ACES: Accelerating Sparse Matrix Multiplication with Adaptive Execution Flow and Concurrency-Aware Cache Optimizations

Xiaoyang Lu*, Boyu Long*, Xiaoming Chen, Yinhe Han, Xian-He Sun

ILLINOIS TECH

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

中国科学院大学
University of Chinese Academy of Sciences

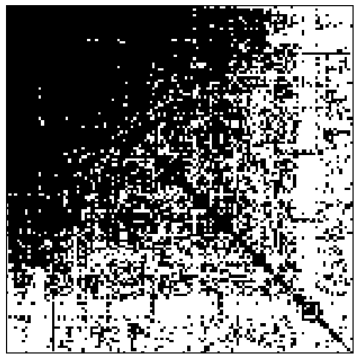
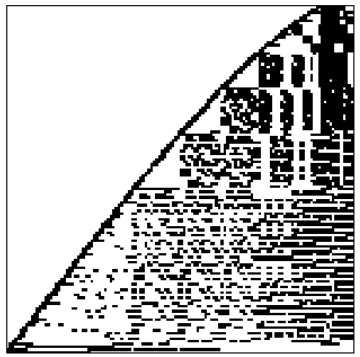
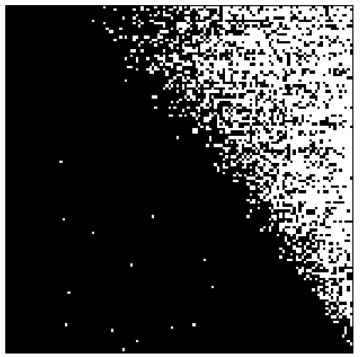
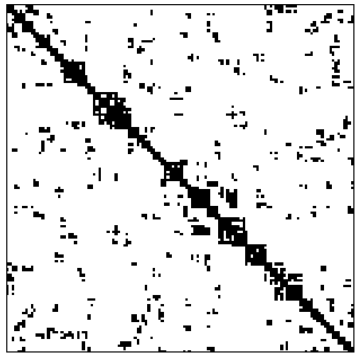
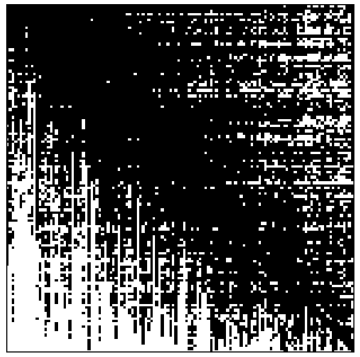
SpMM and Sparse Patterns

SpMM is widely used in machine learning and computation fields

- Applications include sparse/compressed deep neural networks, sparse linear algebra, and tensor algebra…
- There is an increasing demand for higher performance and efficiency in SpMM

Sparse Matrices have various sparse patterns

- Various matrix sizes, densities, and distribution of non-zeros
- Significant challenges for conventional cache-based computing architectures

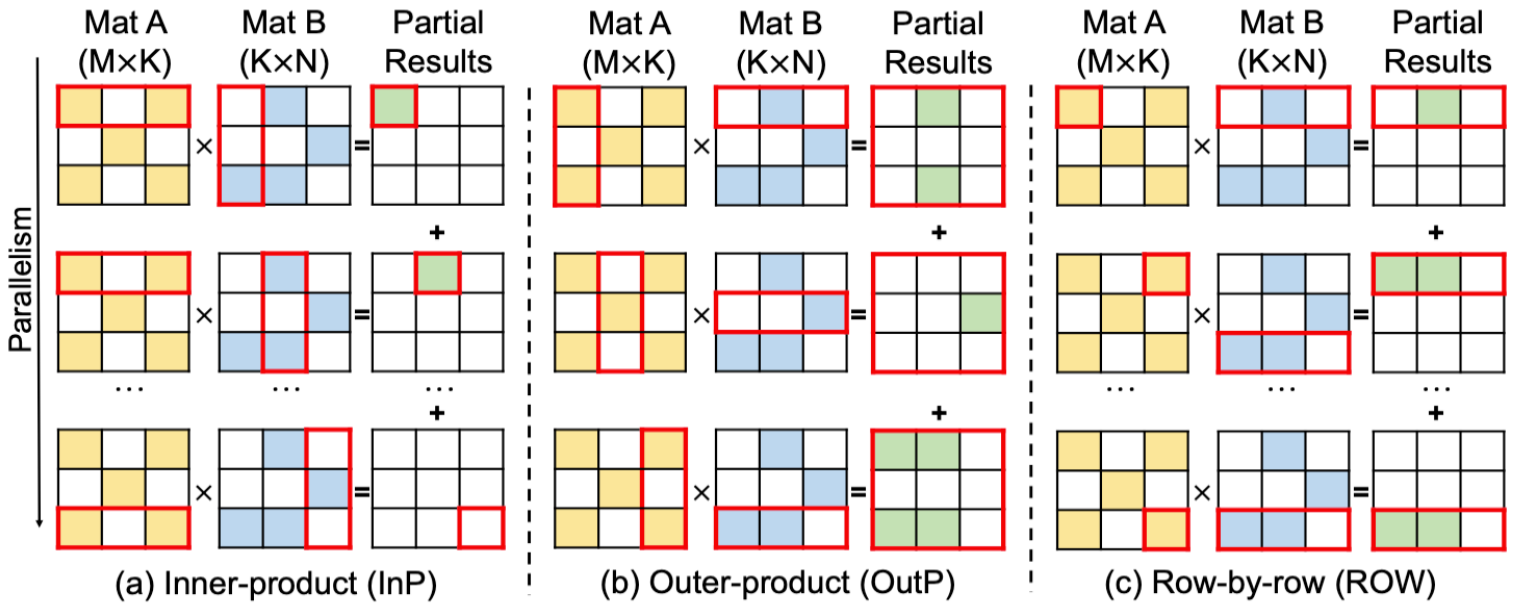
Matrices from SuiteSparse matrix collection

ACES - Motivation

Three common limitations faced by SpMM accelerators:

1 **Tradition: Fixed Execution Flow**

- The efficiency of each execution flow is determined by sparse patterns
- There is inconsistent performance across different sparse matrices



	InP	OutP	ROW
Input Reuse (Mat B)	✗	✓	✗
Output Reuse (Mat C)	✓	✗	✓
Index Intersection	✗	✓	✓
Synchronization	✓	✗	✗
Partial Result	Scalar	Matrix	Vector

An Adaptive Execution Flow is Needed

ACES - Motivation

Three common limitations faced by SpMM accelerators:

2 **Tradition: Overlook the Importance of Concurrency**

- Only focuses on reducing the number of cache misses
- SpMM operations often lead to concurrent cache line demands
- Even a single cache miss can stall the processing chain

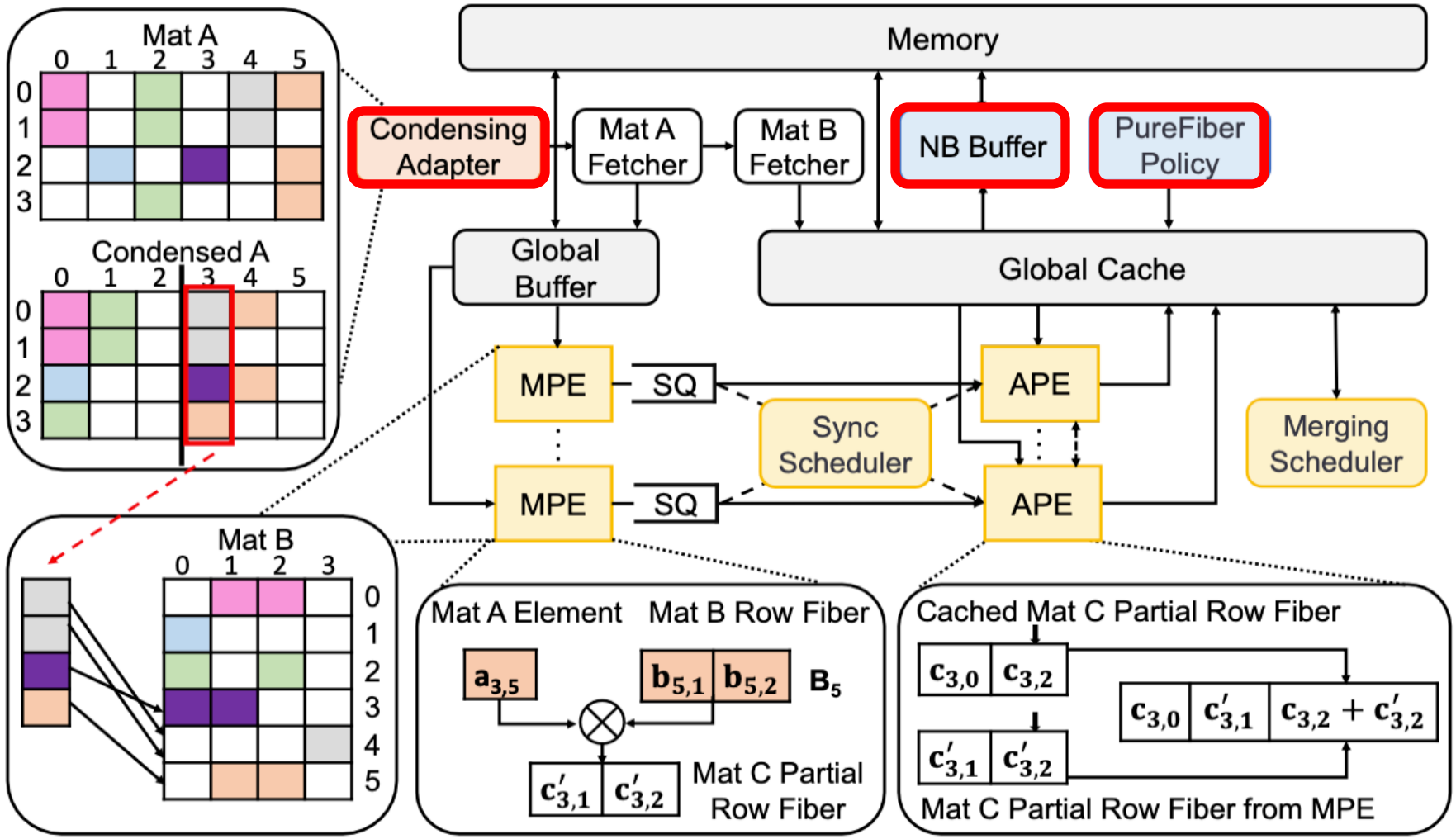
Handle the Concurrent Access Demands of SpMM

3 **Tradition: On-Chip Cache does not Incorporate Non-Blocking Features**

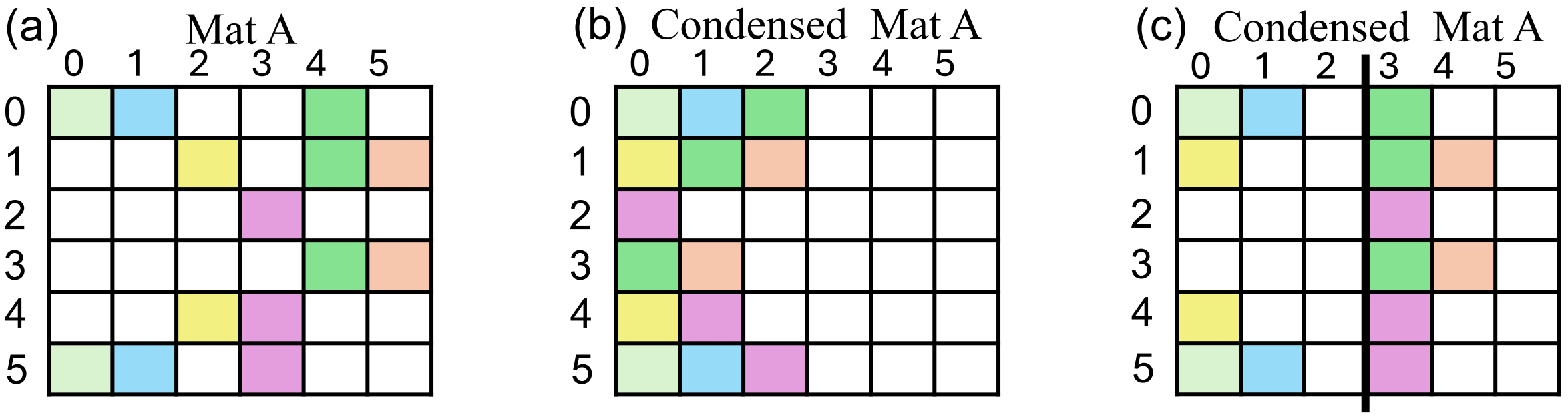
- A single cache miss causes delays in subsequent accesses

Non-Blocking Cache Design in Accelerator

ACES - Overview



ACES - Adaptive Execution Flow



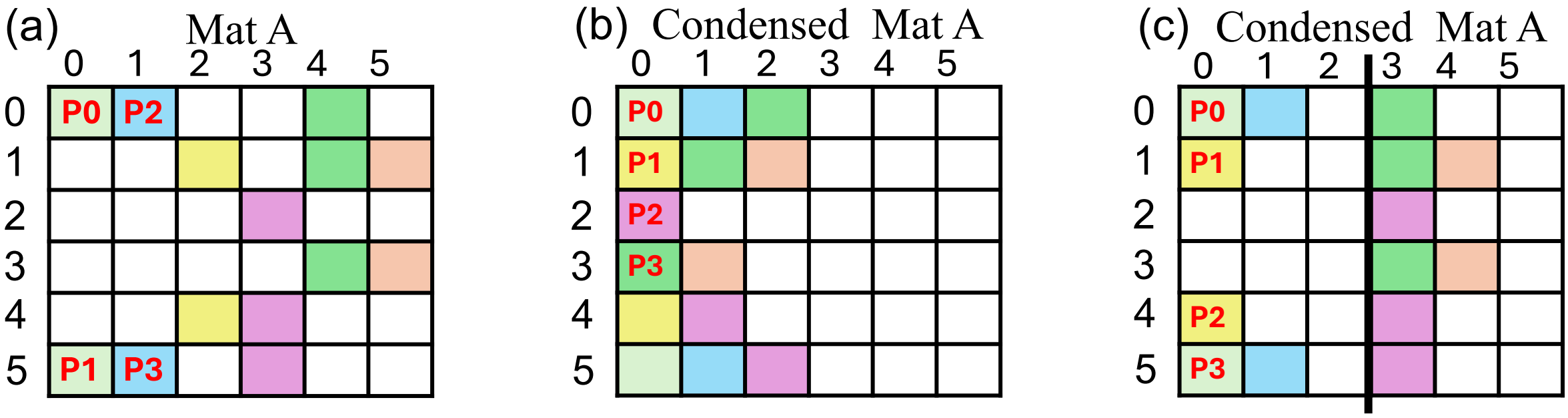
w/o Condensing

Aggressive Condensing

Moderate Condensing

Condensing degree impacts the execution flow of SpMM

ACES - Adaptive Execution Flow



w/o Condensing | Aggressive Condensing | Moderate Condensing

Condensing degree impacts the execution flow of SpMM

- Consider four processing elements, each performing scalar-vector multiplication
- Each element (scalar) is taken from Matrix A and is multiplied with the corresponding row (vector) of Matrix B

ACES - Adaptive Execution Flow

Detect Sparse Patterns + Select Condensing Degrees

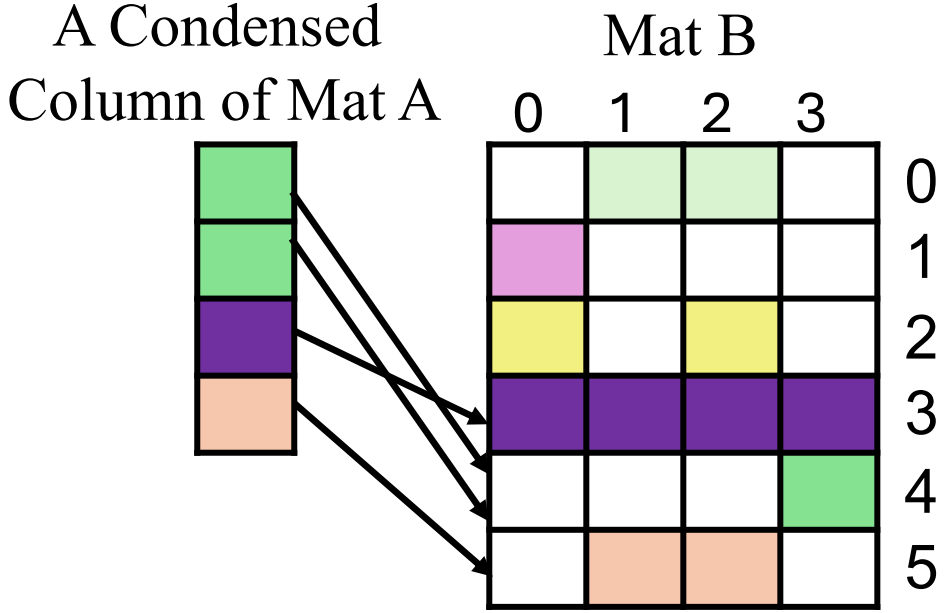
- **Detect Sparse Patterns**
  - Indicate sparse pattern changes by row length variations (SPADA [ASPLOS'23])
  - Adjacent rows with similar distributions of non-zero elements tend to have a stable row length (number of non-zero elements)
  - Partition rows into bands based on changes in row length
  - Several bands: rows in the same band have a similar sparse pattern

ACES - Adaptive Execution Flow

Detect Sparse Patterns + Select Condensing Degrees

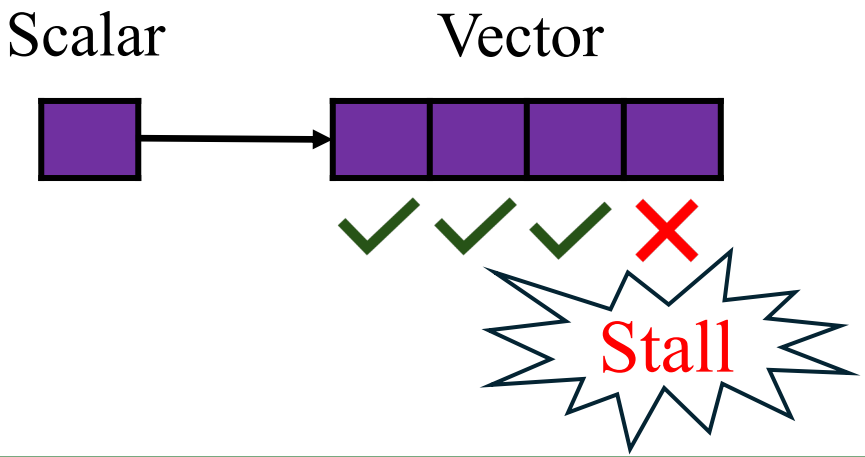
- **Select Condensing Degrees**
  - For a large band:
    - Determine the optimal condensing degree via a sampling phase
    - Conduct three sample passes, one for each condensing degree
    - Choose the best condensing degree for the remaining rows in the band
  - For a small band:
    - Apply a moderate condensing degree directly

ACES - Concurrency-Aware Cache Replacement (PureFiber)



Locality-Awareness:

- Once the condensation is determined, the demand order of the rows in Matrix B is also determined
- Use the **Next Request Distance (RD)** to capture the **reuse** distance of the rows



Concurrency-Awareness:

- A single miss still stalls the scalar-vector multiplication process
- Use **Fiber Density (FD)** to capture the number of cache lines in the corresponding row
- FD is an indicator of **potential concurrent accesses**

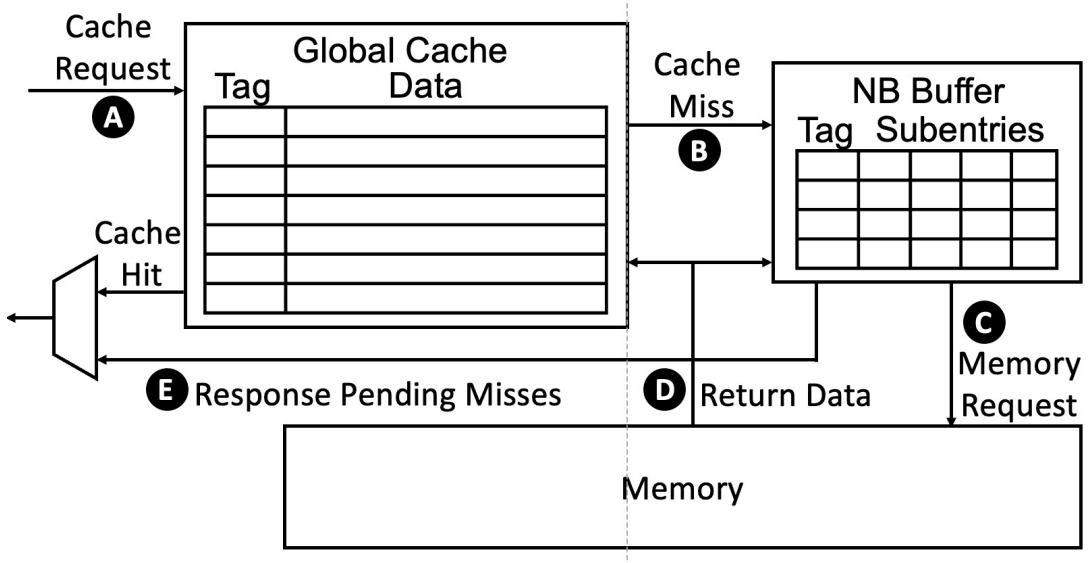
ACES - Concurrency-Aware Cache Replacement (PureFiber)

PureFiber Cache Replacement Policy

- **Pure Fiber** - allows all cache lines of a row to be accessed concurrently without any cache misses
- Aim to achieve a high number of **Pure Fibers**
- Select the cache line with the **highest combined sum of RD and FD** for eviction
- Consider both **data locality and concurrency**

ACES - Non-Blocking (NB) Buffer

- Manage concurrent cache miss accesses
- When the global cache has a miss, instead of stalling like a blocking cache, it stores the miss information in the NB buffer
- The NB buffer initiates a fetch for the missing data from the main memory
- **The NB buffer handles multiple outstanding data requests concurrently, allowing the cache to issue new memory requests even when previous ones are still being serviced**



Methodology

Benchmark

- SuiteSparse

Workload	Density	Workload	Density
2cubes_sphere (cs)	1.6e-04	offshore (of)	6.3e-05
amazon0312 (az)	2.0e-05	p2p-Gnutella31 (pg)	3.8e-05
ca-CondMat (cc)	3.5e-04	patents_main (pm)	9.7e-06
cage12 (cg)	1.2e-04	poisson3Da (p3)	1.9e-03
cop20k_A (ca)	1.8e-04	roadNet-CA (rc)	1.4e-06
email-Enron (ee)	2.7e-04	scircuit (sc)	3.3e-05
filter3D (f3)	2.4e-04	web-Google (wg)	6.1e-06
m133-b3 (mb)	2.0e-05	webbase-1M (w1)	3.1e-06
mario002 (m2)	1.4e-05	wiki-Vote (wv)	1.5e-03

Baselines

- SIGMA [HPCA'20]: Inner-product
- SpArch [HPCA'20]: Outer-product
- SPADA [ASPLOS'22]: Adaptive

Configuration

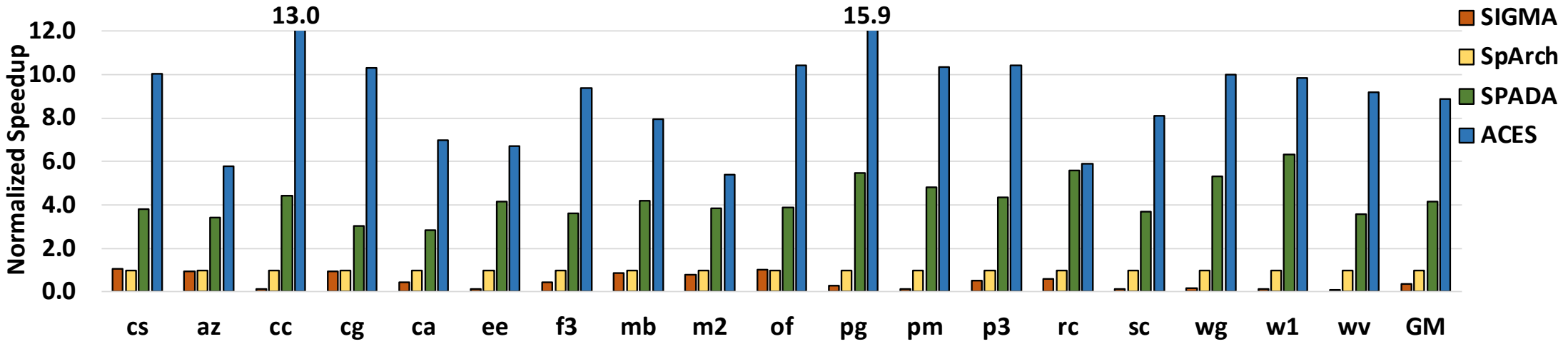
MPEs	16 MPEs (multipliers); 1 GHz
APEs	16 APEs (merger); 1 GHz
SQs	16 SQs, 2 KB per queue
Global Buffer	0.5 KB, 32-entry buffer
Global Cache	1 MB, 16 banks, 16-way associative
Crossbar	16×16 and 16×16, swizzle-switch based
NB buffer	0.5 KB, 64 subentries
Memory	128 GB/s, 16 64-bit HBM channels, 8GB/s per channel

Overhead

- Area: 3.5 mm^2
- Power: 2.8 W

ACES - Performance

Comparison among SIGMA, SpArch, SPADA, and ACES



- **ACES** consistently provides **optimal** performance across **all** workloads
- **25.5×** over SIGMA, **8.9×** over SpArch, and **2.1×** over SPADA

ACES - Performance

ACES vs. ACES-LRU

ACES-LRU:

- Operates with the same workflow as ACES
- Employs an LRU (Least Recently Used) cache replacement policy

ACES shows a **15.9%** improvement over **ACES-LRU**

ACES vs. ACES w/o NB buffer

ACES w/o NB buffer:

- Operates with the same workflow as ACES
- Does not integrate with the NB buffer

ACES shows a **35.8%** improvement over **ACES w/o NB buffer**

ACES - Summary

ACES features an **adaptive execution flow** that dynamically adjusts to diverse sparse patterns

ACES incorporates **locality-concurrency co-optimizations** within the global cache

ACES integrates a **non-blocking buffer** with the global cache to enhance concurrency

ACES **outperforms** state-of-the-art SpMM accelerators

ACES: Accelerating Sparse Matrix Multiplication with Adaptive Execution Flow and Concurrency-Aware Cache Optimizations

Xiaoyang Lu*, Boyu Long*, Xiaoming Chen, Yinhe Han, Xian-He Sun

xlu40@hawk.iit.edu

ILLINOIS TECH

INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

University of Chinese Academy of Sciences