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

Xiaoyang Lu*1, Boyu Long*2,3, Xiaoming Chen2, Yinhe Han2, Xian-He Sun1
1Illinois Institute of Technology, 2Chinese Academy of Sciences, 3University of Chinese Academy of Sciences

Sparse Matrix-Matrix Multiplication (SpMM)

SpMM is widely used in machine learning and computation fields
- 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

Limitations of Current SpMM Accelerators

There are three common limitations faced by current SpMM accelerators:
- Fixed Execution Flow: A fixed execution flow is hard to adapt different sparse patterns
- Overlooking the Importance of Concurrency: SpMM operations often lead to concurrent cache line demands; even a single cache miss can stall the processing chain
- On-Chip Cache does not Incorporate Non-Blocking Features: A single cache miss causes delays in subsequent accesses

Our Solution: ACES

Adaptive Execution Flow

Detect Sparse Patterns:
- 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
- Rows in the same band have a similar sparse pattern

Select Condensing Degrees:
- Large band - Determine the optimal condensing degree via a sampling phase
- Small band - Apply a moderate condensing degree directly

Concurrency-Aware Cache Replacement

- Use the Next Request Distance (RD) to capture the reuse distance of the rows
- Use Fiber Density (FD) to capture the number of cache lines in the corresponding row
- Select the cache line with the highest combined sum of RD and FD for eviction
- Allow all cache lines of a row to be accessed concurrently without any cache misses

Non-Blocking (NB) Buffer

- Handle multiple outstanding data requests concurrently
- Allow the cache to issue new memory requests even when previous ones are still being serviced

Results

ACES outperforms all other SpMM accelerators
- 25.5× over SIGMA, 8.9× over SpArch, and 2.1× over SPADA

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

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
- Condensing degree impacts the execution flow of SpMM

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

Xiaoyang Lu*1, Boyu Long*2,3, Xiaoming Chen2, Yinhe Han2, Xian-He Sun1
1Illinois Institute of Technology, 2Chinese Academy of Sciences, 3University of Chinese Academy of Sciences

Sparse Matrix-Matrix Multiplication (SpMM)

SpMM is widely used in machine learning and computation fields
- 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

Limitations of Current SpMM Accelerators

There are three common limitations faced by current SpMM accelerators:
- Fixed Execution Flow: A fixed execution flow is hard to adapt different sparse patterns
- Overlooking the Importance of Concurrency: SpMM operations often lead to concurrent cache line demands; even a single cache miss can stall the processing chain
- On-Chip Cache does not Incorporate Non-Blocking Features: A single cache miss causes delays in subsequent accesses

Our Solution: ACES

Adaptive Execution Flow

Detect Sparse Patterns:
- 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
- Rows in the same band have a similar sparse pattern

Select Condensing Degrees:
- Large band - Determine the optimal condensing degree via a sampling phase
- Small band - Apply a moderate condensing degree directly

Concurrency-Aware Cache Replacement

- Use the Next Request Distance (RD) to capture the reuse distance of the rows
- Use Fiber Density (FD) to capture the number of cache lines in the corresponding row
- Select the cache line with the highest combined sum of RD and FD for eviction
- Allow all cache lines of a row to be accessed concurrently without any cache misses

Non-Blocking (NB) Buffer

- Handle multiple outstanding data requests concurrently
- Allow the cache to issue new memory requests even when previous ones are still being serviced

Results

ACES outperforms all other SpMM accelerators
- 25.5× over SIGMA, 8.9× over SpArch, and 2.1× over SPADA

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

Xiaoyang Lu*1, Boyu Long*2,3, Xiaoming Chen2, Yinhe Han2, Xian-He Sun1
1Illinois Institute of Technology, 2Chinese Academy of Sciences, 3University of Chinese Academy of Sciences

Sparse Matrix-Matrix Multiplication (SpMM)

SpMM is widely used in machine learning and computation fields
- 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

Limitations of Current SpMM Accelerators

There are three common limitations faced by current SpMM accelerators:
- Fixed Execution Flow: A fixed execution flow is hard to adapt different sparse patterns
- Overlooking the Importance of Concurrency: SpMM operations often lead to concurrent cache line demands; even a single cache miss can stall the processing chain
- On-Chip Cache does not Incorporate Non-Blocking Features: A single cache miss causes delays in subsequent accesses

Our Solution: ACES

Adaptive Execution Flow

Detect Sparse Patterns:
- 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
- Rows in the same band have a similar sparse pattern

Select Condensing Degrees:
- Large band - Determine the optimal condensing degree via a sampling phase
- Small band - Apply a moderate condensing degree directly

Concurrency-Aware Cache Replacement

- Use the Next Request Distance (RD) to capture the reuse distance of the rows
- Use Fiber Density (FD) to capture the number of cache lines in the corresponding row
- Select the cache line with the highest combined sum of RD and FD for eviction
- Allow all cache lines of a row to be accessed concurrently without any cache misses

Non-Blocking (NB) Buffer

- Handle multiple outstanding data requests concurrently
- Allow the cache to issue new memory requests even when previous ones are still being serviced

Results

ACES outperforms all other SpMM accelerators
- 25.5× over SIGMA, 8.9× over SpArch, and 2.1× over SPADA