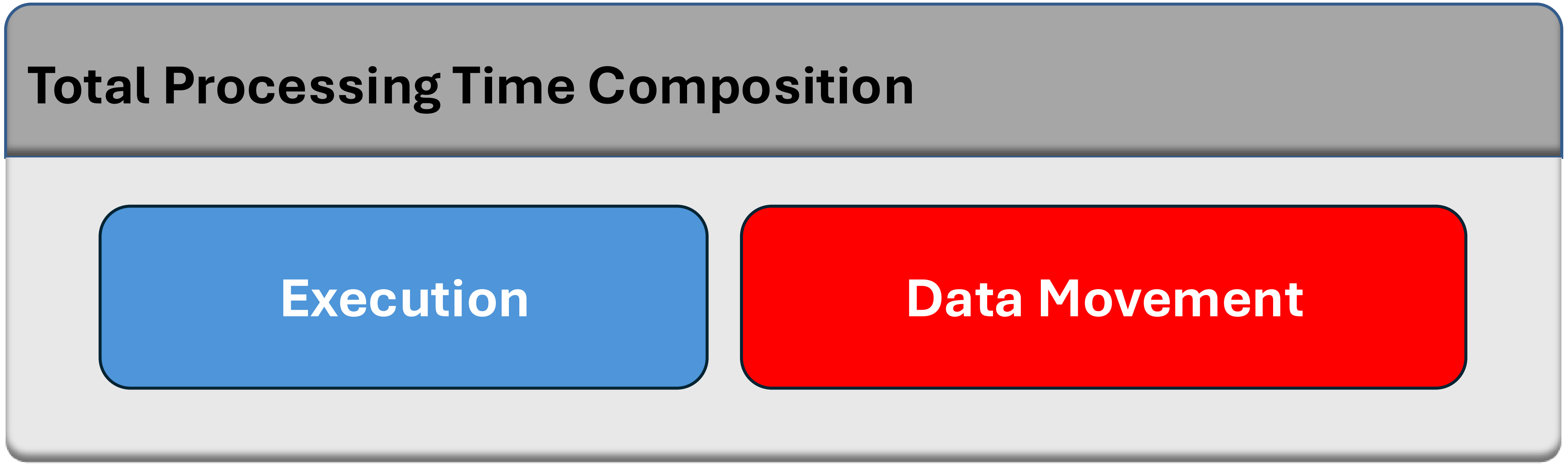
Step 3 — run the workload in-place to maximize throughput



🟪 DRAM page

🟦 CXL memory page

# The Hidden Overhead



**The Blindspot**

Most large datasets start in CXL (the capacity tier) due to capacity/economic constraints.

**The Trap**

Moving overhead sometimes offsets bandwidth benefits, sometimes slower than just running in CXL!

## Shall We Stick to the Current Approach?

**The Status Quo Approach**

1. create an **interleaved memory tier** (DRAM + CXL)
2. relocate the **entire workload** to this memory tier
3. run the workload **in-place** to maximize throughput

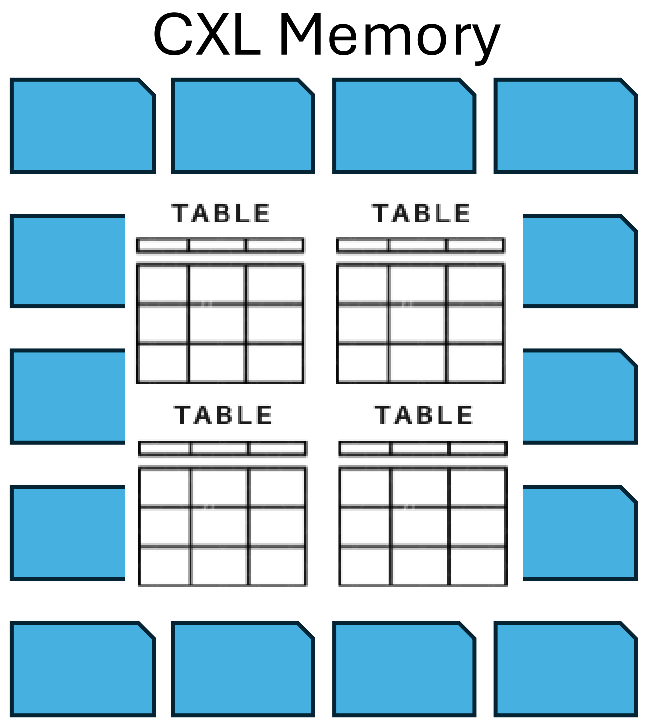
## Maybe a Better Way?

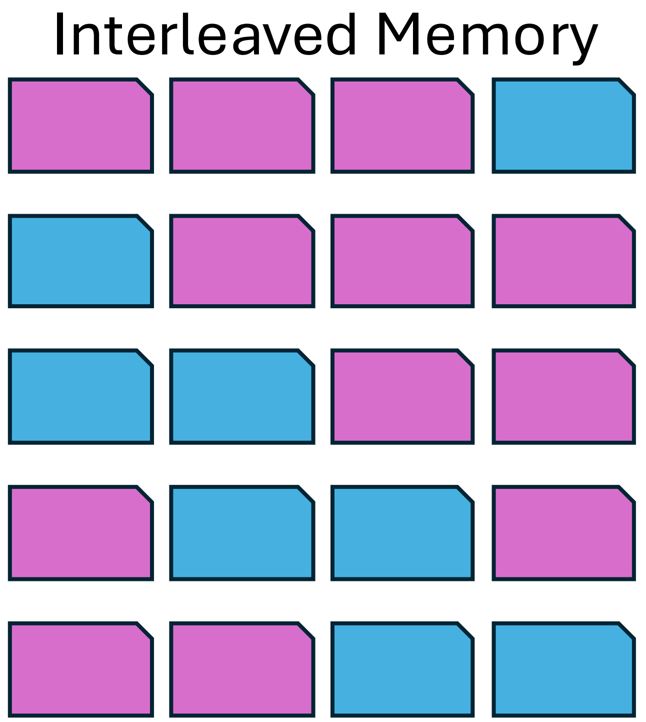
1. **Where should we move the workload?**
2. **How much workload should we move?**

**Reduce Data Movement Cost**

**Maximize Overall Processing Throughput**

## Data Movement Analysis: Destination





**Total Memory Traffic**

12 × 🟪 28 × 🟦

**The Fact**

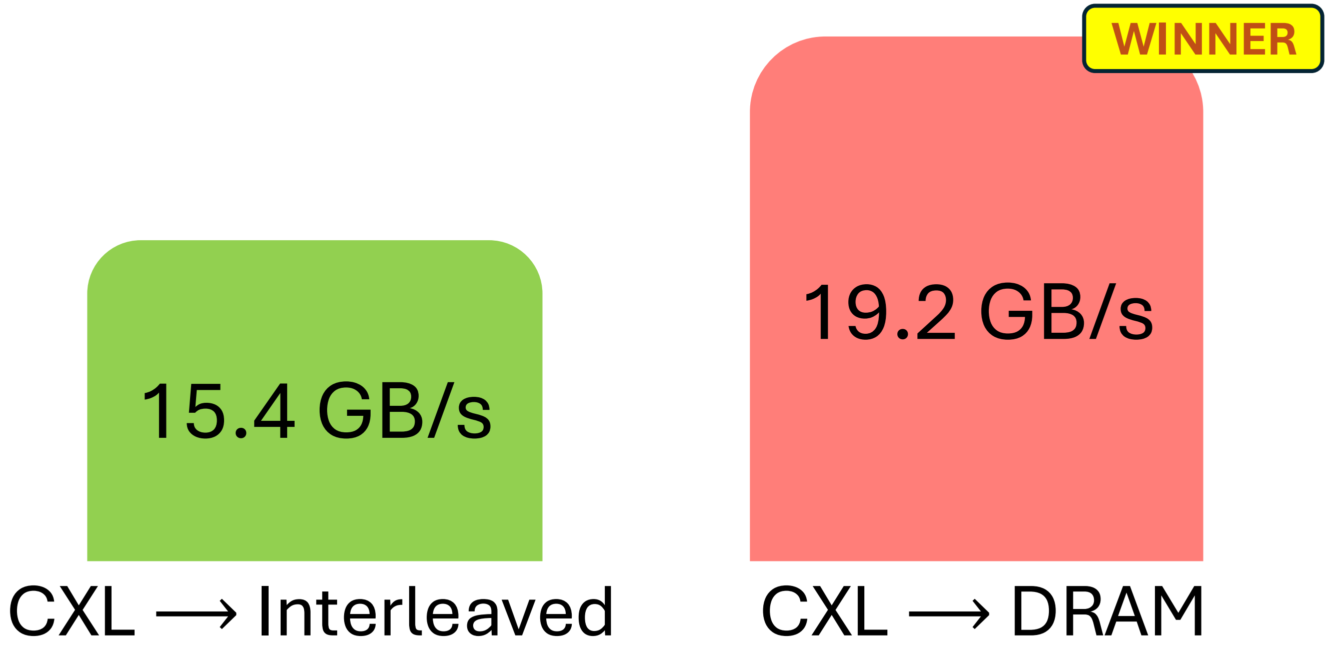
🟪 : 🟦 = 3 : 7

**The Goal**

🟪 : 🟦 = 3 : 2

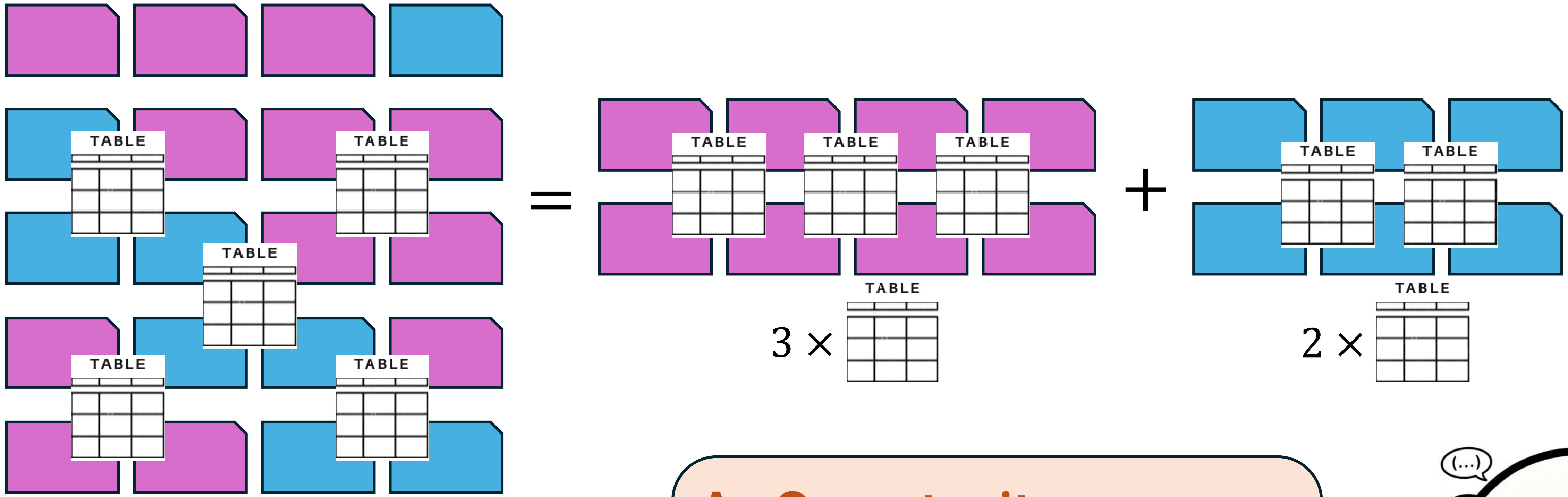
🟪 DRAM page 🟦 CXL memory page

# Data Movement Analysis: Destination



**Move to DRAM for peak performance**

# Data Movement Analysis: Volume



🟪 DRAM page

🟦 CXL memory page

**An Opportunity**

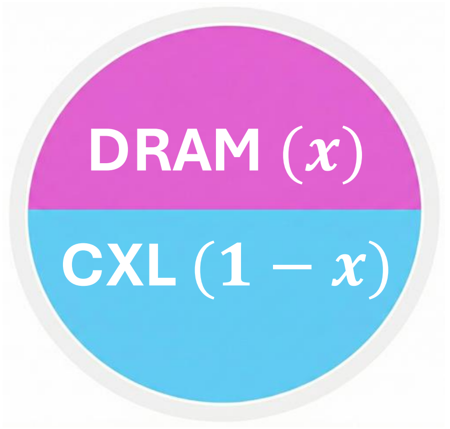
High processing throughput with reduced data movement?

# An End-to-End Processing Approach

Our Proposal

## Partial Data Movement

- **Do not move everthing**
  Avoid massive overhead of full workload movement.
- **Software-defined Interleaving**
  Accessing both DRAM and CXL for higher throughput
- **Calculate movement fraction $x$**
  Move only optimal fraction $x$ to DRAM; Leave $(1 - x)$ in CXL memory.



**Virtual "Interleaved" Tier**

**Model**

**Tradeoff in $x$**

data movement cost vs. processing throughput

## Applying to Main-Memory Hash Joins

**Partitioned Hash Join (PHJ)**

Bandwidth heavy: partition phase

- Apply model to the **parititon** phase
- Move $x$ amount of input to DRAM
- Partition output goes to the **interleaved** tier (for maximizing **write** throughput).

**Non-Partitioned Hash Join (NPHJ)**

Latency hiding: build & probe phase

- Apply model to **build and probe** phase
- Move $x$ amount of the build side to DRAM
- The built Hash table goes to the **interleaved** tier (for maximizing **write and probe** throughput).

Source: Schuh et al. 2016

## Experimental Setup

**CXL 1.1**

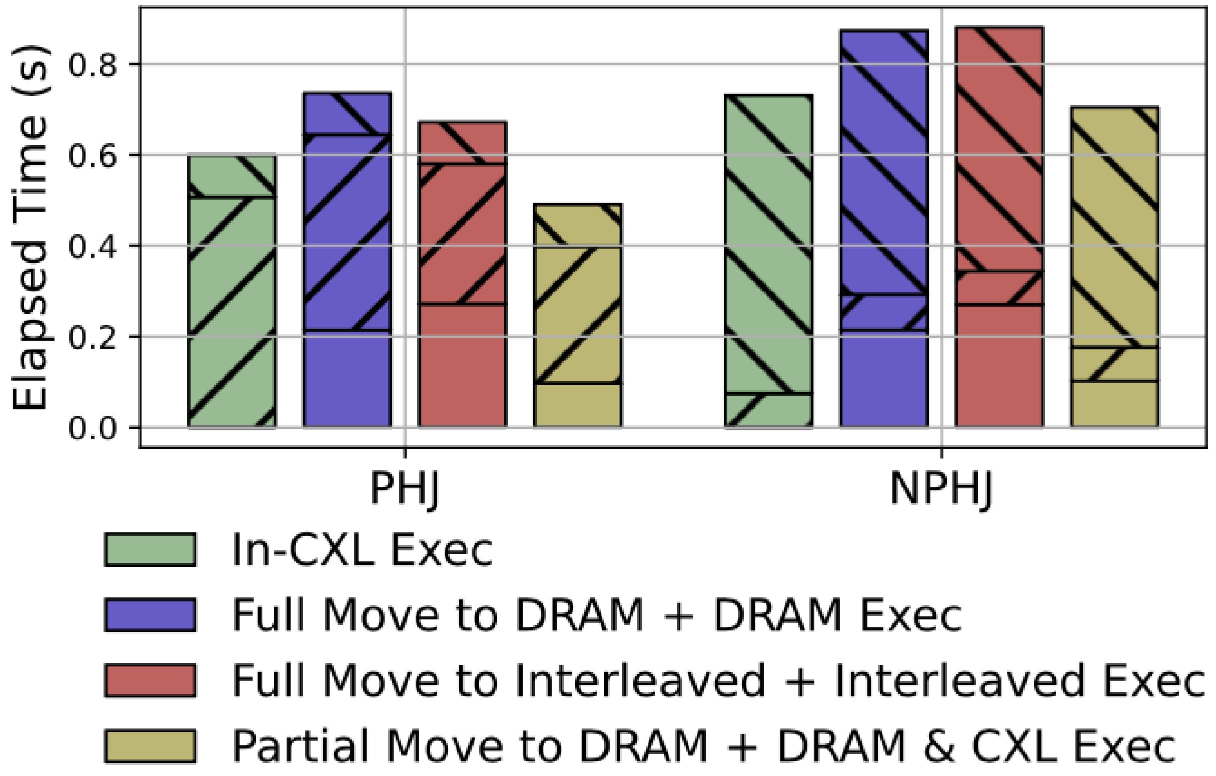
Real CXL hardware platform

| DRAM | CXL | Interleaved |
|---|---|---|
| 32 GB | 64 GB | 53 GB |
| 30 GB/s Bandwidth | 21 GB/s Bandwidth | 47 GB/s Bandwidth |

**Workload**

- Synthetic equi-join benchmark (16-byte tuples)
- Cardinatlity: build side 16M, probe side 256M

# Experimental Results



**"Partial move" beats all baselines**

**Partitioned Hash Join**

**~22% runtime reduction**

**vs. In-CXL execution**

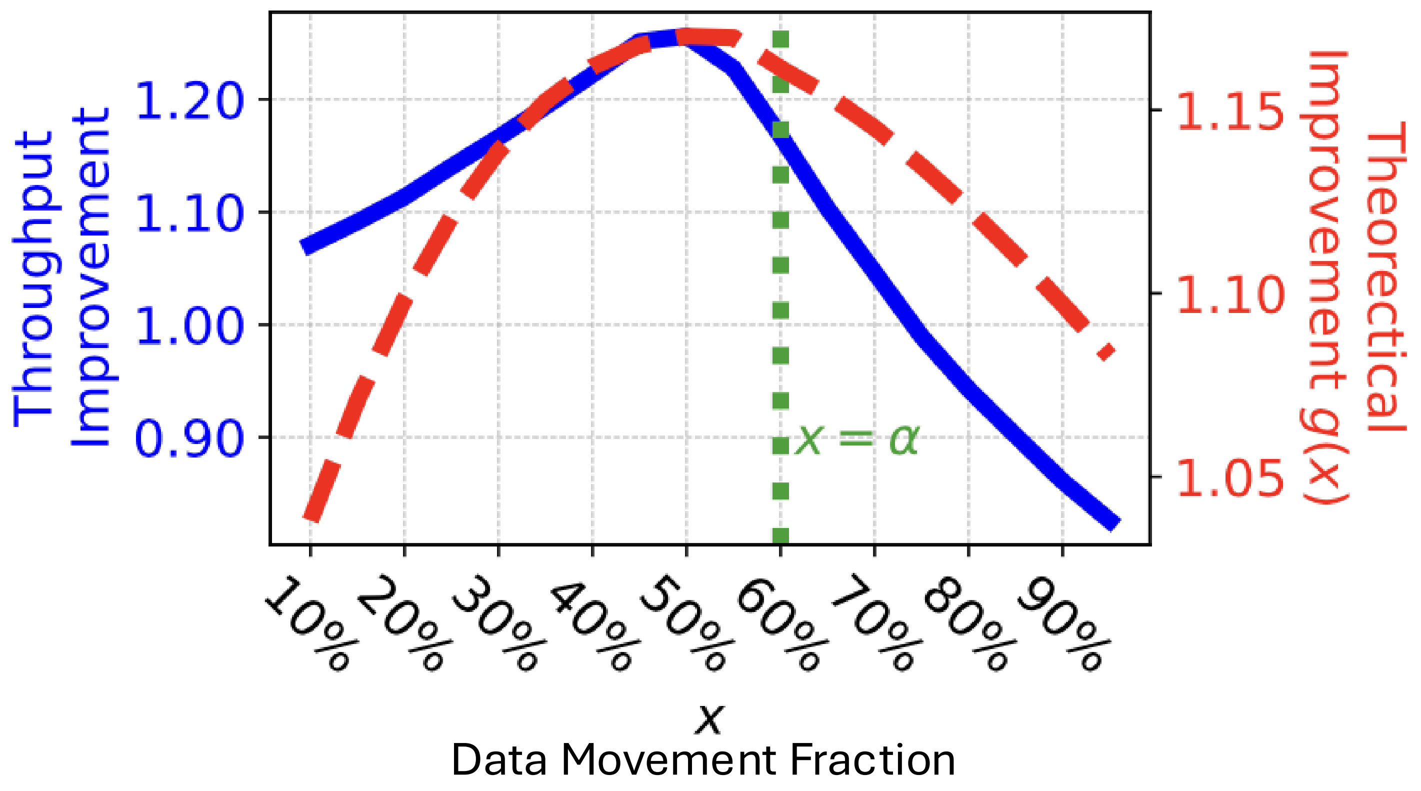
**Non-Partitioned Hash Join**

**~4% runtime reduction**

**vs. In-CXL execution**

## Sentivity Analysis

**Theoretical vs. Measured in Partitioning**



**The model can determine the optimal data movement fraction**

## Conclusion

**Takeaway**

- **Interelaving ≠ Answer:** data movements costs are real
- **Less can be More:** Parital movement beats full relocation
- **The Winning Strategy:** Our model finds the optimal balance

Q & A