SAGE: A Storage-Based Approach for Scalable and Efficient Sparse Generalized Matrix-Matrix Multiplication

(ACM CIKM 2023)

25 October 2023

Presenter

Myung-Hwan Jang

Hanyang University

Republic of Korea

This is a joint work with Hanyang University (Myung-Hwan Jang, Yunyong Ko, Hyuck-Moo Gwon, Ikhyeon Jo, and Sang-Wook Kim) and Yonsei University (Yongjun Park)

Graph

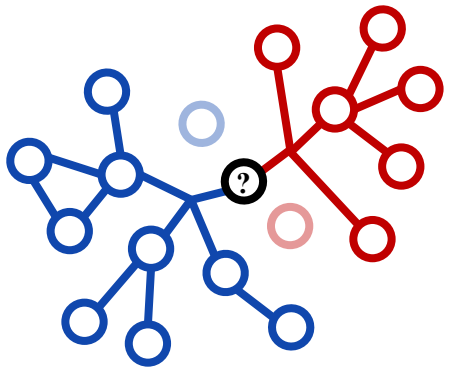
Widely-used data structure to model real-world networks

- Consisting of nodes (objects) and edges (their relationships)

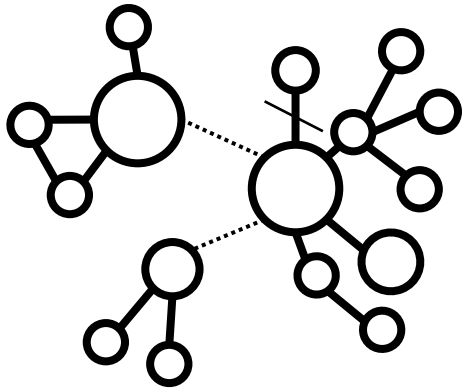
Real-world networks

Importance of Graph Analysis

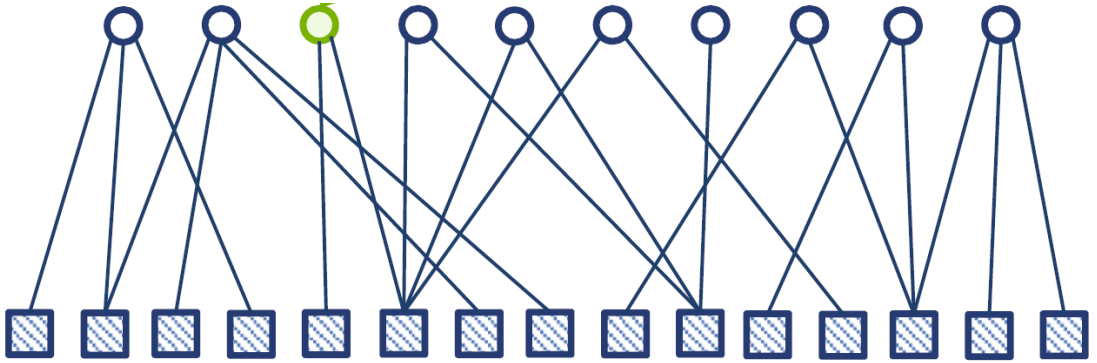
- We can obtain useful knowledge that helps improve downstream tasks



Community detection



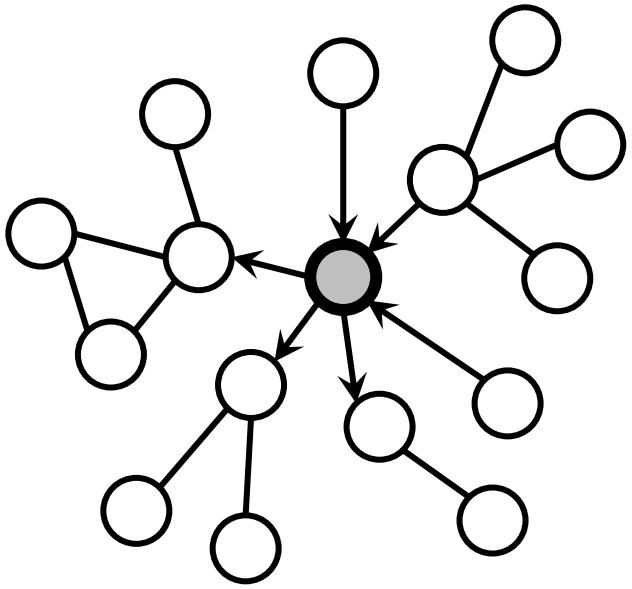
Link prediction



Recommendation systems

Matrix Representation

- A graph is generally represented as a *dense matrix*



Graph

Dst. nodes

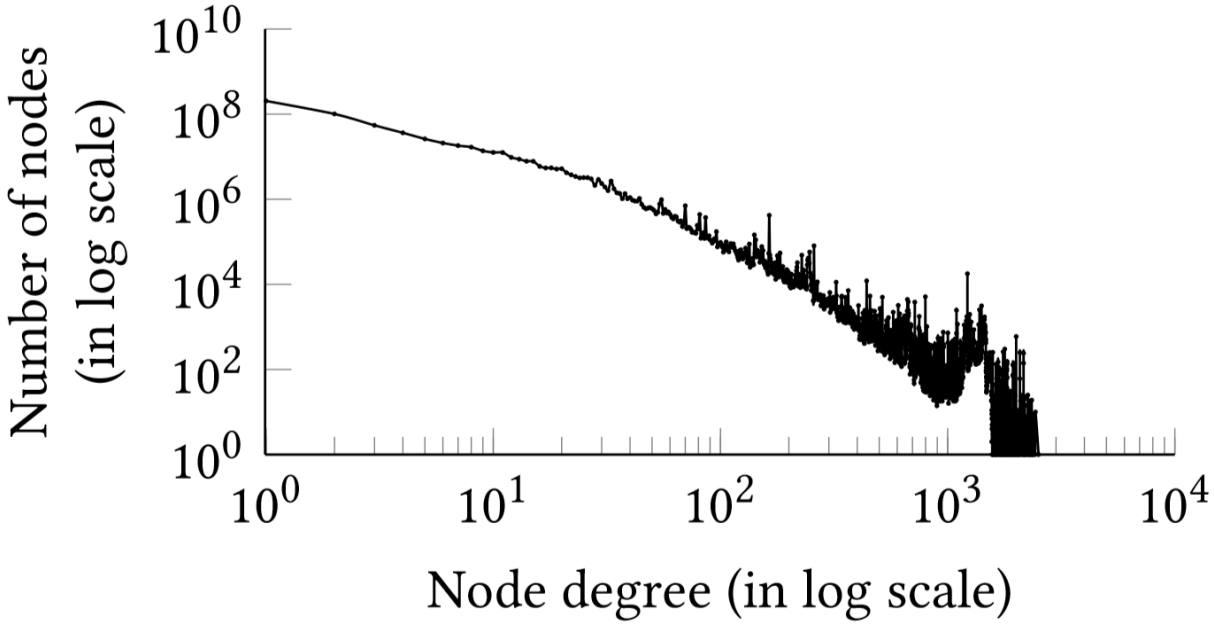
Src. nodes

0	1	1	1	0
1	0	0	0	1
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0

Matrix

Generalized Matrix-Matrix Multiplication (GEMM)

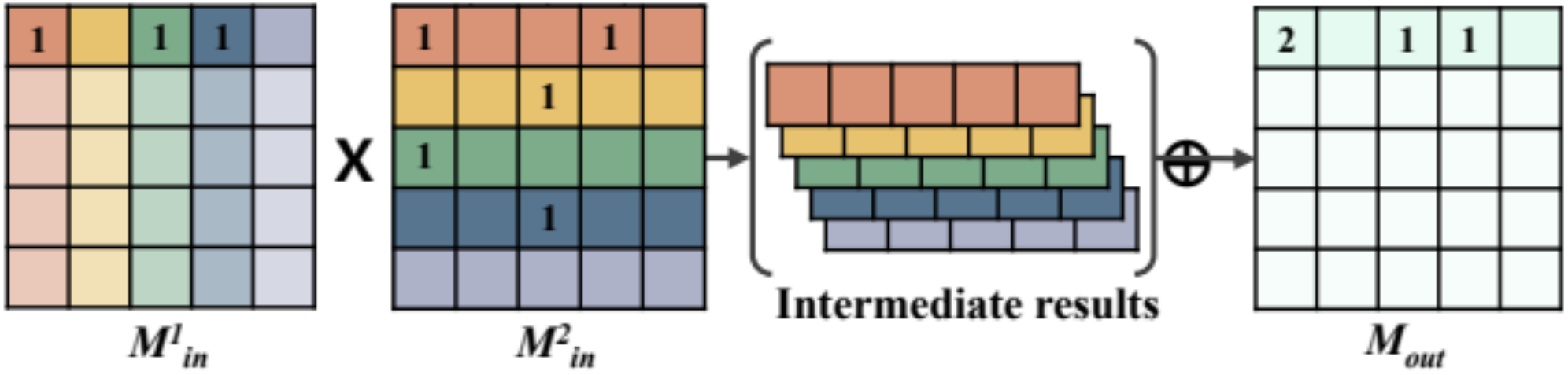
- ***One of the key operations*** **for analyzing graphs**
- **Critical challenge of GEMM**
  - Real-world graphs tend to follow a power-law degree distribution
  - GEMM often requires an unnecessarily large amount of space/computational cost



Sparse GEMM (SpGEMM)

- Represents a graph as a *sparse matrix containing only existing edges*
- Adopts *row-wise product* for SpGEMM in many cases
  - Operations of multiplication are performed in a row-wise manner

$$M_{out}(i,;) = \sum_{j=1}^{n} M_{in}^{1}(i,j) \times M_{in}^{2}(j,;),$$



Existing SpGEMM Approaches

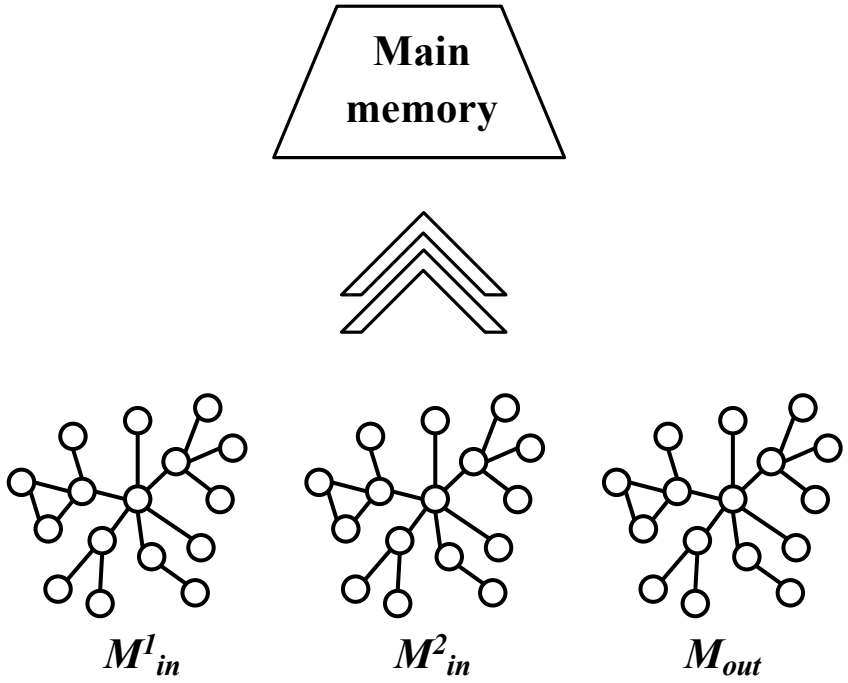
- Single-machine-based (i.e., in-memory-based) approach
- Distributed-system-based approach

A single machine

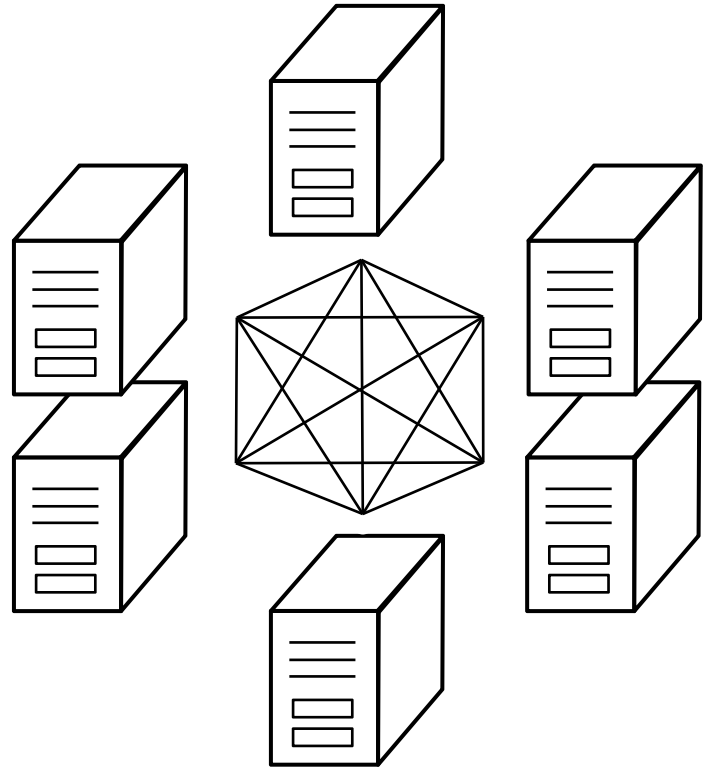
Distributed systems

Challenges with Large-Scale SpGEMM

- Limited main memory (i.e., *Not scalable*)
- Frequent network communication (i.e., *Not efficient*)



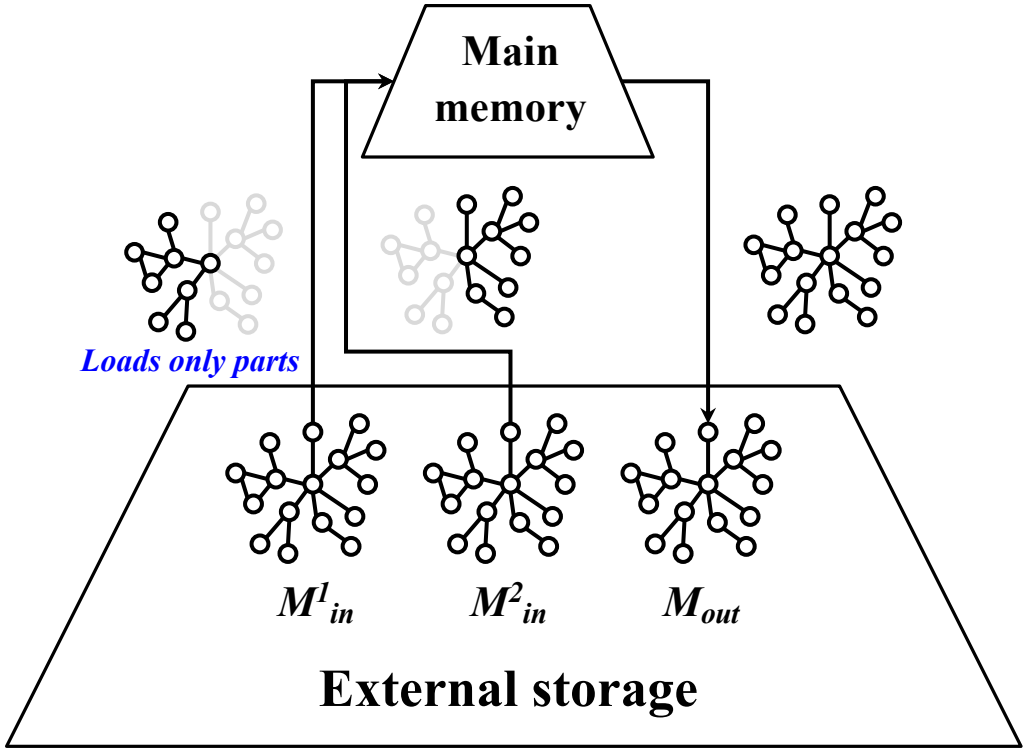
Limited main memory



Frequent network communication

Storage-Based Approach for SpGEMM (SAGE)

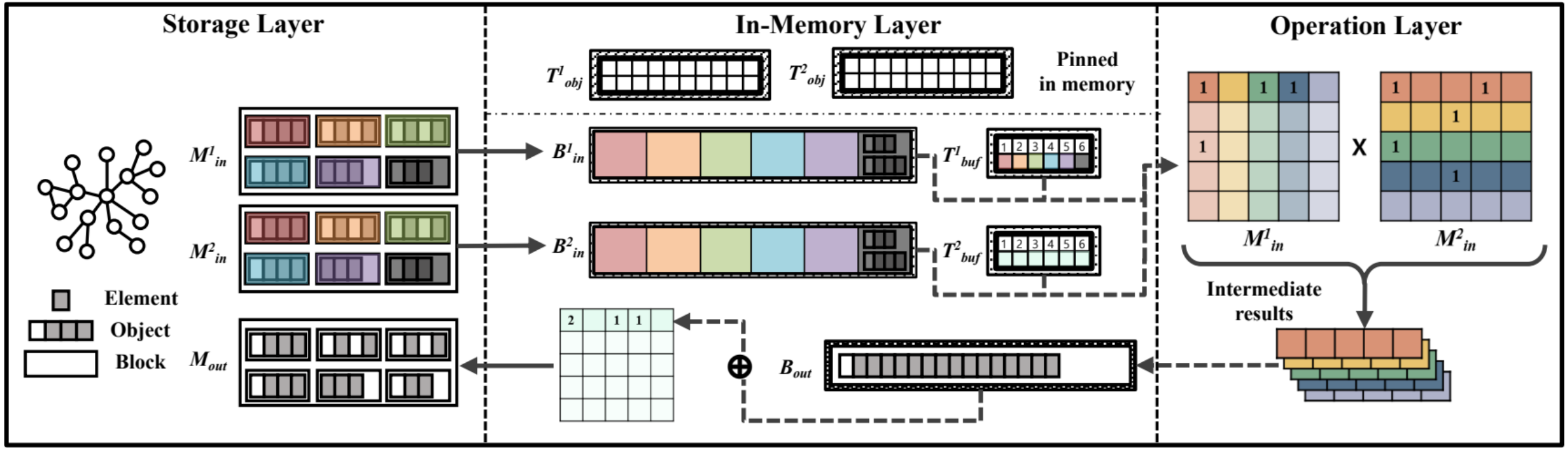
- **Performs SpGEMM utilizing *external storage of a single machine***
  - ***Scalable:*** Processing a large-scale graphs that does not fit in main memory
  - ***Efficient:*** Requiring only intra-machine communication overhead



Overview of SAGE

3-layer architecture

- *Storage layer*
- *In-memory layer*
- *Operation layer*



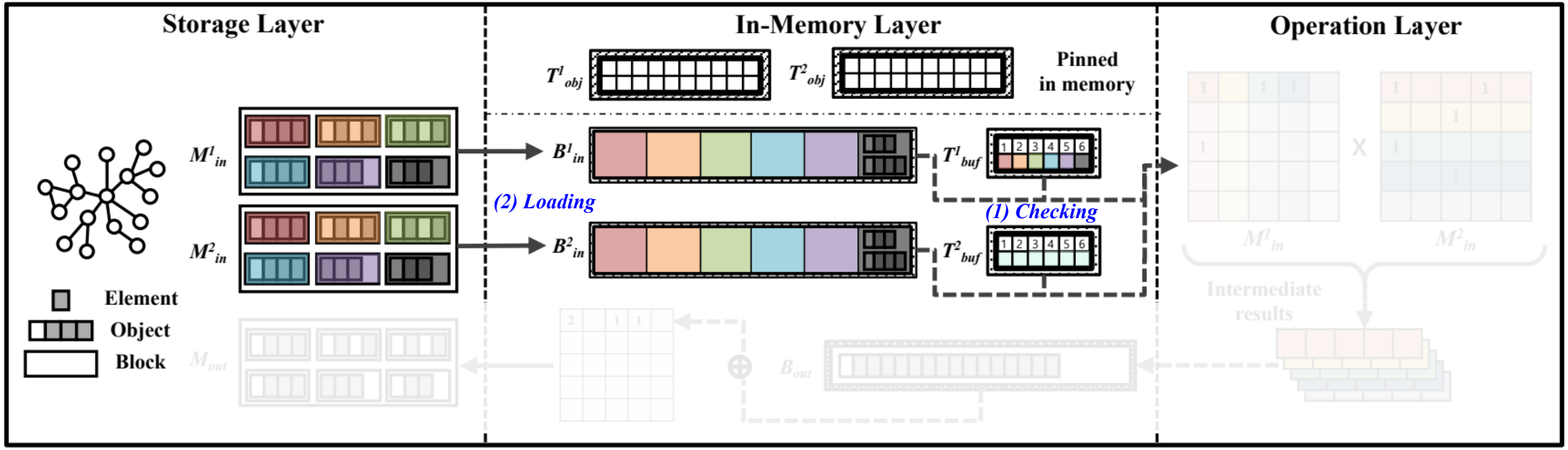
Overview of SAGE

Algorithm of SAGE (1/2)

Data loading

(1) Checking whether the blocks having the required rows are in the input buffers

(2) Loading the blocks into the input buffers



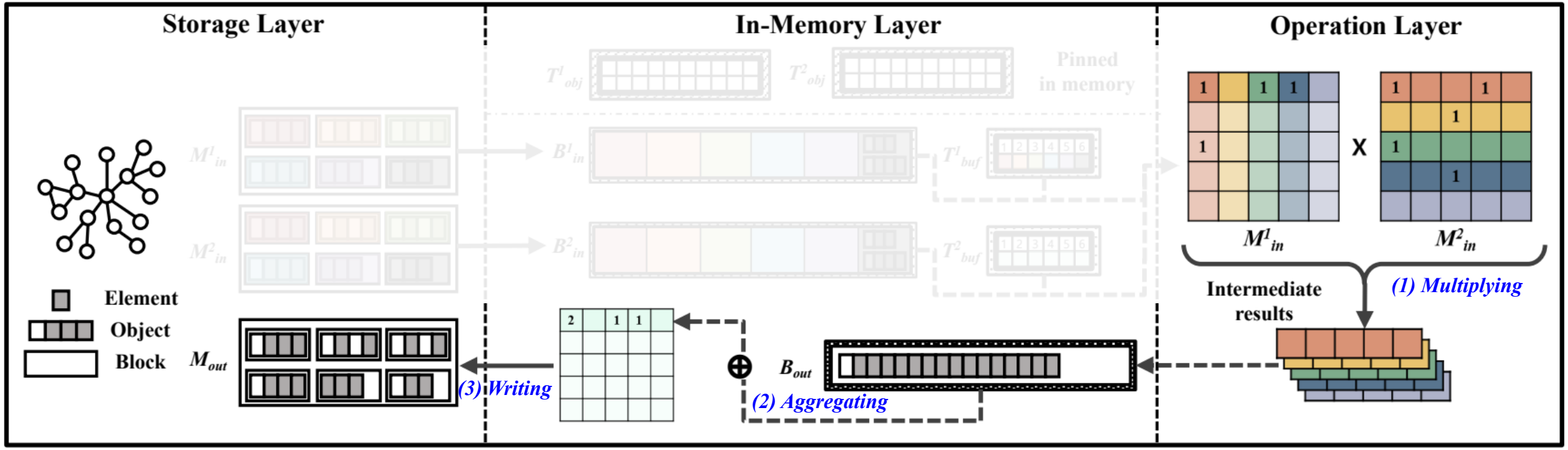
Algorithm of SAGE (2/2)

Row-wise product

(1) Multiplying a pair of rows in the loaded blocks

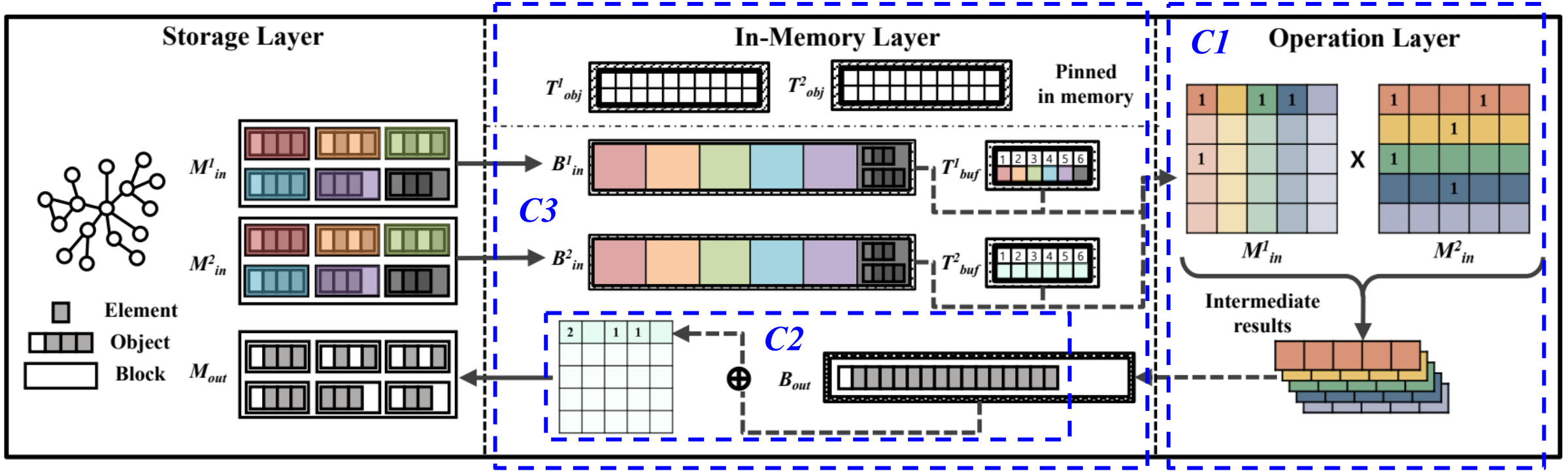
(2) Aggregating the intermediate results into final results

(3) Writing the final results to the storage



Performance-Critical Challenges

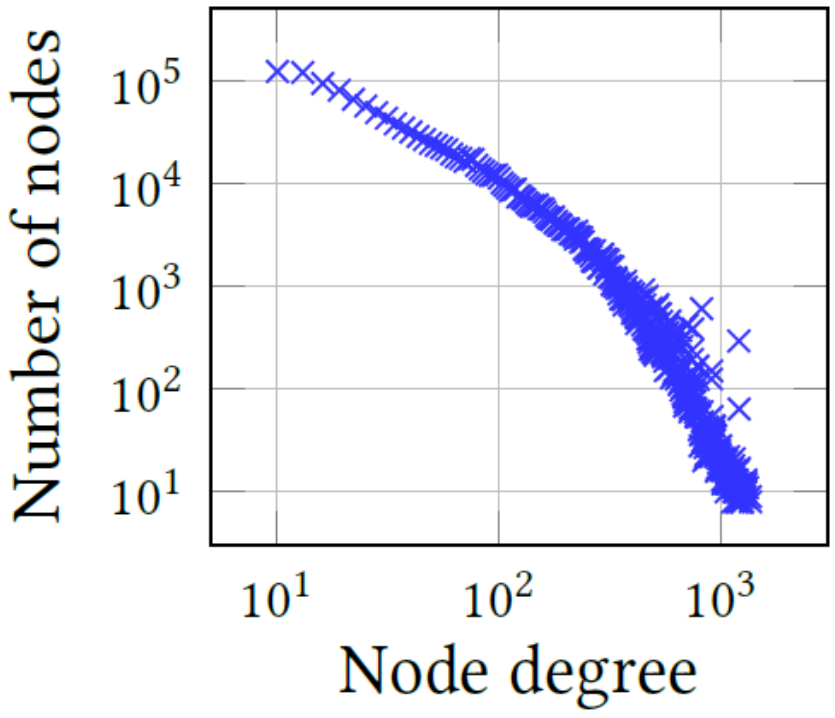
- Three challenges in processing *storage-memory I/Os*
  - *C1: Workload balancing*
  - *C2: Intermediate handling*
  - *C3: Memory management*

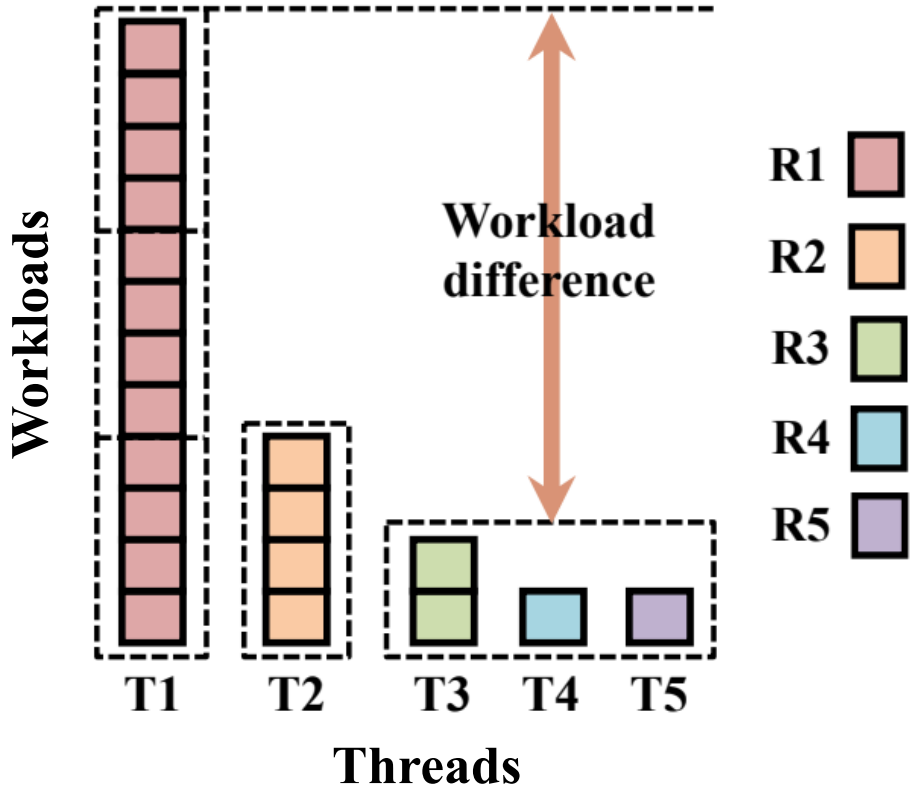


C1: Workload Balancing

Distributing workloads of SpGEMM evenly across multiple threads

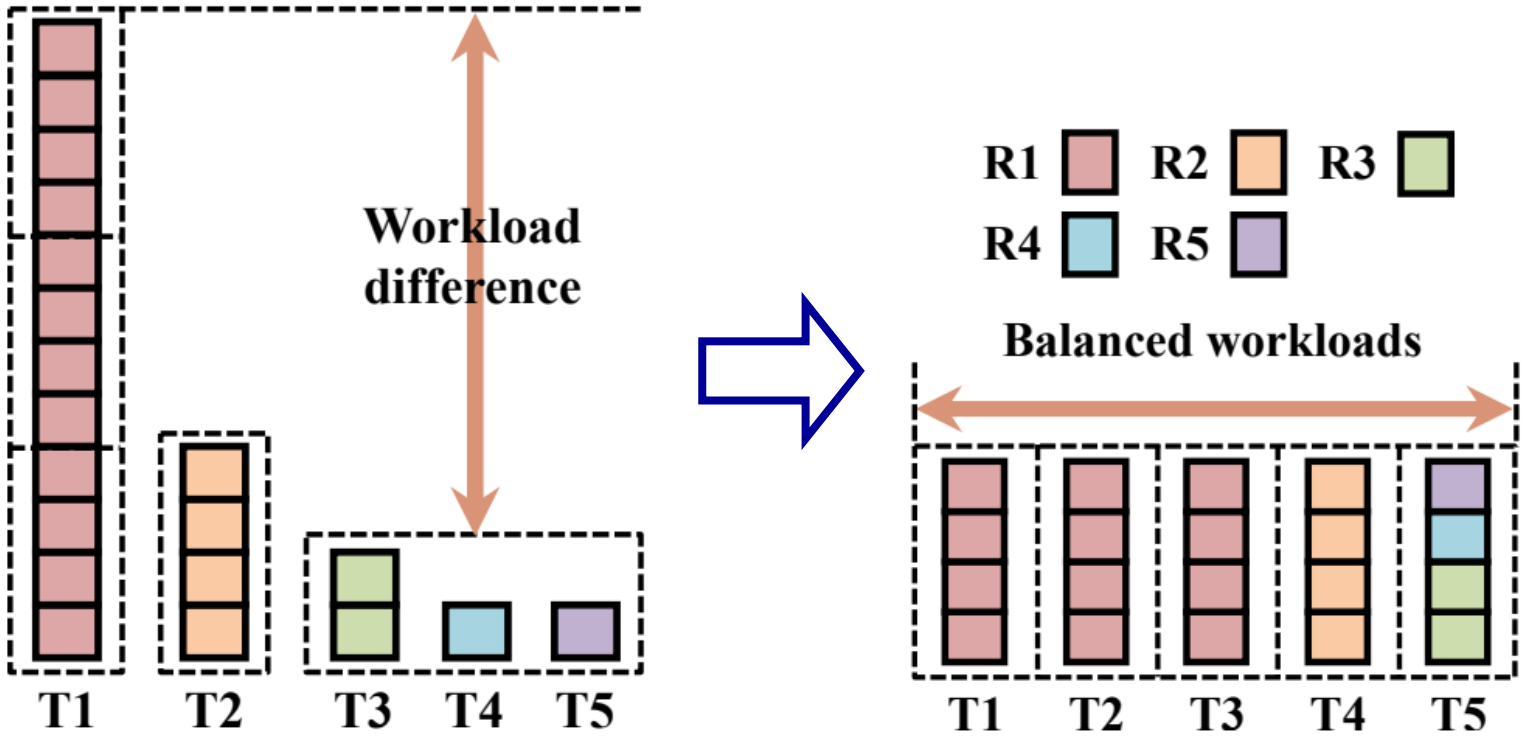
- Real-world graphs tend to follow ***power-law degree distribution***
- Row-based workload allocation could cause ***workload imbalance among threads***





Op1: Block-Based Workload Allocation

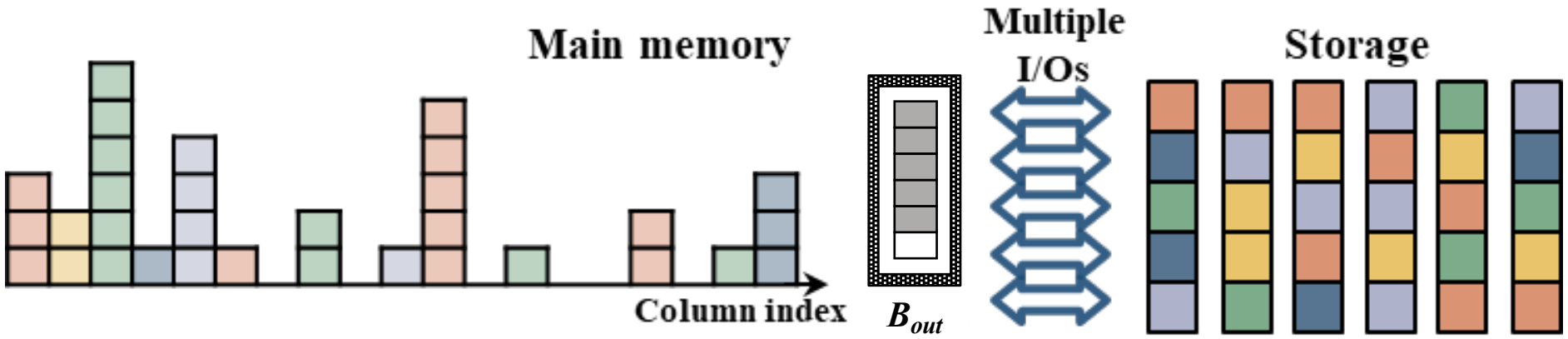
- Assigns the *same number of blocks* to each thread



C2: Intermediate Handling

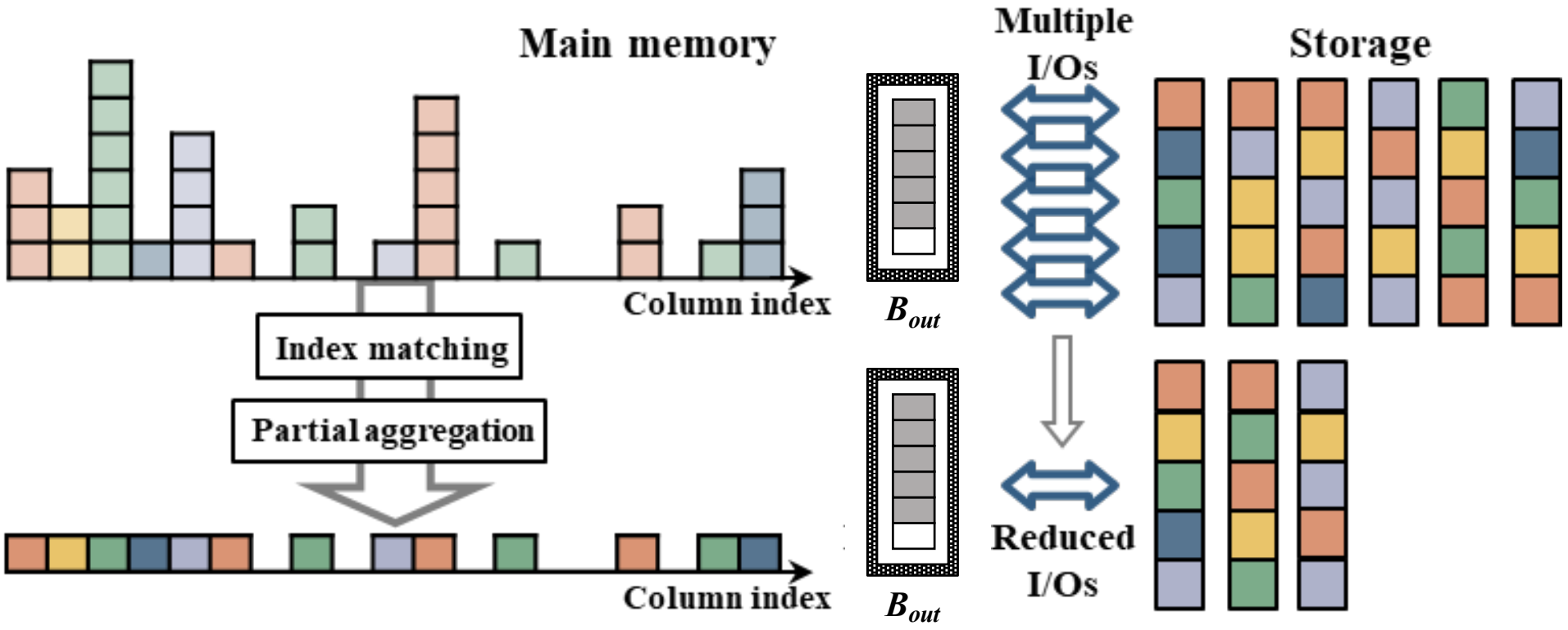
Handling the storage-memory I/Os generated by the intermediate results

- The output buffer may not be sufficient to store all intermediate results
- The intermediate results should be ***flushed to the storage multiple times***



Op2: In-memory Partial Aggregation

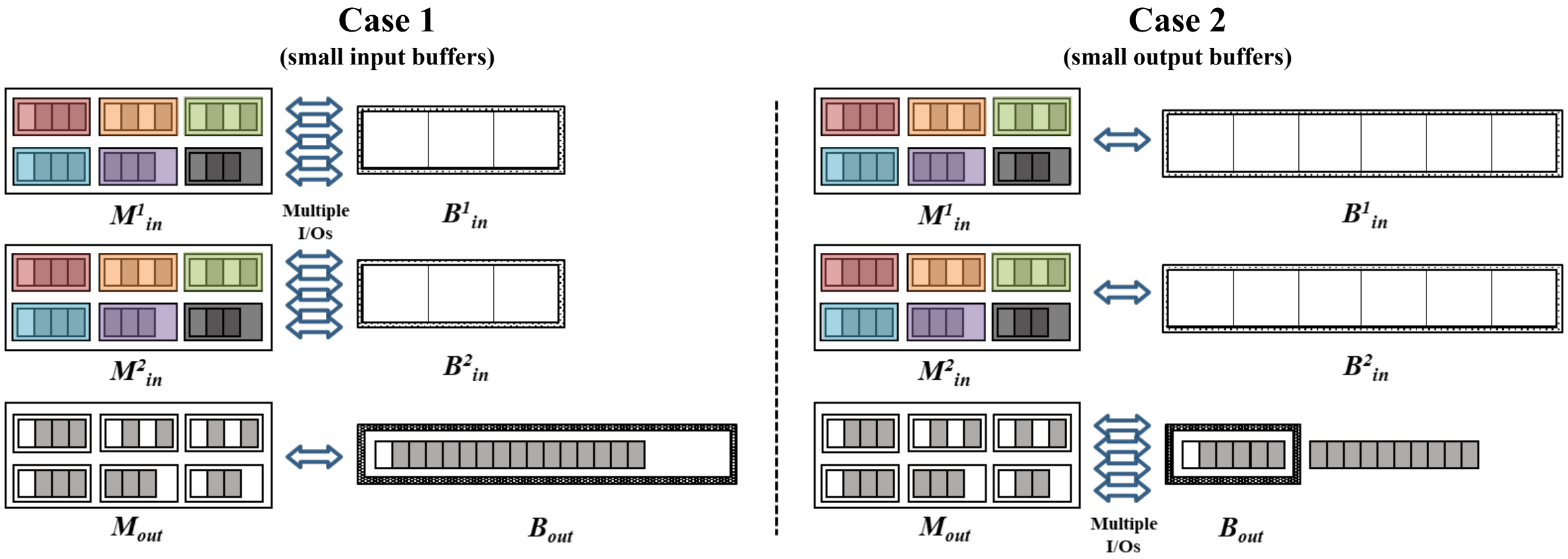
- Aggregates the intermediate results *directly in the output buffer*



C3: Memory management

Allocating the main memory space to three input/output buffers

- ***The size of main memory is limited*** in a single machine
- ***The amount of the I/Os can vary*** depending on the proportions of the three buffers



Op3: Distribution-Aware Memory Allocation

☐ Adjusts the proportions based on the characteristics of real-world graphs

(1) B_{in}^{1}: Allocate the size that all threads can process its workload simultaneously

(2) B_{out}: Allocate a portion of the size required for in-memory SpGEMM

(3) B_{in}^{2}: Allocate the remaining main memory

$$\text{SAGE\_}mem(C, b, t, \alpha) = \begin{cases} b \times t, & B_{in}^{1} \\ C - (|B_{in}^{1}| + |B_{out}|), & B_{in}^{2} \\ \alpha \times max\_row \times t, & B_{out}. \end{cases}$$

C, b: the size of the main memory and a block, respectively

t: the number of threads

α: the hyperparameter to adjust the output buffer size

max_row: the maximum size of the intermediate results among all rows

Experimental Questions

☐ EQ1: Comparison with single-machine-based approach

- Does SAGE improve the performance of SpGEMM, compared to the existing single-machine-based methods?

☐ EQ2: Comparison with distributed-system-based approach

- Does SAGE improve the performance of SpGEMM, compared to the existing distributed-system-based methods?

☐ EQ3: Scalability

- How does the SpGEMM performance of SAGE scale up with the increasing size of graphs?

☐ EQ4: Effectiveness of optimization strategies

- How effective are the proposed strategies of SAGE in improving the performance of SpGEMM?

Experimental Setup

Datasets

- 26 Real-world datasets
  - Ranging from 10MB to 60GB
- 42 Synthetic datasets
  - R-MAT (Graph500, SSCA, and ER)

Parameters of SAGE

- # of threads t as 4
  - # of physical cores in the CPU
- Hyperparameter α as 12.5%

Competing methods

- Single-machine-based approach
  - Intel MKL
- Distributed-system-based approach
  - SpSUMMA
  - Graphulo
  - gRRp

Evaluation protocols

- (1) SpGEMM of “the same two matrices” ($M \times M$)
- (2) SpGEMM of “two different matrices” ($M_1 \times M_2$ or $M \times M^T$)

EQ1: Comparison with Single-Machine-Based Approach

SpGEMM performance in real-world graphs

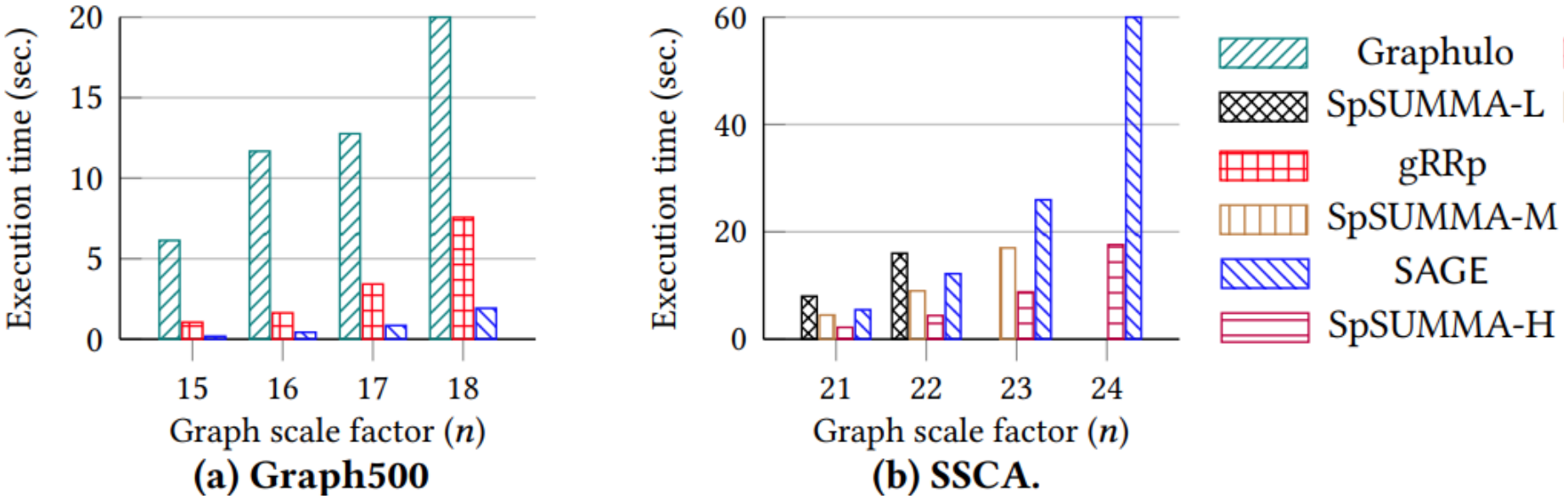
- SAGE ***successfully performs*** SpGEMM on very large datasets (***i.e., Scalable***)
  - Intel MKL could not handle SpGEMM on these datasets due to the limited main memory
  - SAGE also provides comparable performance to the Intel MKL

Dataset	Poisson3D	Enron	Epinions	Sphere	Filter3D	598a	Torso2	Cop20k	Cage12
Intel MKL	**0.1131**	**0.5792**	**0.6343**	**0.2521**	**0.3921**	**0.2660**	**0.0940**	**0.4120**	**0.3669**
SAGE	0.3206	0.7497	0.7001	0.5335	0.7436	0.5626	0.3976	0.7433	0.6331
Difference (sec.)	0.2075↑	0.1705↑	0.0658↑	0.2814↑	0.3515↑	0.2966↑	0.3036↑	0.3313↑	0.2662↑

Dataset	Slashdot	Gowalla	Pokec	Youtube	Livejournal	Wikipedia	UK-2005	SK-2005	Yahoo
Intel MKL	1.9890	7.8299	12.8732	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM
SAGE	**1.7558**	**6.772**	**10.6391**	**38.734**	**94.8283**	**2,276**	**616**	**3,709**	**12,994**
Difference (sec.)	0.2332↓	1.0579↓	2.2341↓	-	-	-	-	-	-

EQ2: Comparison with Distributed-System-Based Approach

SpGEMM performance in synthetic graphs



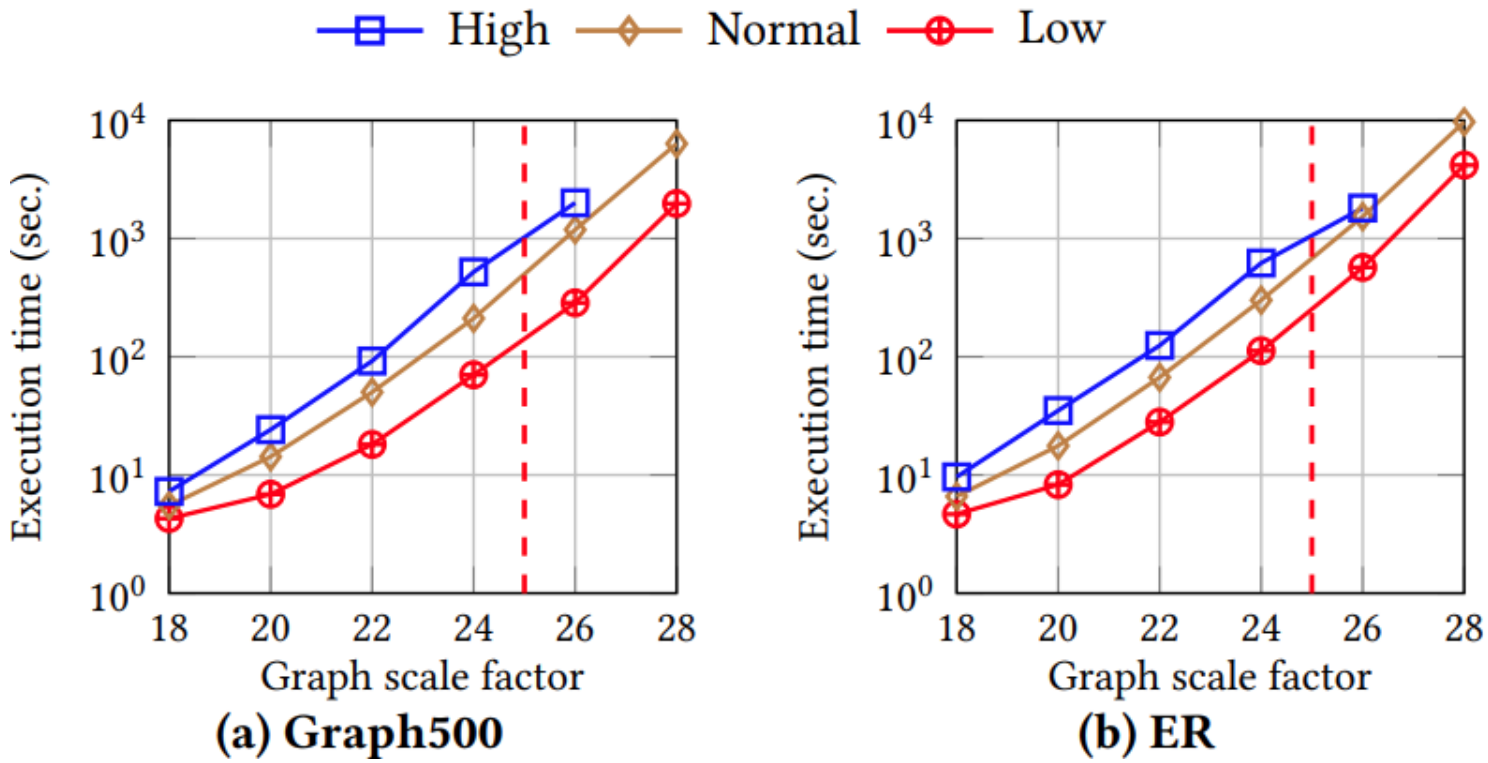
SpGEMM performance in real-world graphs

Dataset	Sphere	Filter3D	598a	Torso2	Cage12	144	Wave	Majorbasis	Scircuit	Mac	Offshore	Mario	Tmt_sym
Graphulo	7.37	12.91	9.57	4.81	13.28	11.97	11.85	8.07	6.36	4.63	18.86	9.07	13.55
gRRp	1.50	3.71	1.65	0.86	2.63	2.35	1.87	1.52	1.02	1.31	3.89	1.70	3.34
SAGE	**0.36**	**0.77**	**0.34**	**0.11**	**0.54**	**0.48**	**0.41**	**0.25**	**0.18**	**0.25**	**0.93**	**0.30**	**0.62**
Difference (sec.)	1.14↓	2.94↓	1.31↓	0.75↓	2.09↓	1.87↓	1.46↓	1.27↓	0.84↓	1.06↓	2.96↓	1.40↓	2.72↓

EQ3: Scalability

SpGEMM performance in synthetic graphs

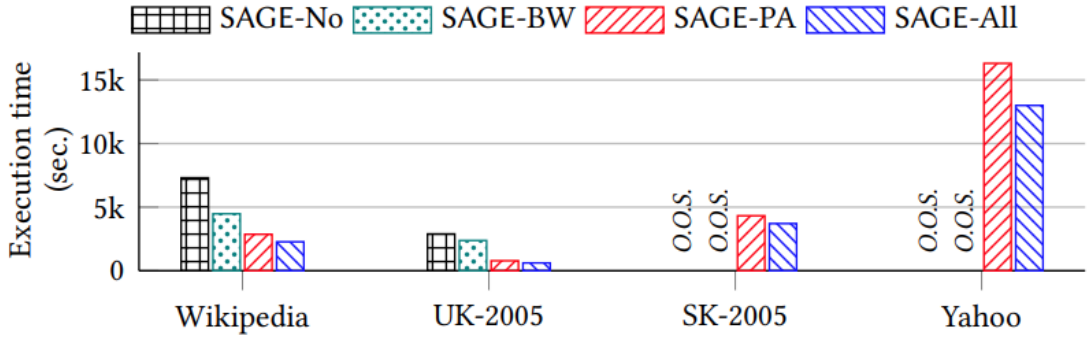
- SAGE provides ***(almost) linear scalability*** with the increasing sizes of graphs



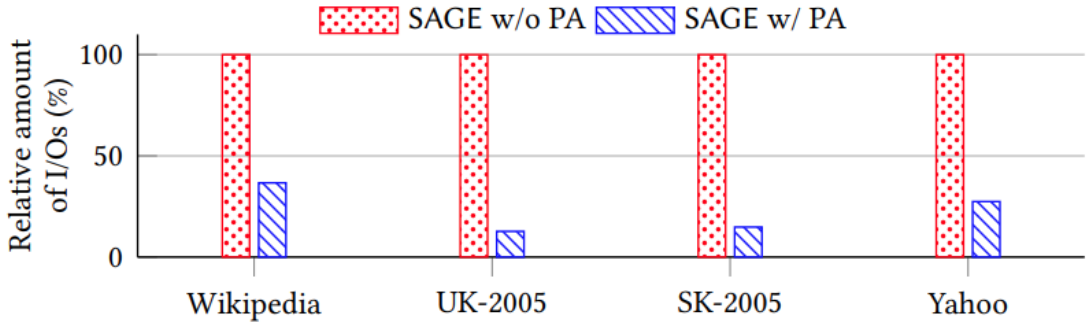
EQ4: Effectiveness of Optimization Strategies

☐ Block-based workload allocation and in-memory partial aggregation

- ***Substantially improve*** the performance of SAGE



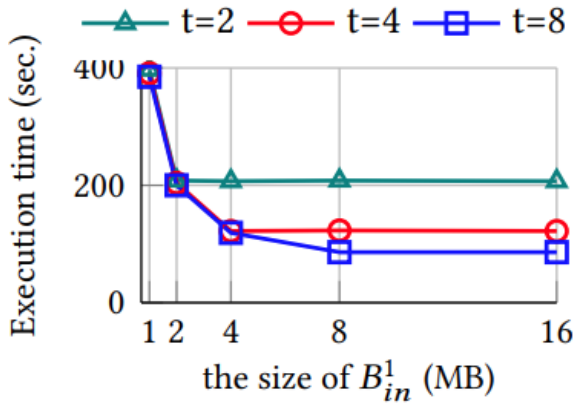
(a) Effects on the SpGEMM performance.



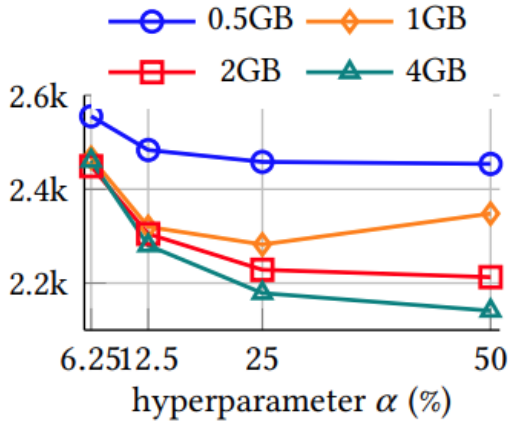
(b) Analysis of the in-memory-partial aggregation.

☐ Distribution-aware memory allocation

- ***Helps reducing*** the amount of storage-memory I/Os



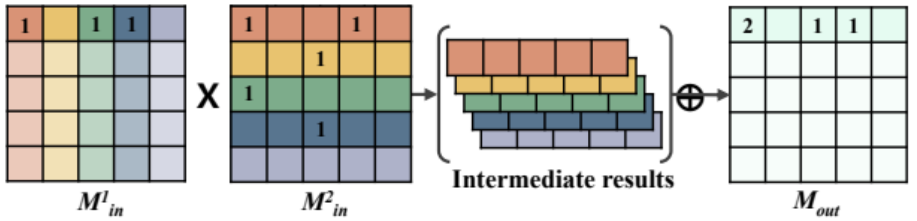
(a) Input buffer

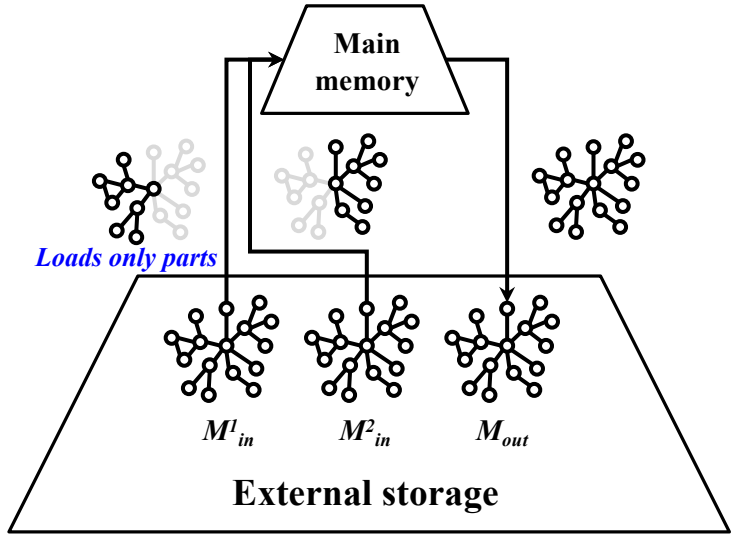


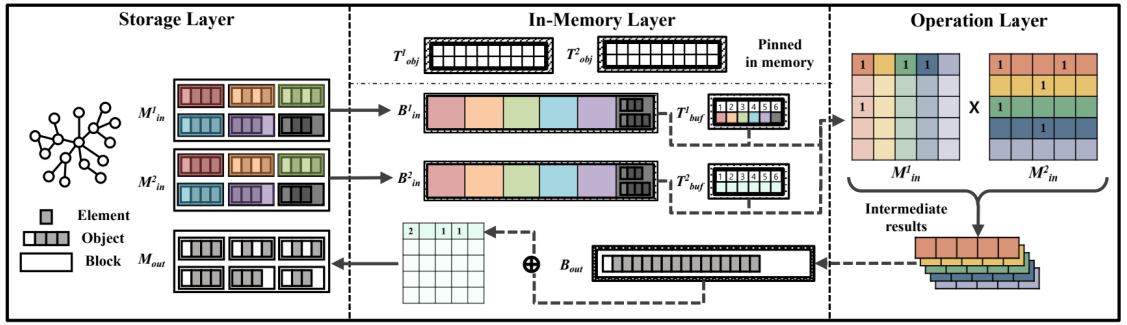
(b) Output buffer

Conclusions

- **Problem**
  - Large-scale SpGEMM for graph analysis
- **Existing approaches and challenges**
  - Single-machine-based approach (i.e., Not scalable)
  - Distributed-system-based approach (i.e., Not efficient)
- **Proposed method**
  - SAGE: A storage-based approach
    - Block-based workload allocation
    - In-memory partial aggregation
    - Distribution-aware memory allocation
- **Evaluation**
  - Scalability
  - Efficiency
  - Effectiveness







Thank You !