Bandwidth-Optimized Algorithms for Sparse Matrix-Matrix Multiplication

Based on: Bandwidth Optimized Parallel Algorithm for Sparse Matrix-Matrix Multiplication using Propagation Blocking, published at **ACM SPAA 2020**

Ariful Azad, Indiana University

In collaboration with
Zhixiang Gu, Facebook Inc.
Jose Moreira, IBM Research
David Edelsohn, IBM Research

Introduction

Sparse General Matrix-Matrix Multiplication (SpGEMM)

A key kernel in GraphBLAS with many applications

- Graph analytics
  - betweenness centrality, clustering coefficients, triangle counting, colored intersection search
- Scientific computing
  - algebraic multigrid, linear solvers
- Machine learning
  - dimensionality reduction (e.g. NMF, PCA), spectral clustering and Markov clustering

Questions and Contributions

- Given two input matrices (A and B) and a given processor
  - What is best possible performance attained by any algorithm?
  - What is the best possible performance that a given algorithm can attain?
  - **We consider a Roofline model for SpGEMM to answer these questions**

- Given the observed performance from an algorithm
  - Can we explain why the best possible performance may or may not be achieved under a performance model?
  - **We explain based on bandwidth utilization**

- Can we develop an algorithm that always achieves the performance predicted by the Roofline model?
  - PB-SpGEMM: **Predictable performance by saturating memory bandwidth**

Toward A Performance Model for SpGEMM Algorithms

Goal: **Find arithmetic Intensity (AI) of SpGEMM**

- flops/bytes moved.

Compression factor (cf) = flops/nnz(C)

Assume **b bytes** (including indices) per nonzero

Best case: **All matrices are accessed exactly once**

$$AI \leq \frac{nnz(C) * cf}{[nnz(A) + nnz(B) + nnz(C)] * b} \leq \frac{cf}{b}$$

$Peak\ FLOPS \leq \beta \frac{cf}{b}$, assuming a memory-bound operation

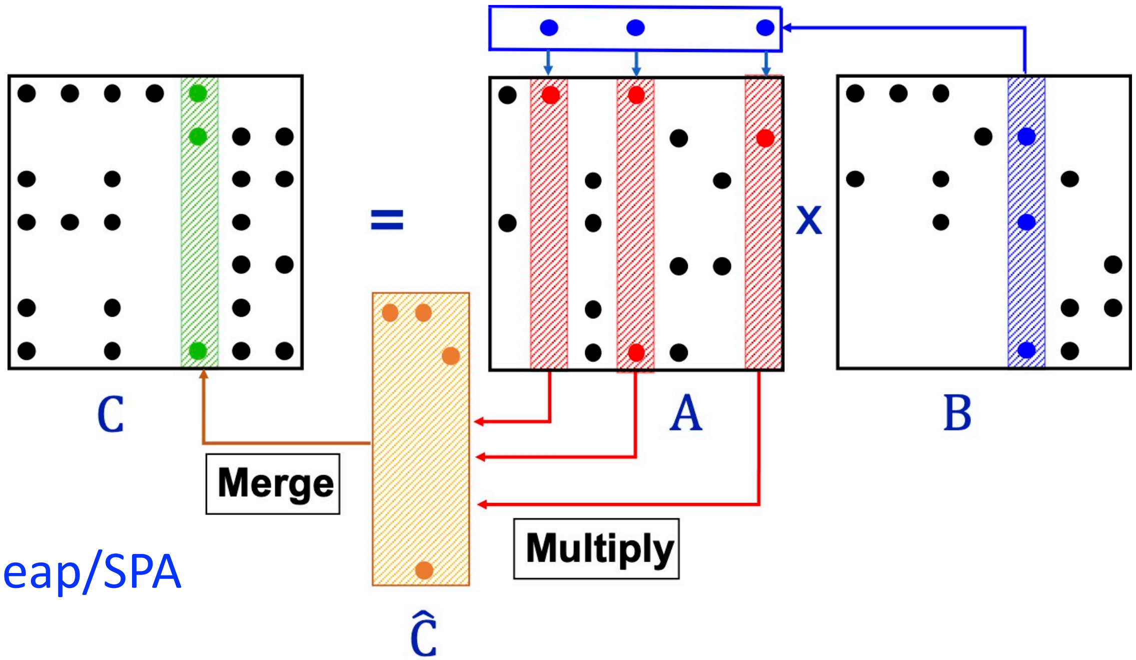
Is this a good bound?

Think random ER matrices: cf =1, let b=16 bytes, bandwidth 50GB/s

Best Attainable FLOPS : 3.1 GFLOPS.

Actual performance is much worse. **Matrices are accessed more than once**

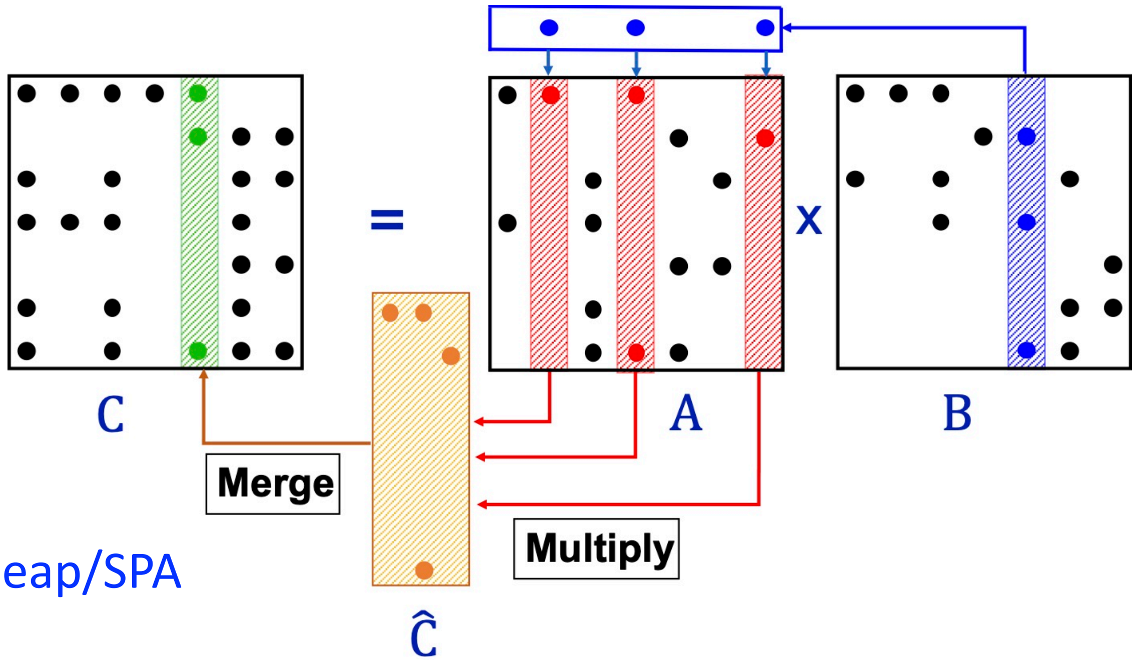
Case1: Column SpGEMM



Merge: Hash/Heap/SPA

Matrix	Access Pattern
Access of B	Stream
Access of A	Non-Stream, Accessed multiple times
Access of C	Stream

Case1: Column SpGEMM
Access-pattern-specific Performance Bounds



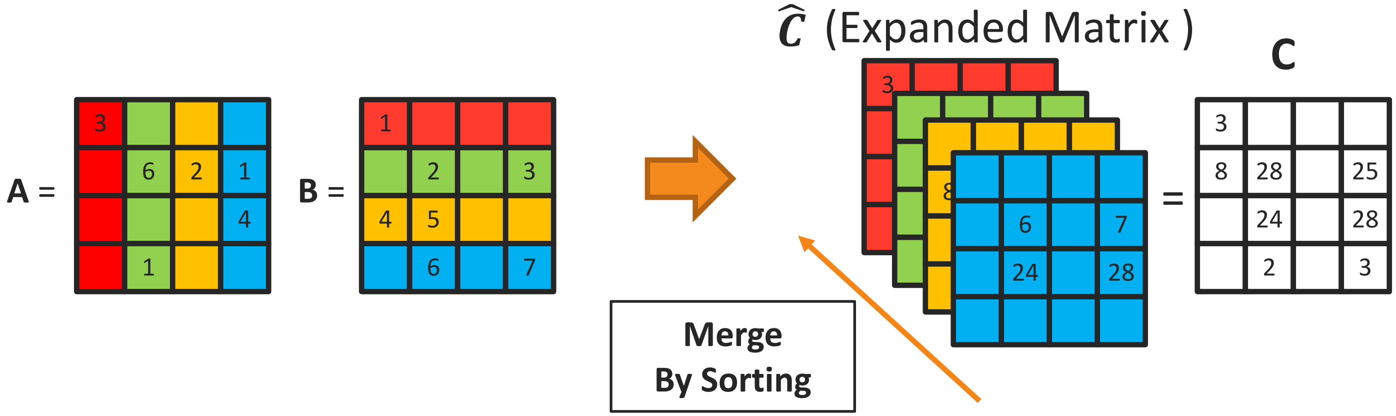
Merge: Hash/Heap/SPA

In the worst case, each column of A is accessed from memory

$$AI(Col\ SpGEMM) \geq \frac{nnz(C) * cf}{[nnz(C) * cf + nnz(B) + nnz(C)] * b}$$

$$\geq \frac{cf}{(2 + cf) * b}$$

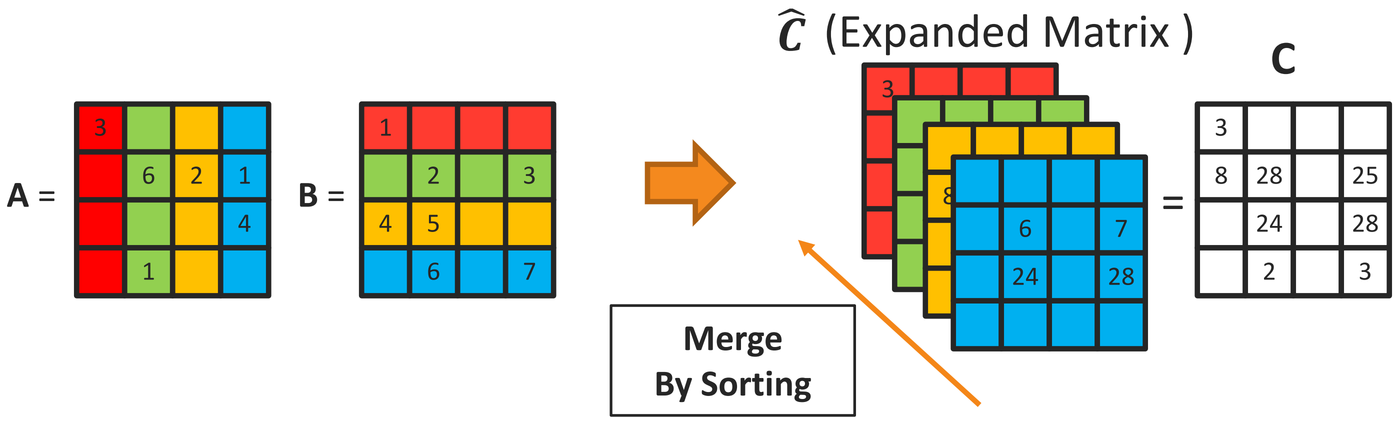
Case2: Outer-Product SpGEMM



Matrix	Access Pattern
Access of B	Stream
Access of A	Stream
Access of Ĉ	Non-Stream, Accessed multiple times
Access of C	Stream

Case2: Outer-Product SpGEMM

Access-pattern-specific Performance Bounds



$$AI(Outer\ SpGEMM) \geq \frac{nnz(C) * cf}{[nnz(A) + nnz(B) + 2 * nnz(C') + nnz(C)] * b}$$

$$= \frac{nnz(C) * cf}{[nnz(A) + nnz(B) + 2 * flops + nnz(C)] * b}$$

$$\geq \frac{cf}{(3 + cf) * b}$$

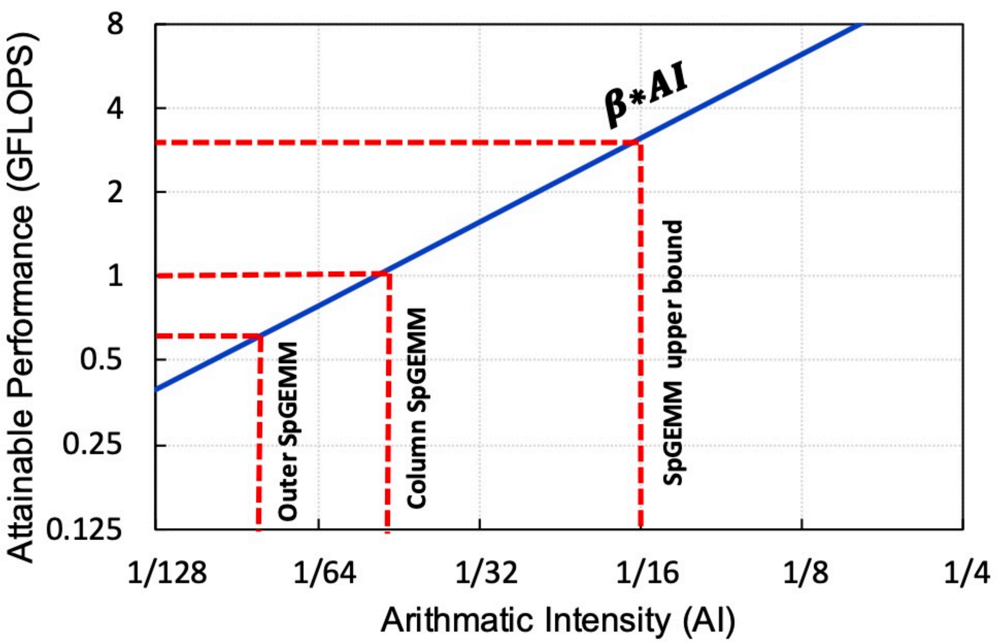
Roofline Performance Model for SpGEMM Algorithms

Consider Erdos-Renyi model (*cf≈1*) and using tuple (*rowid*, *colid*, *val*) to represent non-zeros (*b=16 bytes*)

$$AI(Col\ SpGEMM) \geq \frac{1}{48}$$

$$AI(Outer\ \mathrm{SpGEMM}) \geq \frac{1}{80}$$

Assuming bandwidth(***β***) = 50GB/s

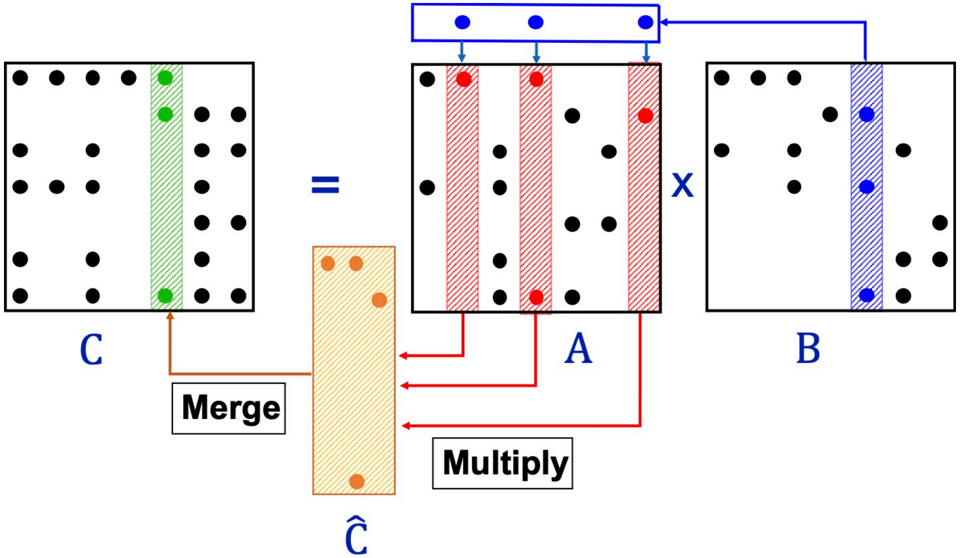


Using roofline model[1] to estimate performance when multiplying two Erdos-Renyi matrices on an Intel Skylake machine (single socket)

[1] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures

Can Existing Algorithms Achieve Performance Predicted by this model?

- Column SpGEMM:
  - Prediction for ER matrices



Expecting $FLOPS(Col\ SpGEMM) = \beta\ *\ AI(Col\ SpGEMM) \approx 1\ GFLOPS$

Getting… $FLOPS(Col\ SpGEMM) \approx 0.5\ GFLOPS$ or less

Why?

- Random memory access -> huge latency overhead.
- **It may not be possible to avoid the irregular data access problem in Column SpGEMM**

New Algorithm: PB-SpGEMM

Based on the Expand-Sort-Merge strategy

Algorithm 1: ESC-SpGEMM algorithm

Input: A , B
Output: C
$\hat{C} \leftarrow$ `Symbolic` (A, B) ▹ Create space for $\hat{C}$;
$\hat{C} \leftarrow$ `Expand` (**A**, **B**) ▹ Create unmerged tuples ;
`Sort` ($\hat{C}$) ▹ sort tuples using (rowid, colid) as keys;
C $\leftarrow$ `Compress` ($\hat{C}$) ▹ merge duplicated tuples ;

How do we expand?

Outer product formation. Streaming accesses of input matrices

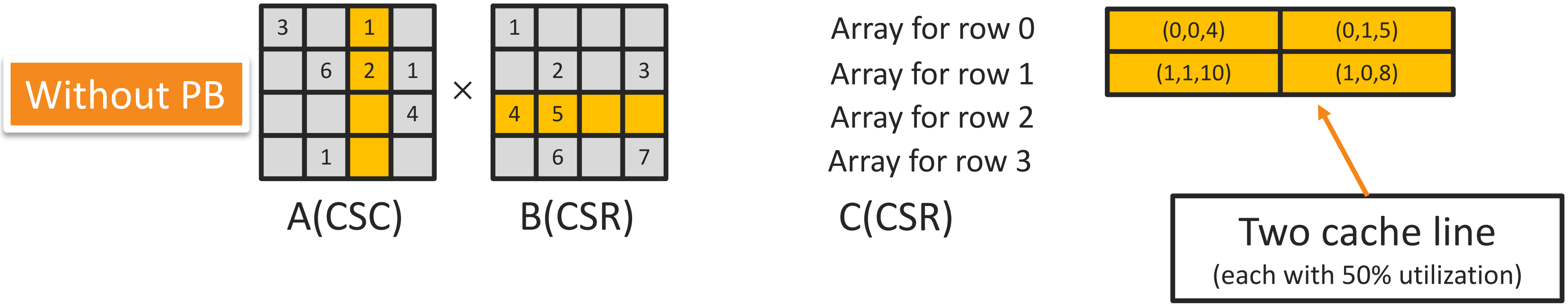
How do we organize intermediate results?

Propagation blocking (Beamer et al. IPDPS 2017 for PageRank, Azad and Buluç IPDPS 2017 for SpMSpV)

Propagation Blocking with Outer Product

Assuming:

- Cache Line = 64 bytes
- Each Tuple = 16 bytes

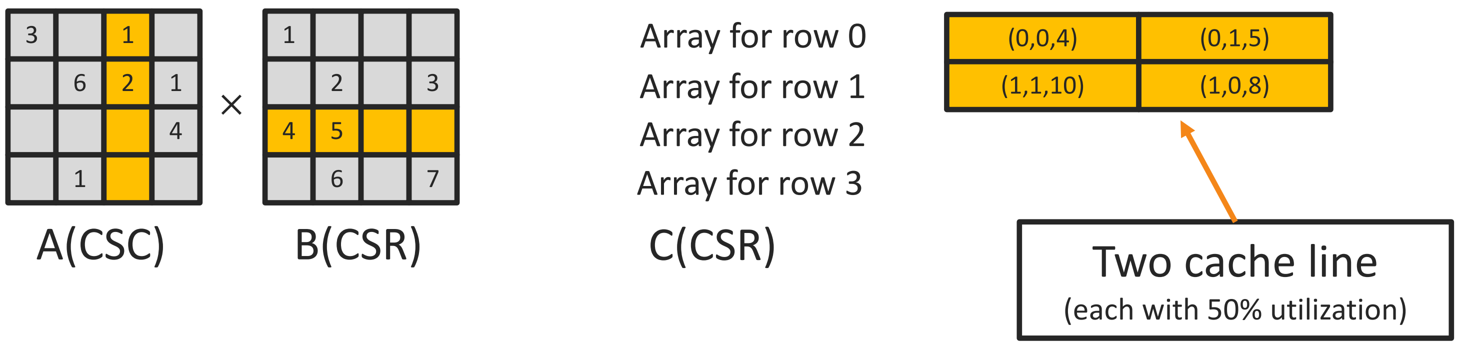


What is Propagation Blocking?

Assuming:

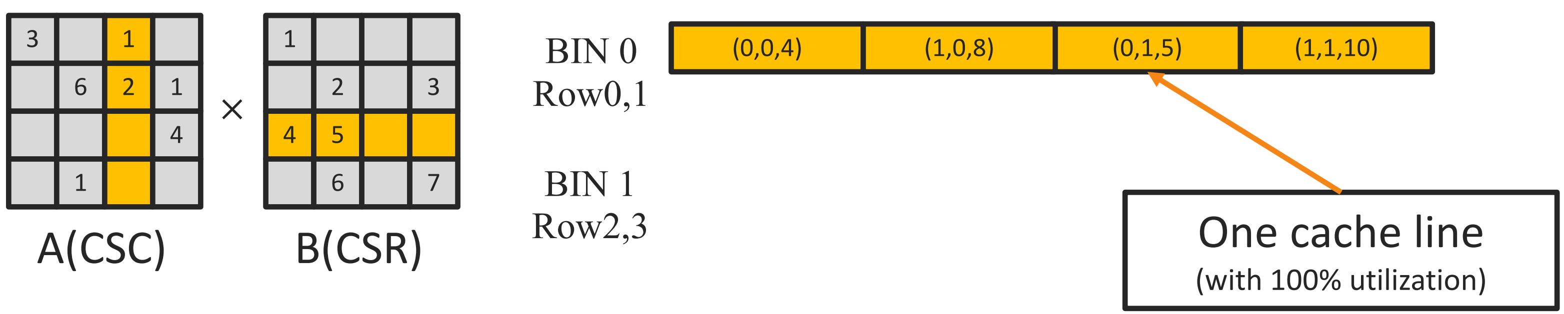
- Cache Line = 64 bytes
- Each Tuple = 16 bytes

Without PB



➢ Propagation-blocking[1]: partition the data transfers during multiplication

With PB



[1] Beamer, Asanović, Patterson: Reducing PageRank communication via propagation blocking [IPDPS 2017]

A full example of PB-SpGEMM

3 Steps in PB-SPGEMM

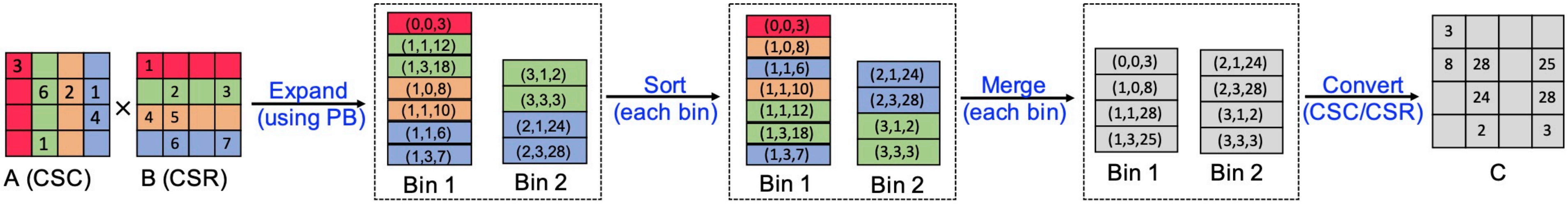


Figure: An example of PB-SpGEMM multiplying two 4×4 matrices with two bins

Number of bins is set such that each bin fits in L1/L2 cache

Sort: in cache

In-place radix sort

- Concatenate rowid and colid into an 8-byte integer key
- Adjust number of bins to make sure sorting in cache

Compress (sorted indices): in cache

PB-SpGEMM performance model

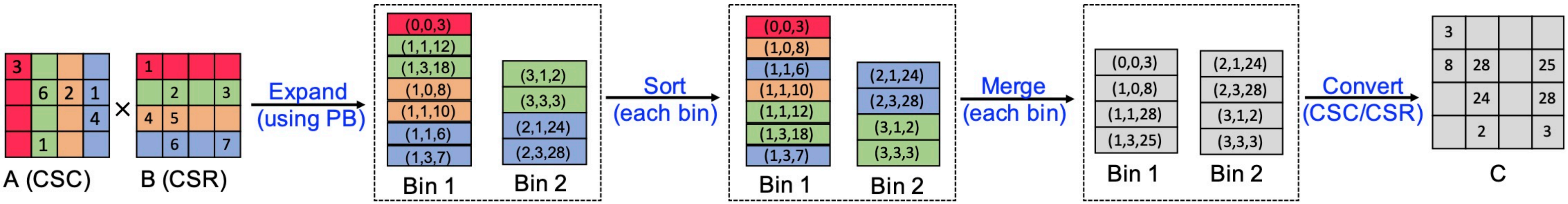


Figure: An example of PB-SpGEMM multiplying two 4×4 matrices with two bins

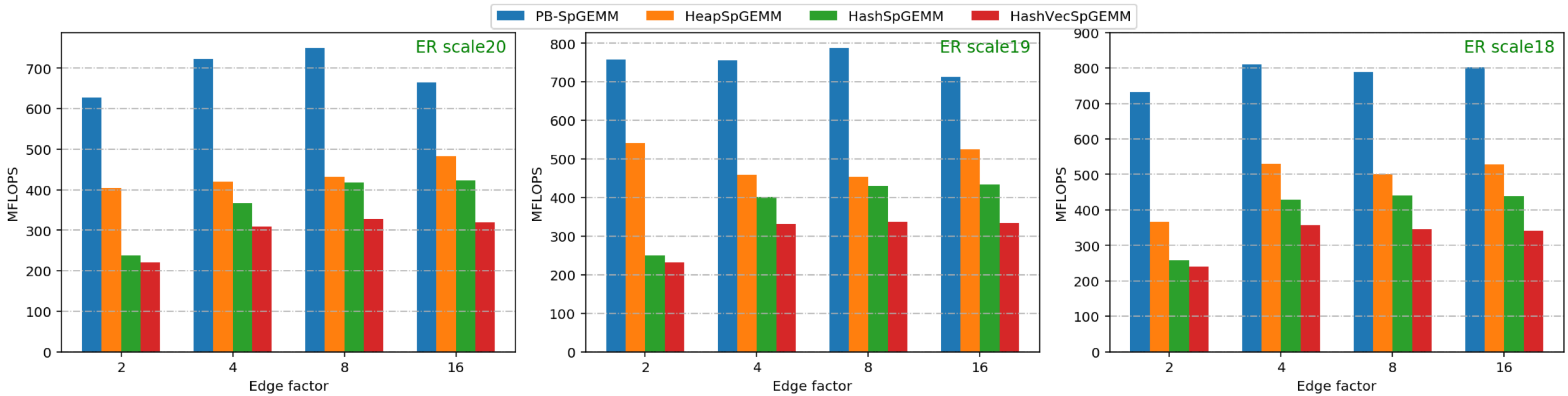
The design of PB-SpGEMM ensures **exact bound** on AI

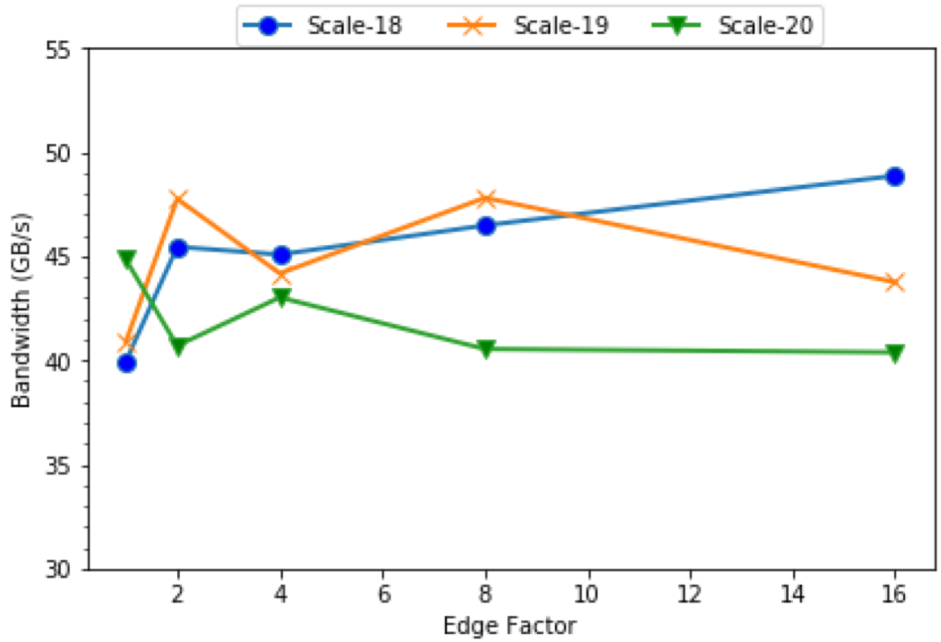
$$AI(Outer\ SpGEMM) = \frac{nnz(C) * cf}{[nnz(A) + nnz(B) + 2 * nnz(C') + nnz(C)] * b}$$

$$= \frac{nnz(C) * cf}{[nnz(A) + nnz(B) + 2 * flops + nnz(C)] * b}$$

$$= \frac{cf}{(3 + cf) * b}$$

Performance Evaluation (ER matrices on Skylake)



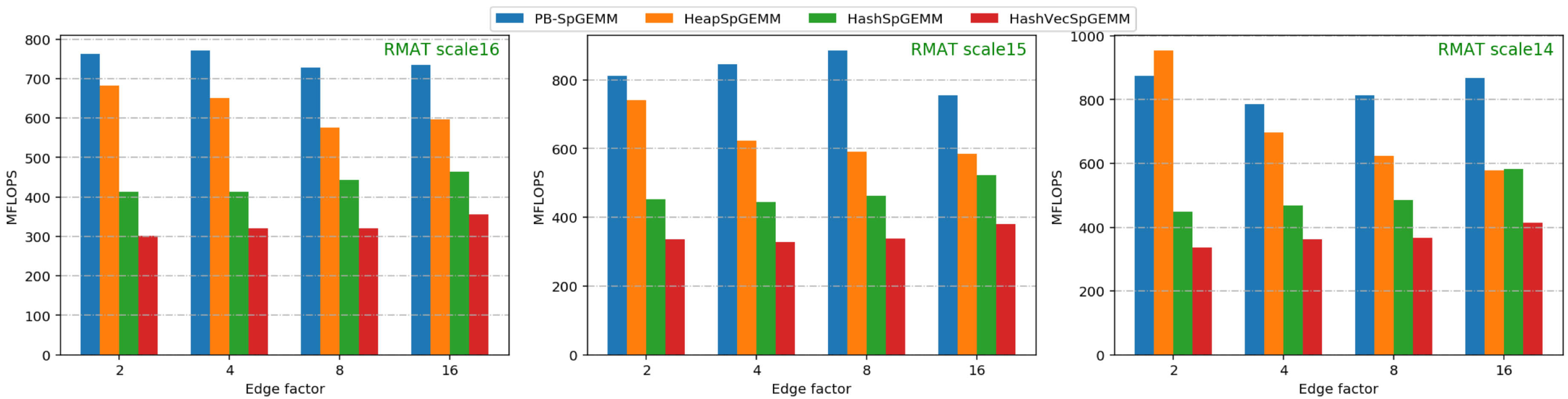


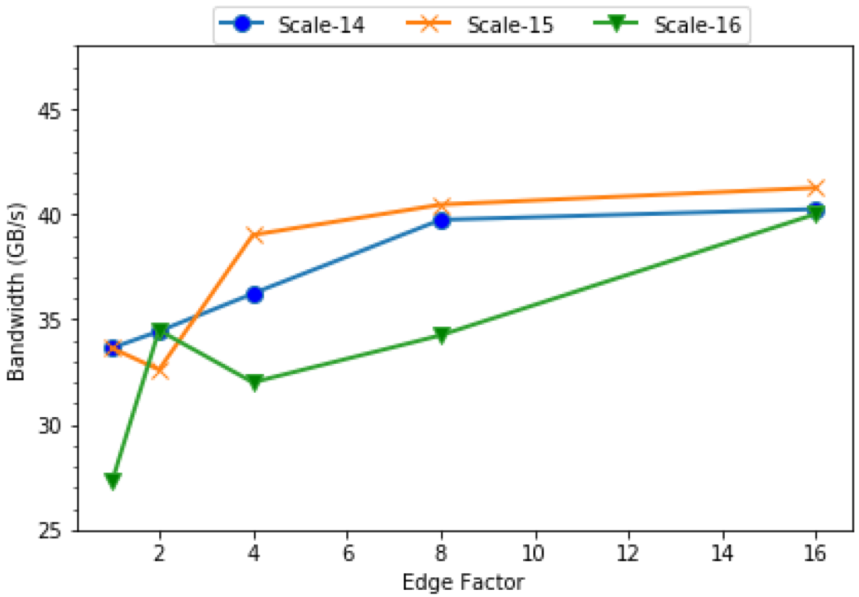
HeapSpGEMM, HashSpGEMM, HashVecSpGEMM
Column SpGEMM
Nagasaka et al. Parallel Computing, 2019

24 cores (1 socket)
50GB/s bandwidth

PB-SpGEMM approximately achieves the predicted performance

Performance Evaluation (RMAT matrices on Skylake)



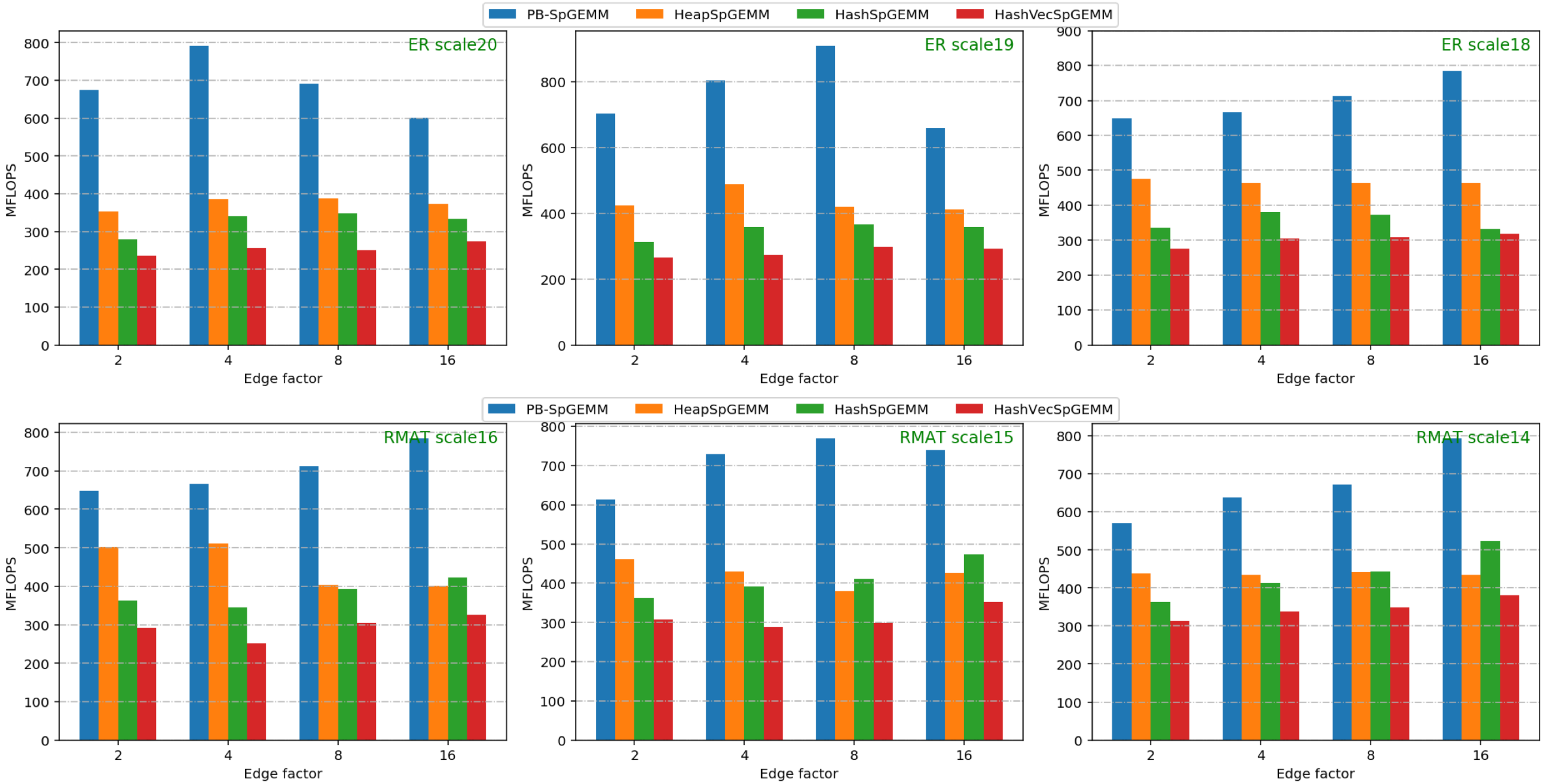


24 cores (1 socket)
50GB/s bandwidth

PB-SpGEMM approximately achieves the predicted performance (worse than ER)

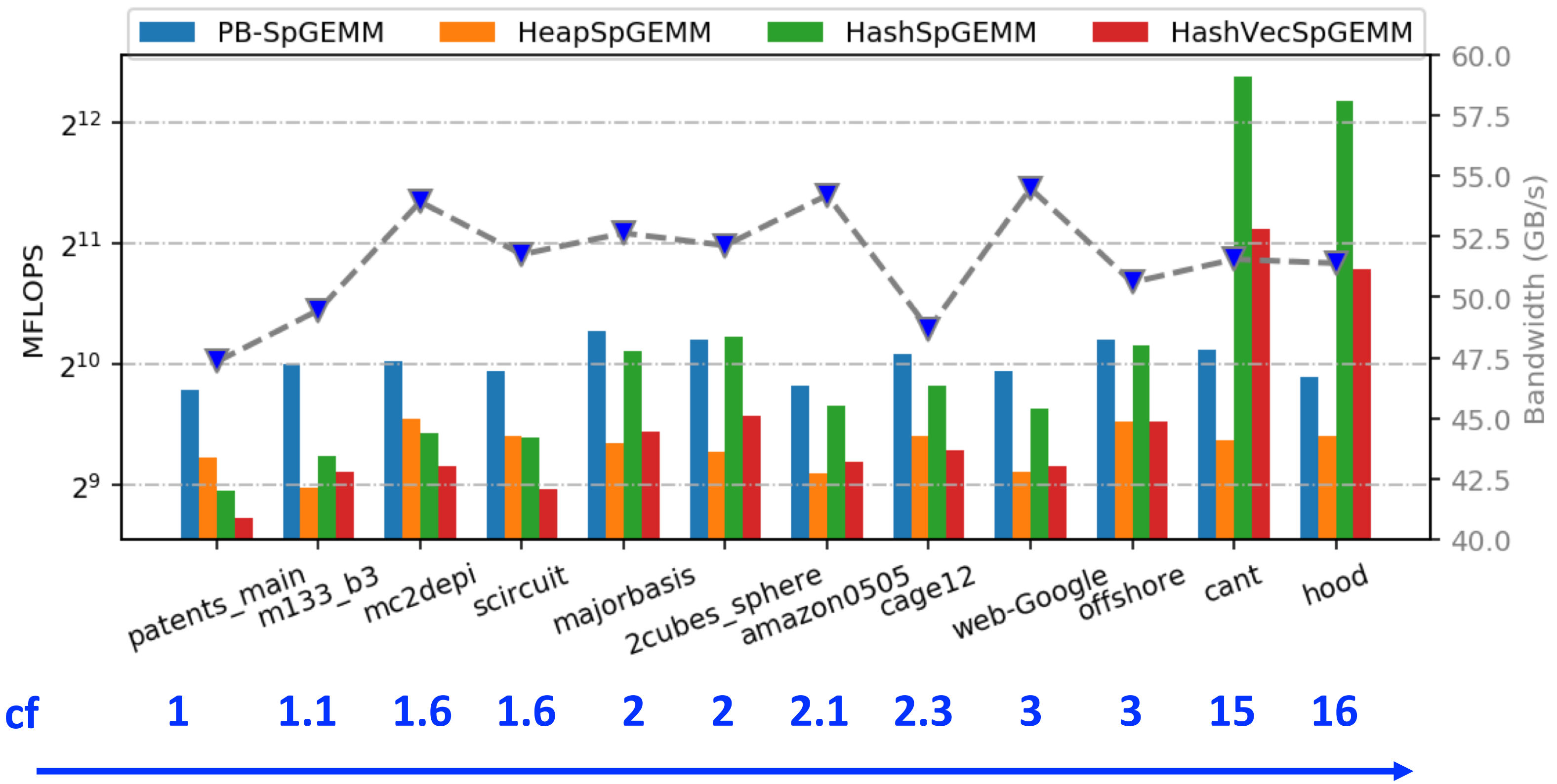
Performance Evaluation (IBM Power9)

20 cores (1 socket)
125GB/s bandwidth



Performance Evaluation

Real metrices (from the SuiteSparse Matrix Collection)



PB-SpGEMM approximately achieves the predicted performance for matrices with **low compression factors**

Limitations

High compression factor: The expanded matrix gets bigger.
PB-SpGEMM still obtains **predictable** but **poor** performance.
✓ When squaring matrices, more 90% matrices in the SuiteSparse Matrix Collection have **a compress factor of four or less**

Dual socket performance: falls well behind the model even for matrices with low compression ratio
Inter-socket bandwidth contention

Summary

- We can estimate the **arithmetic intensity** (AI) of an SpGEMM algorithm based on the compression factor of the multiplication and number of bytes needed to store each nonzero
- The peak performance (**β*AI**) can only be attained if the algorithm fully utilizes the memory bandwidth
- Column SpGEMM algorithms do not achieve the predicted performance because of irregular data accesses
- **PB-SpGEMM approximately saturates the memory bandwidth** in all of its three phases and attains performance as predicted by the Roofline model.
- PB-SpGEMM does not perform well when the compression factor is large (Hash-SpGEMM performs better in that case)