Collaboration between CPU and Near-Data Accelerators for ML Training

Benjamin Cho, Yongkee Kwon, Sangkug Lym, and Mattan Erez
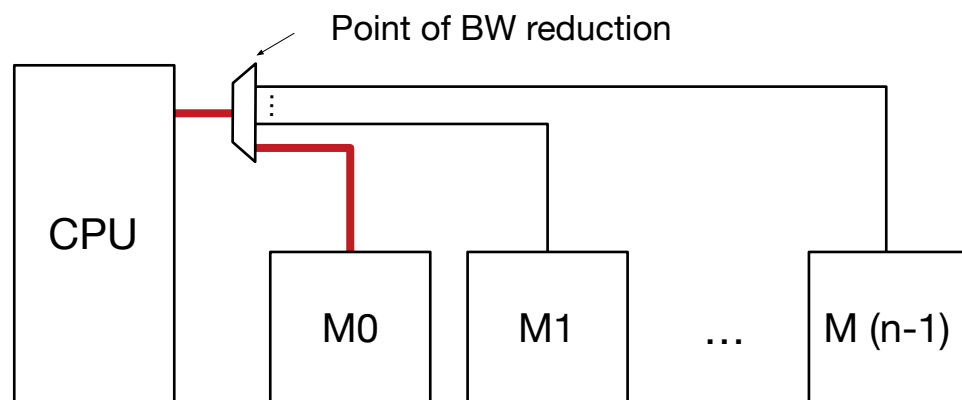
The University of Texas at Austin
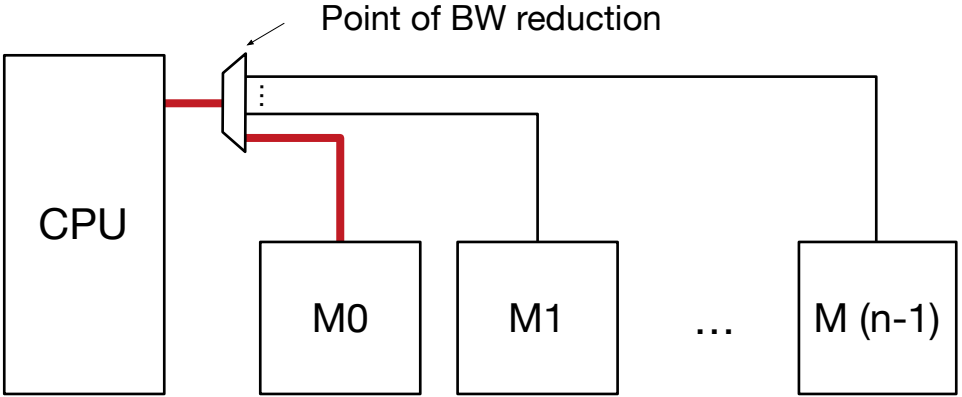
OCP REGIONAL SUMMIT
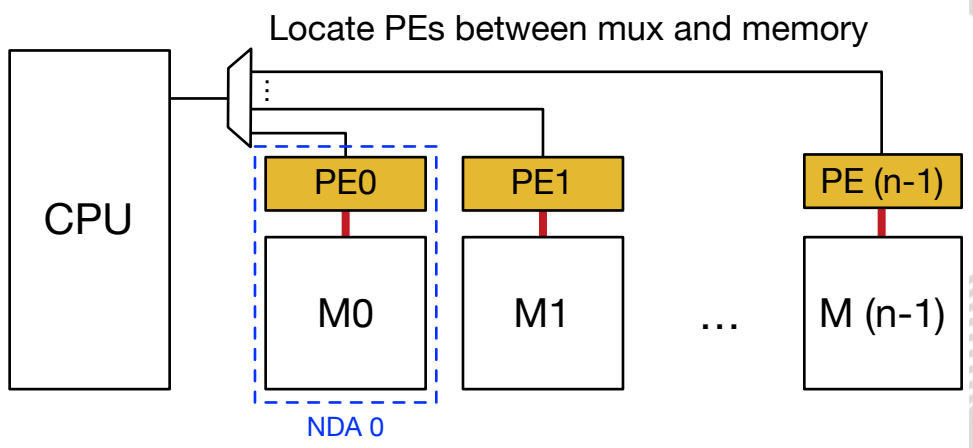
Near-Data Acceleration

- Normal memory access

Point of BW reduction

CPU

M0   M1   …   M (n-1)

Near-Data Acceleration

- Normal memory access

Point of BW reduction

CPU

M0   M1   ...   M (n-1)

- Near-data memory access

Locate PEs between mux and memory
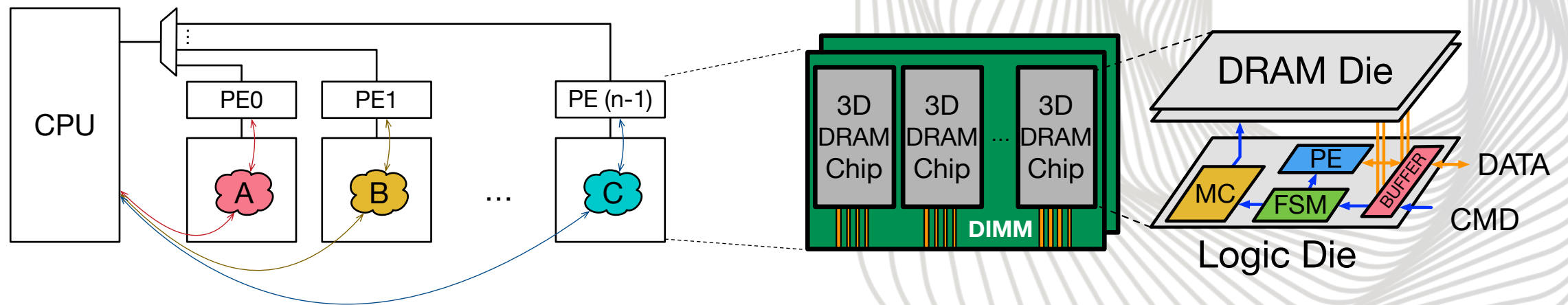
CPU

PE0   PE1   PE (n-1)

M0   M1   ...   M (n-1)

NDA 0

High memory BW + Low memory-access energy

Near-Data Acceleration **in Main Memory**

- NDAs: accelerators with high memory capacity
- Collaboratively process the same data

Collaborative processing

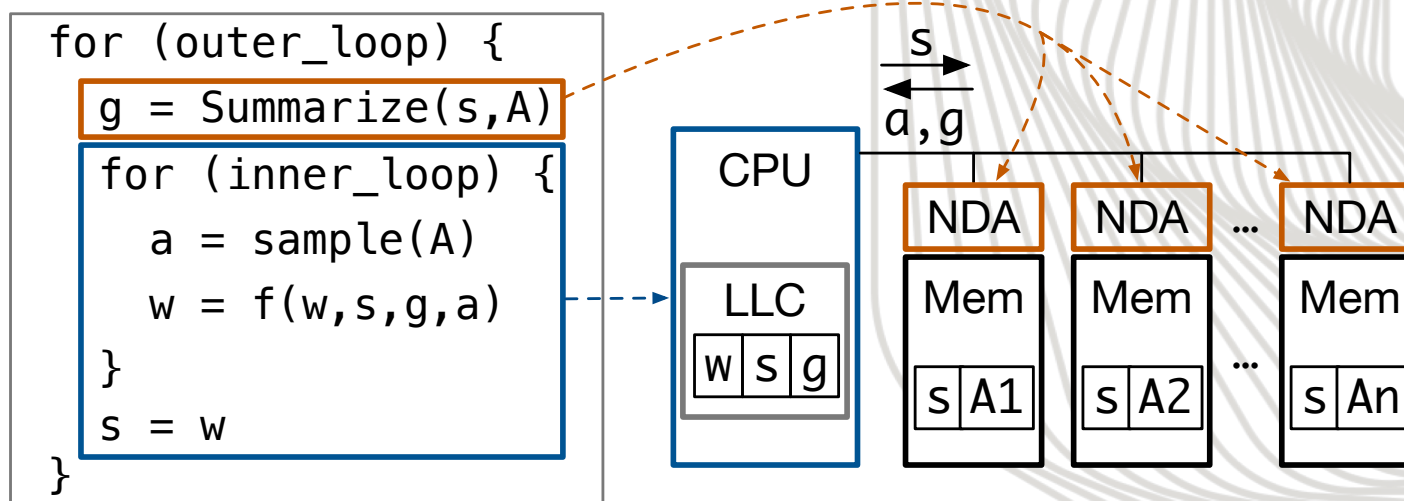Example per-rank NDA in high-capacity DRAM

An Ideal Scenario for CPU-NDA Collaboration

- **Two workloads with different requirements**
  - Latency sensitive → CPU
  - Throughput oriented → NDAs

- **Sharing large read-only data**
  - Data replication → expensive
  - No coherence required

Any applications that meet above conditions?

Collaboration Opportunity in SVRG

Proc.	Workload	Advantage
CPU	Main training	Cache locality
NDA	Summarization	Memory BW


```
for (outer_loop) {
  g = Summarize(s,A)
  for (inner_loop) {
    a = sample(A)
    w = f(w,s,g,a)
  }
  s = w
}
```

Exploiting Concurrent Access in SVRG

Original algorithm

Exploiting Concurrent Access in SVRG

Collaboration Results

- Algorithm: Logistic regression with $l2$ regularization
- Dataset: CIFAR10
- Compare convergence speed
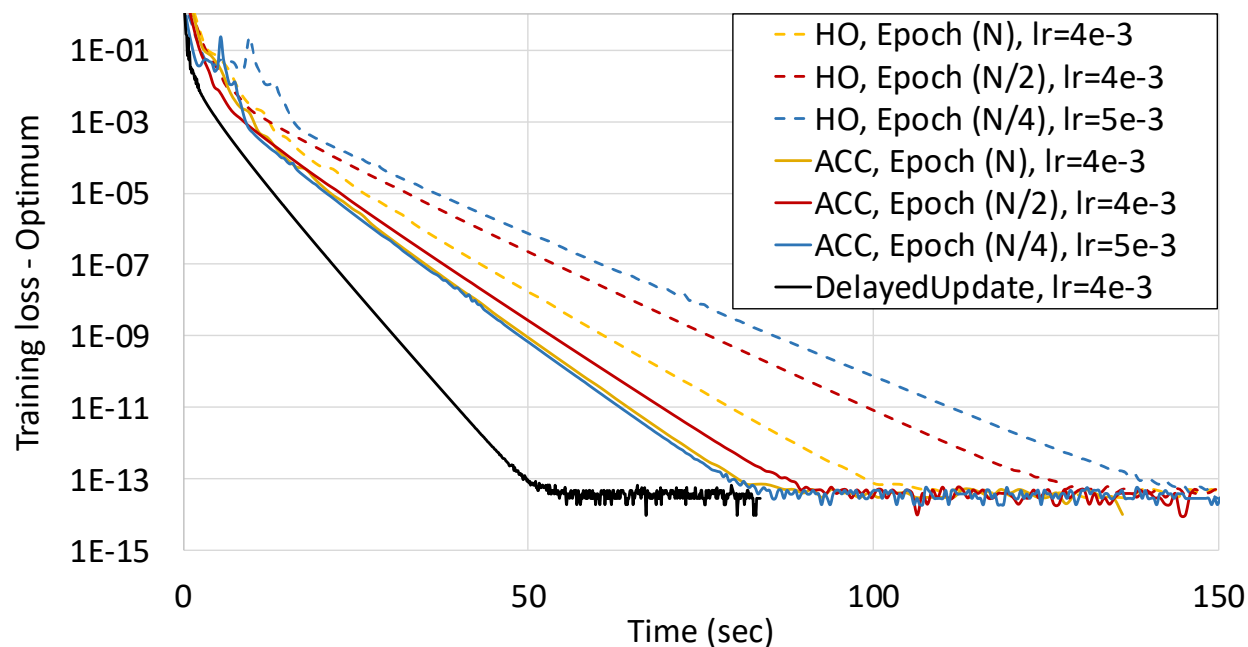 - **HO**: host-only execution
  - **ACC**: accelerating summarization with NDAs
  - **DelayedUpdate**: concurrent host-NDA execution

Legend (chart):
- HO, Epoch (N), lr=4e-3
- HO, Epoch (N/2), lr=4e-3
- HO, Epoch (N/4), lr=5e-3
- ACC, Epoch (N), lr=4e-3
- ACC, Epoch (N/2), lr=4e-3
- ACC, Epoch (N/4), lr=5e-3
- DelayedUpdate, lr=4e-3

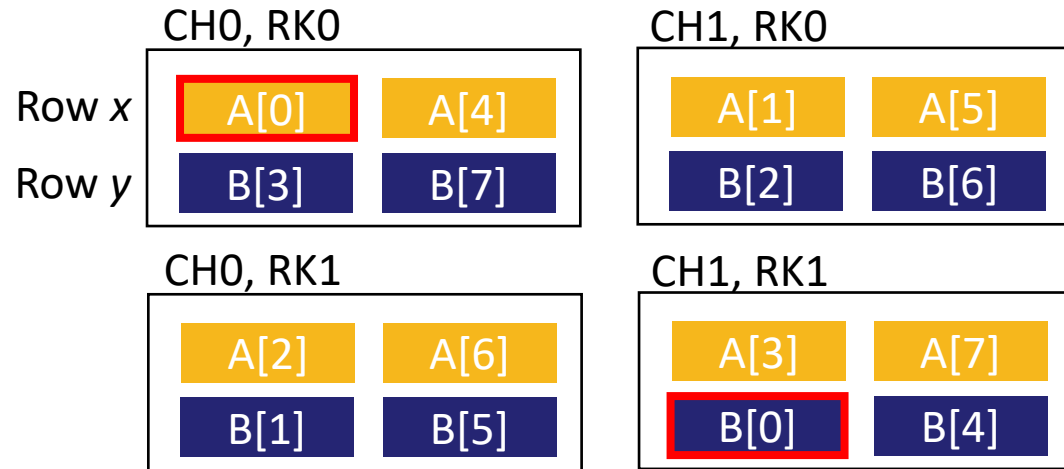Y-axis: Training loss - Optimum
X-axis: Time (sec)

Main Challenges for Concurrent Access

1. A common **data layout** for the CPU and NDAs

2. **Tracking** global memory controller (MC) **state**

Interleaving Pattern-Aware Memory Allocation

For A[i] += B[i],   Naïve Layout

CH0, RK0

Row x	A[0]	A[4]
Row y	B[3]	B[7]

CH1, RK0

A[1]	A[5]
B[2]	B[6]

CH0, RK1

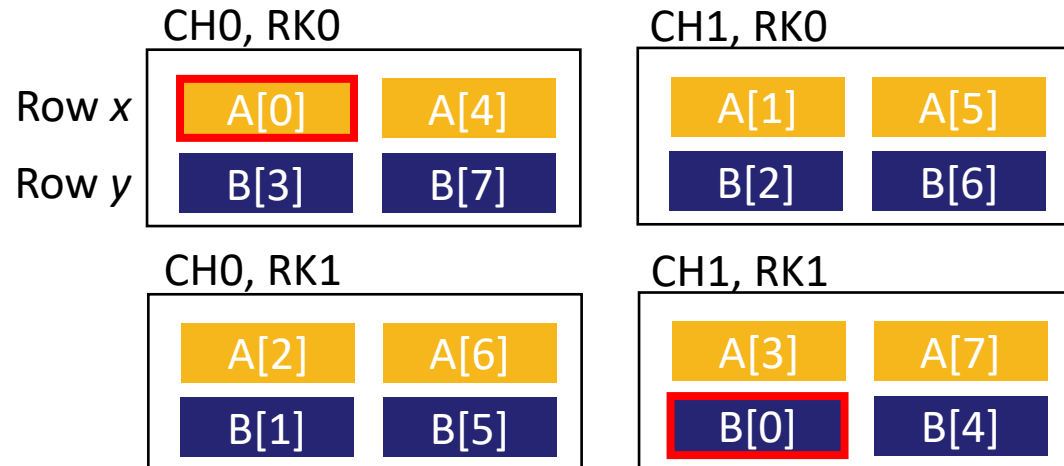|	
A[2]	A[6]
B[1]	B[5]

CH1, RK1

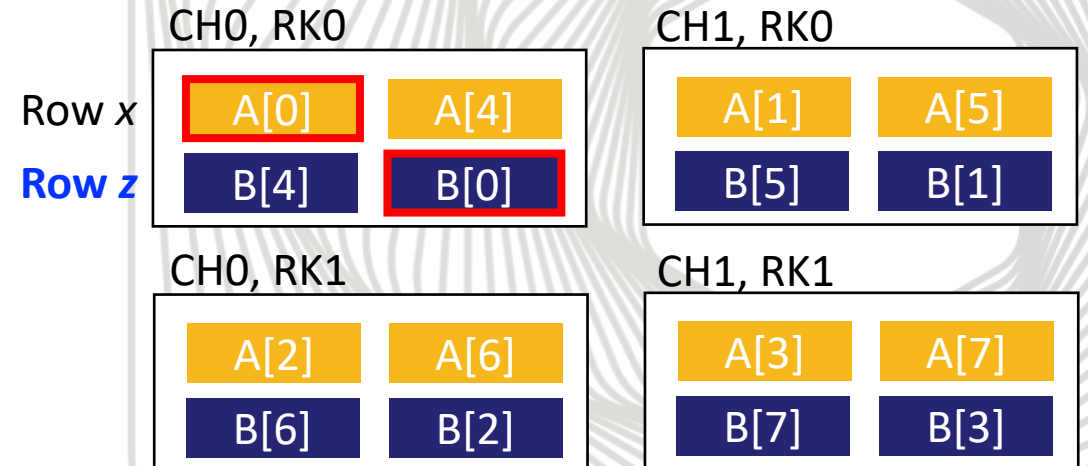|	
A[3]	A[7]
B[0]	B[4]

Interleaving Pattern-Aware Memory Allocation

For A[i] += B[i],

Naïve Layout

Proposed Layout

Address interleaving (for CPU) + Rank locality (for NDAs)

Problem of Uncoordinated MCs

@cycle **x**

Host MC → NDA0 MC → M0   Open Bank 1, Row 54

Invisible to host MC

NDA1 MC → M1   Read Bank 7, Row 12

Tracks incoming host commands

Problem of Uncoordinated MCs

Invisible to host MC

@cycle **x**

| Host MC | → | NDA0 MC | → | M0 | Open Bank 1, Row 54 |

| | | NDA1 MC | → | M1 | Read Bank 7, Row 12 |

Tracks incoming host commands

@cycle **x + α**

| Host MC | → | NDA0 MC | → | M0 | Host MC doesn't know bank & timing state of M0 |

| | | NDA1 MC | → | M1 | Close Bank 7 after β cycles |

Replicated FSMs

Standardizing Concurrent Access

- Standardize **address mapping interface**

  for concurrently accessible data layout

- Standardize **NDA operations**

  for low-cost MC state tracking

Concurrent Access to Different Data*

- **Main direction: avoiding/mitigating contention**
  - Fine-grained access interleaving
  - Coarse-grained NDA operations
  - Partitioning into CPU-only and shared memory regions
  - Mitigating write/read turnaround penalties with NDA write throttling

*CHoNDA: Near Data Acceleration with Concurrent Host Access (arXiv 2019)
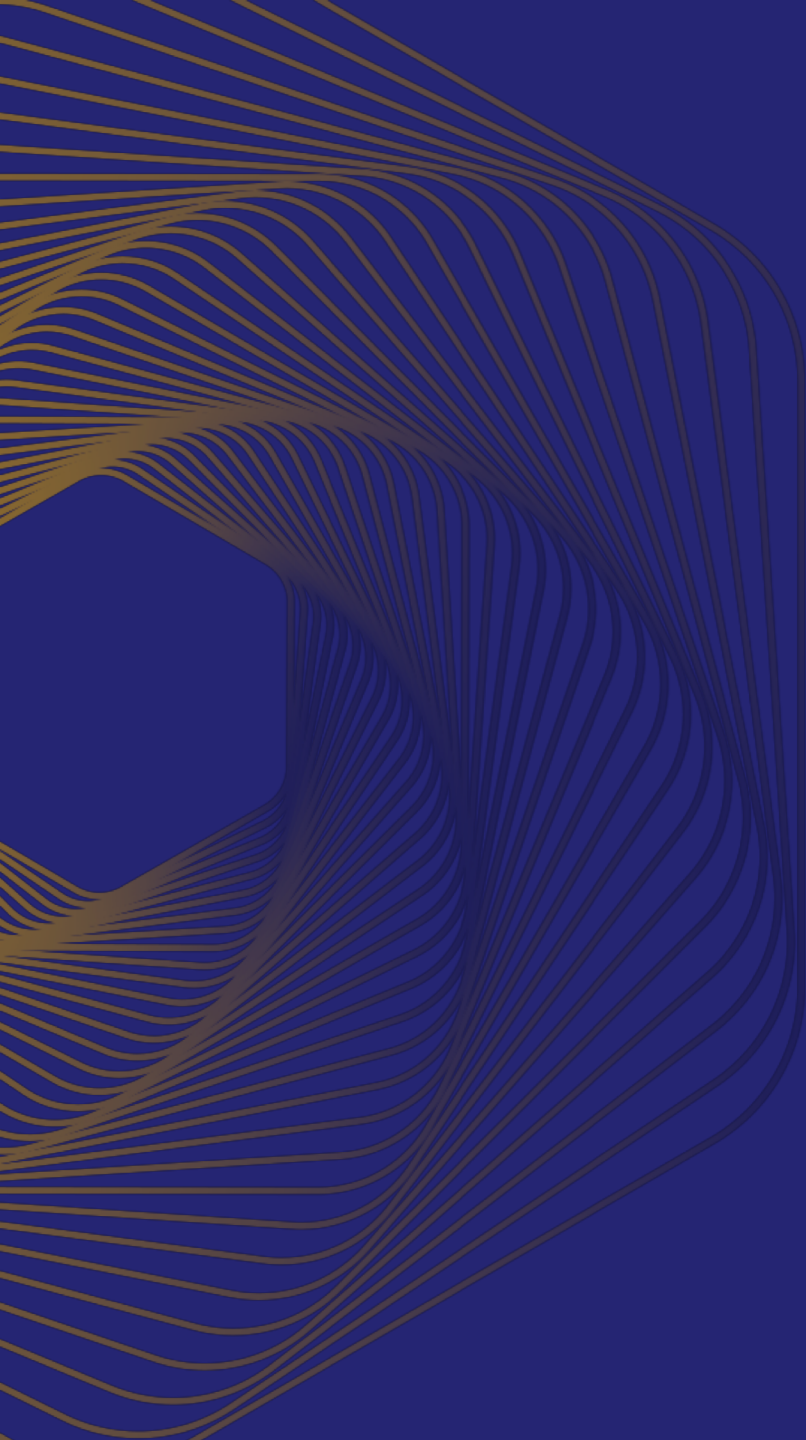
Open. Together.

Conclusion

- **Opportunity**: Collaborative CPU-NDA execution for shared data
- **Potential benefit**: 2x faster convergence speed in the SVRG case study
- **Challenges**
  - A common data layout for concurrent access
  - Tracking global memory controller state
- **Solutions**
  - Interleaving pattern-aware memory allocation
  - Replicated FSMs
- **And more…**
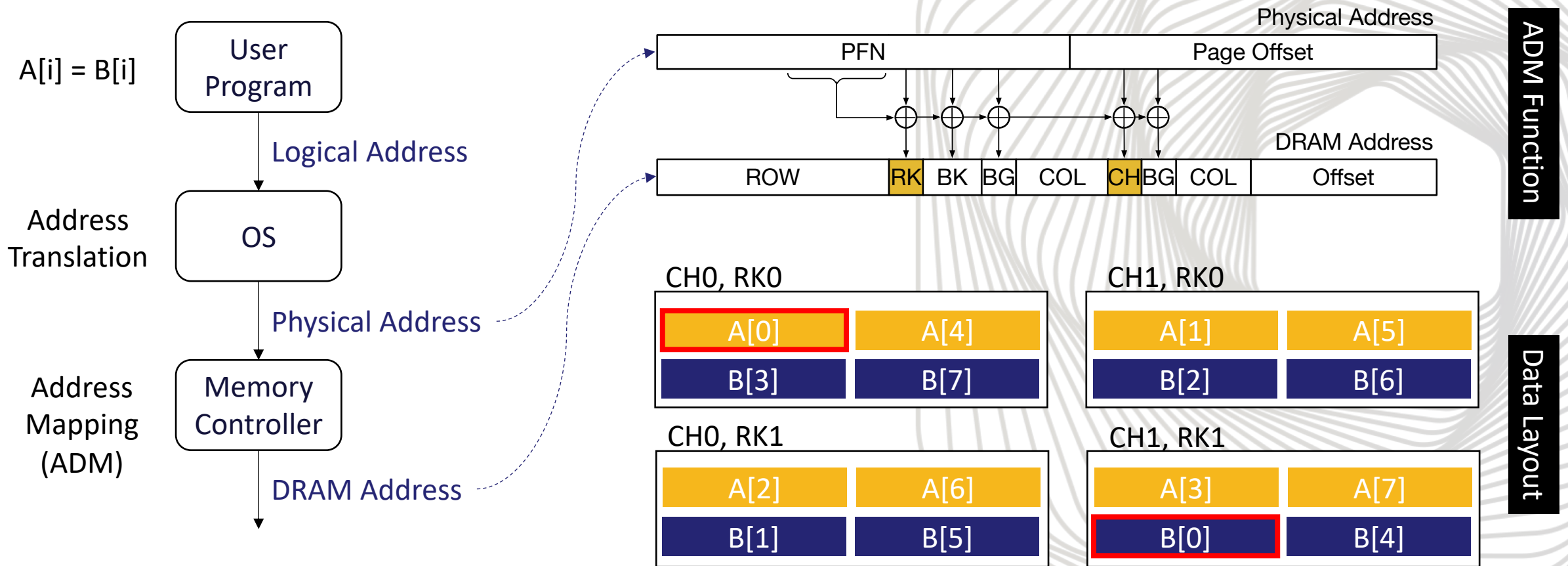 - Concurrent access to different data

Open. Together.

OCP
REGIONAL
SUMMIT

Backup Slides

Challenge 1: Data Layout

Challenge 2: MC State Tracking

Invisible to host MC

@cycle **x**

Host MC → NDA0 MC → M0   Open Bank 1, Row 54

Host MC → NDA1 MC → M1   Read Bank 7, Row 12

Tracks incoming host commands

@cycle **x + α**

Host MC → NDA0 MC → M0   Host MC doesn't know bank & timing state of M0

Host MC → NDA1 MC → M1   Close Bank 7 after β cycles
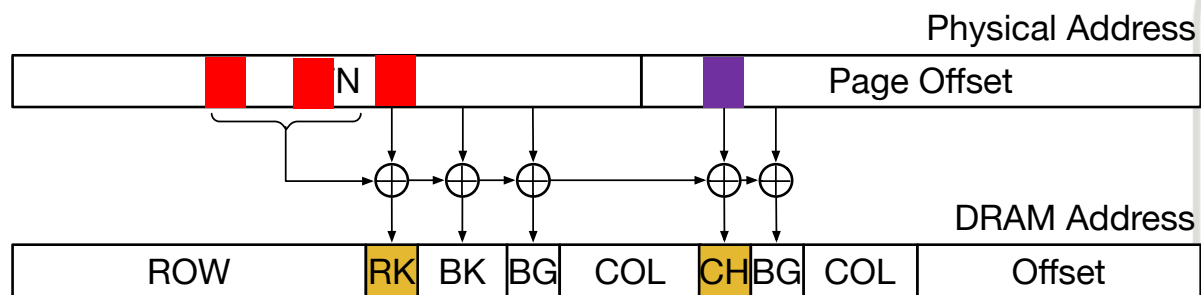
NDA Workloads in SVRG

- Linear algebra kernels
  - Streaming access pattern
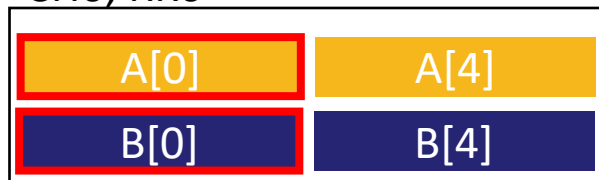 - Element-wise operations
  - Deterministic execution flow

Operations	Description	Operations	Description
AXPBY	$\vec{z} = \alpha\vec{x} + \beta\vec{y}$	DOT	$c = \vec{x} \cdot \vec{y}$
AXPBYPCZ	$\vec{w} = \alpha\vec{x} + \beta\vec{y} + \gamma\vec{z}$	NRM2	$c = \sqrt{\vec{x} \cdot \vec{x}}$
AXPY	$\vec{y} = \alpha\vec{y} + \vec{x}$	SCAL	$\vec{x} = \alpha\vec{x}$
COPY	$\vec{y} = \vec{x}$	GEMV	$\vec{y} = A\vec{x}$
XMY	$\vec{z} = \vec{x} \odot \vec{y}$		

Solution 1: ADM-Aware Frame Coloring

Physical Address

| | | | | N | | | | | | | Page Offset |

DRAM Address

| ROW | RK | BK | BG | COL | CH | BG | COL | Offset |

■ Colored by the OS

■ Alignment + Same access pattern

CH0, RK0

A[0]	A[4]
B[0]	B[4]

CH1, RK0

A[1]	A[5]
B[1]	B[5]

CH0, RK1

A[2]	A[6]
B[2]	B[6]

CH1, RK1

A[3]	A[7]
B[3]	B[7]

- Satisfies NDA locality
- No data reorganization
- No ADM change

Solution 2: Replicated FSMs