



清華大學  
Tsinghua University

Berkeley  
UNIVERSITY OF CALIFORNIA

# GraphSAR: A Sparsity-Aware Processing-in-Memory Architecture for Large-Scale Graph Processing on ReRAMs

Guohao Dai<sup>1</sup>, Tianhao Huang<sup>1,2</sup>, Yu Wang<sup>1</sup>,  
Huazhong Yang<sup>1</sup>, John Wawrzynek<sup>3</sup>

<sup>1</sup>Tsinghua University, <sup>2</sup>MIT, <sup>3</sup>UCB

[dgh14@mails.tsinghua.edu.cn](mailto:dgh14@mails.tsinghua.edu.cn)

1/22/2019





# Content



- **Background**
- **Motivation**
- **Related Work**
- **GraphSAR design**
- **Experiment Results**
- **Conclusion**



# Content

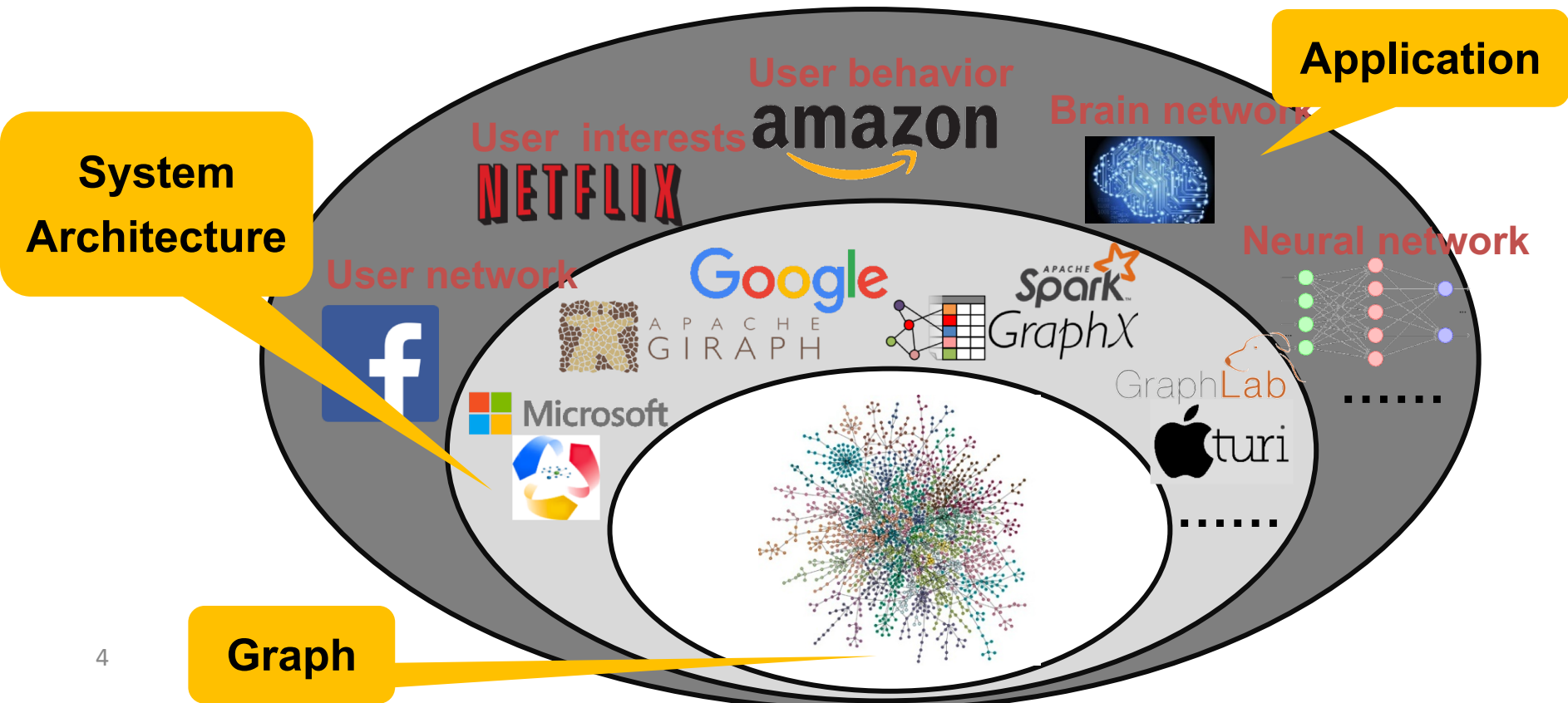


- **Background**
- Motivation
- Related Work
- GraphSAR design
- Experiment Results
- Conclusion



# Graphs are widely used!

- Graph: represent **data** and their **relationships**
- Application: social network analysis, neural network modeling, user behavior analysis, brain network modeling ...
- System & Architecture: support for graph-based applications

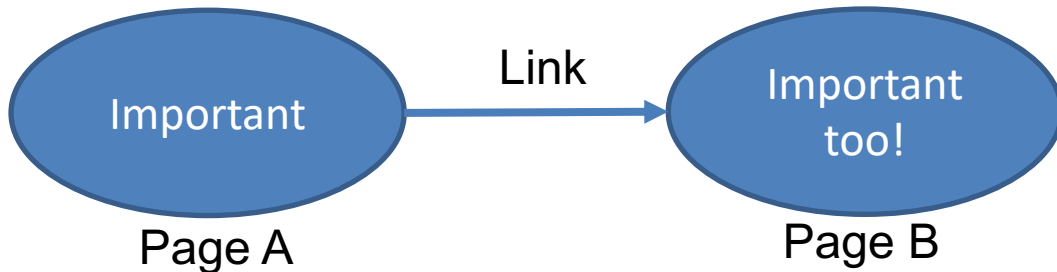






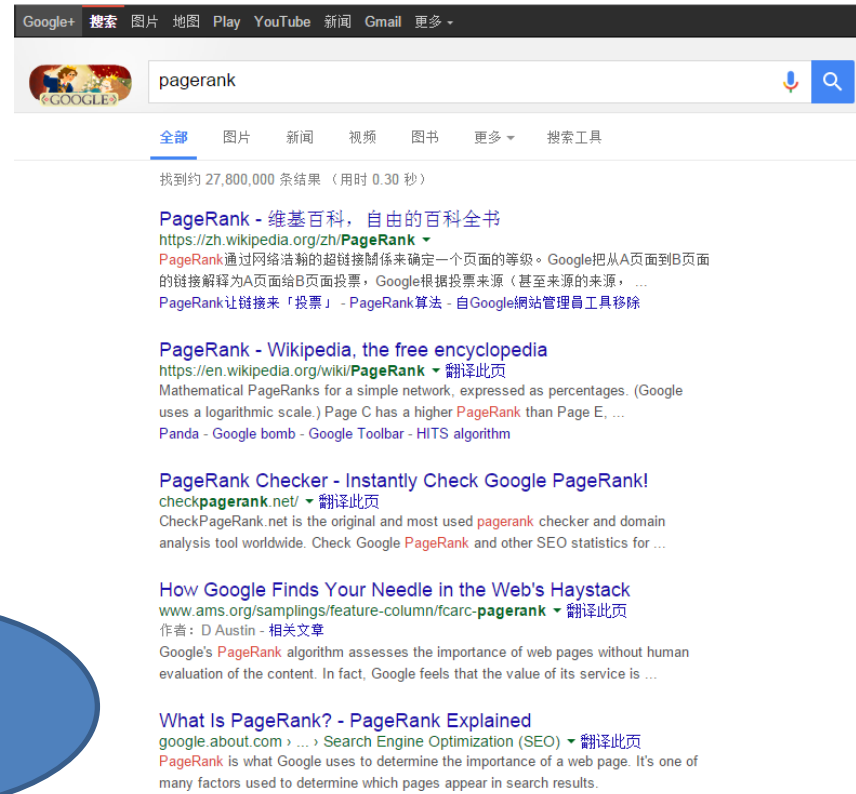
# Application I: PageRank

- Sorting billions of pages according to key words in one second
  - Graph: 2.9 b, 0.36 s
  - PageRank: 0.2 b, 0.30 s
- Google PageRank Algorithm
  - The rank of a page depends on ranks of pages which link to it



- Network = Graph

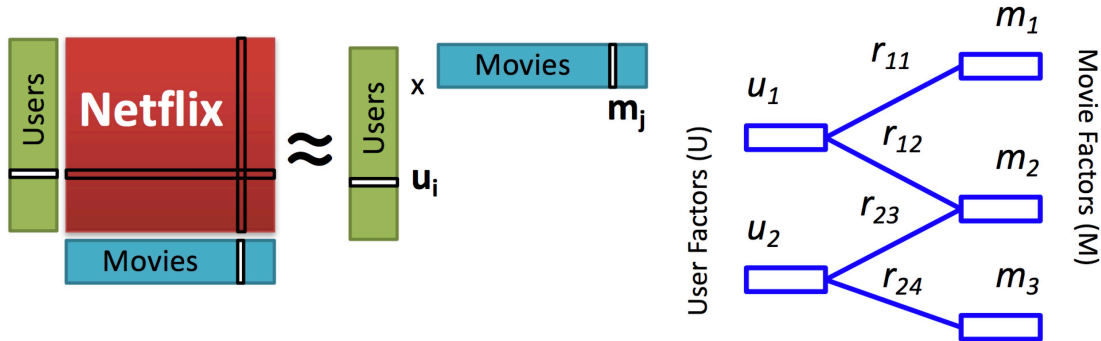
$$\text{PageRank}(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{\text{PageRank}(p_j)}{L(p_j)}$$





# Application II: Film Recommendation

- Collaborative filtering based on similar users



- ALS: Alternating Least Squares
  - Minimize Mean Square Error (MSE)

- Calculating using tags
- Recommending using non-tags

Iterate:

$$u_i = \arg \min_w \sum_{j \in N[i]} (r_{ij} - m_j \cdot w)^2$$

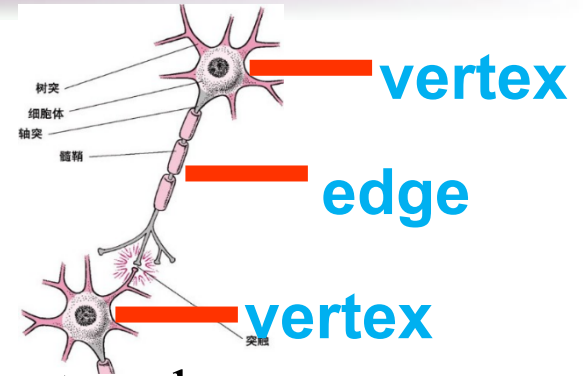
$$m_j = \arg \min_w \sum_{i \in N[j]} (r_{ij} - u_i \cdot w)^2$$

- Sparse matrix = Graph

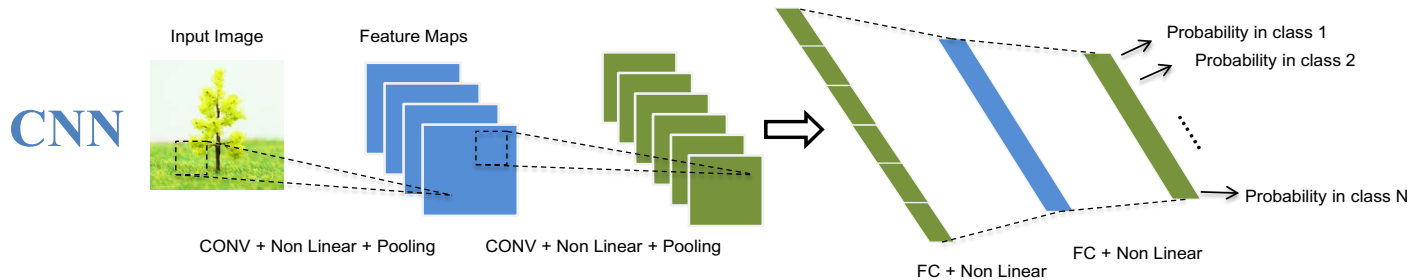
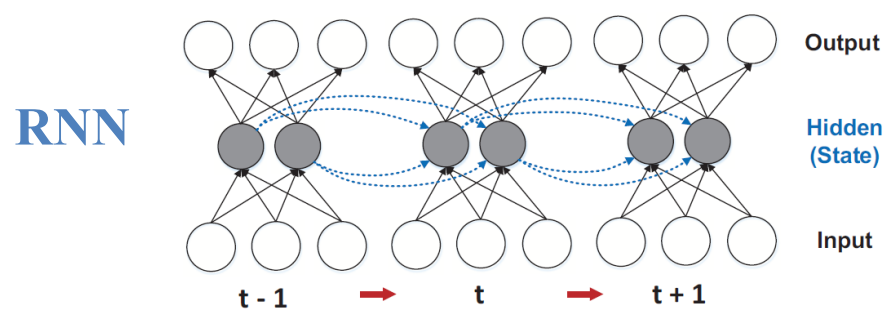
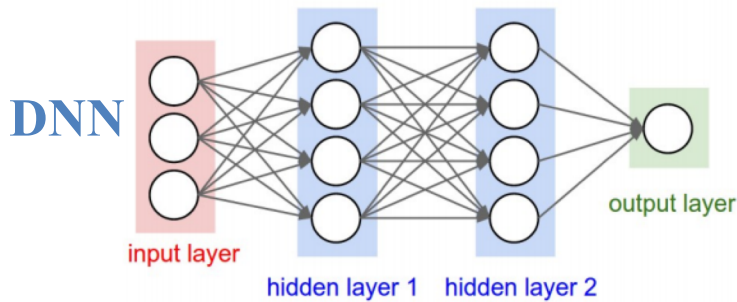
# Application III: Deep Learning

- Data & relationship = Graph

- Neuron: vertex
- Synapse: edge
- Stimulus intensity: value

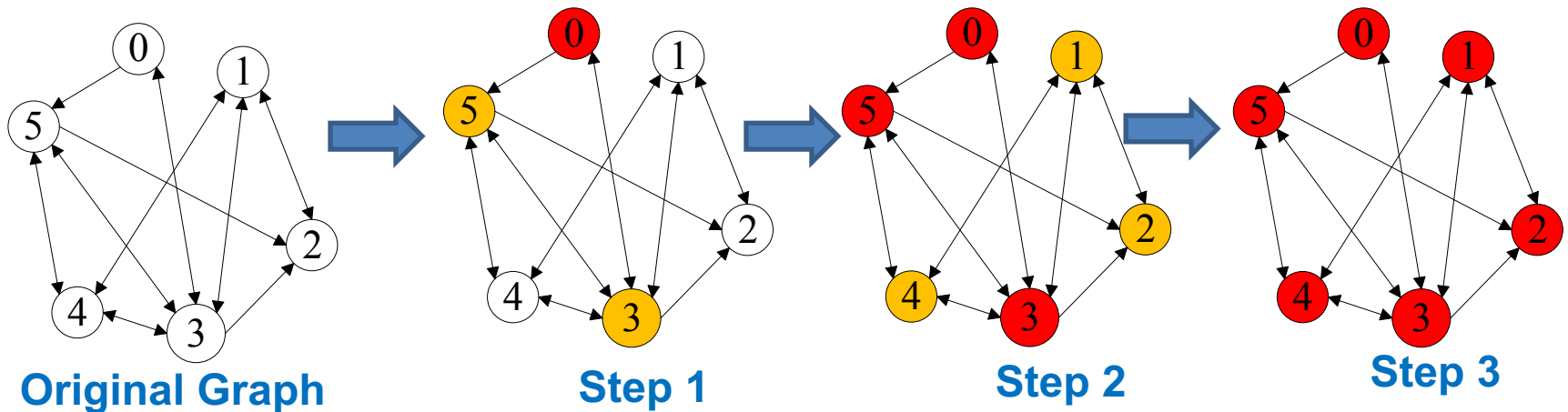
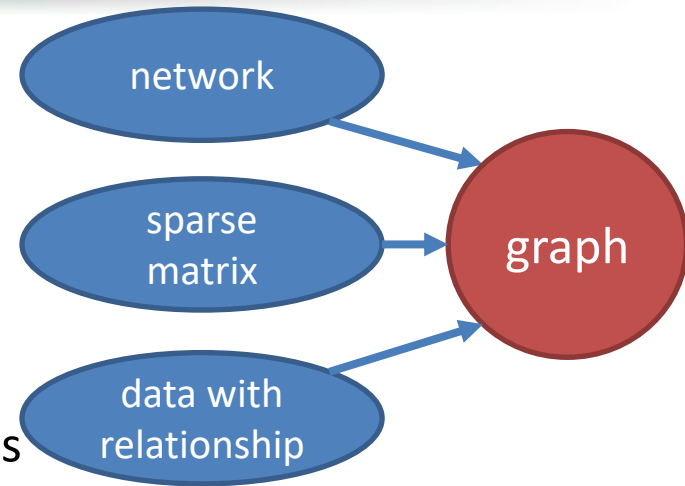


- Using graphs to represent different neural networks



# Generality requirement

- High-level abstraction model
  - Read-based/Queue-based Model for BFS/APSP [Stanford, PACT'10] ✗
  - GAS Model [Google, SIGMOD'10] ✓
- In GAS (Gather-Apply-Scatter) Model
  - Different algorithms → Different Apply functions
  - Traverse edges and scatter src to dst





# Content

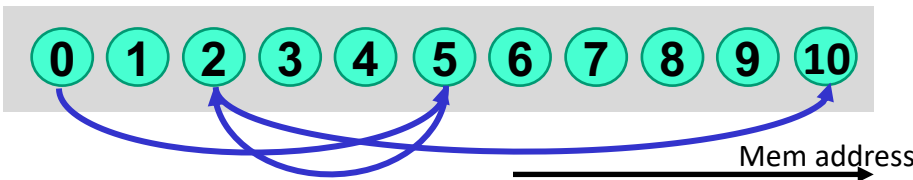
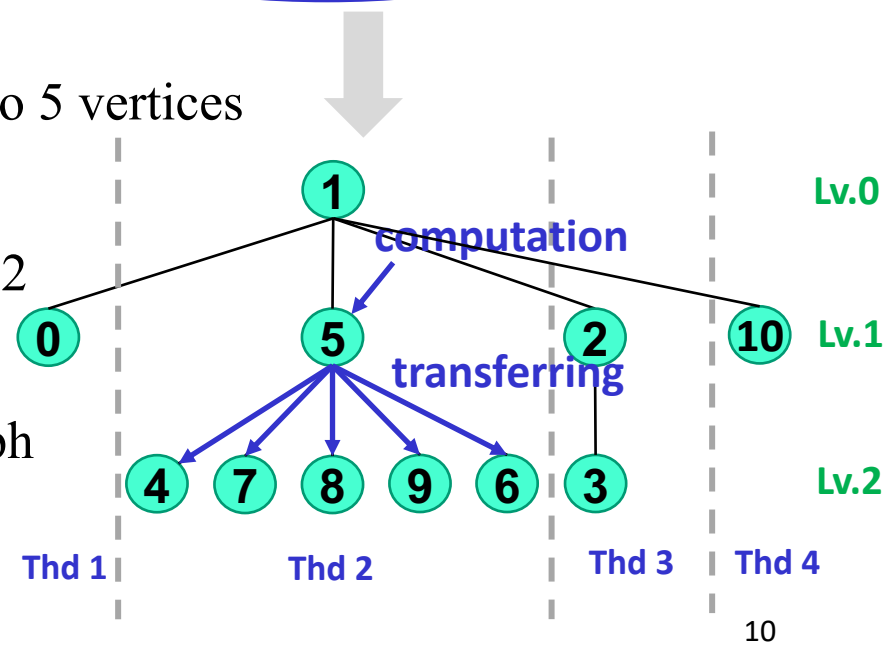
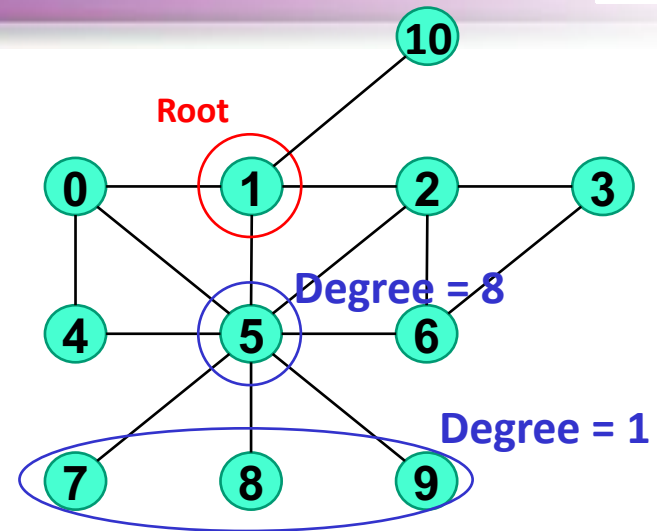


- Background
- **Motivation**
- Related Work
- GraphSAR design
- Experiment Results
- Conclusion and Future Work



# Characteristics of graph processing

- Example: Breadth-First Search
  - Root: v1
  - Generate BFS tree
- Unstructured
  - Degree = 1: v7, 8, 9, 10
  - Degree = 8: v5
- Data driven
  - Example: v5, transfer updated value to 5 vertices
- Unbalanced
  - Balanced in Lv.1 → unbalanced in Lv.2
- Poor locality
  - 4 vertices in Lv.1 spread all over graph





# Challenges in graph processing

- Key in graph processing: **efficient data transferring**

## Characteristics

## Difficulties

## Solutions

Unstructured  
Unbalanced

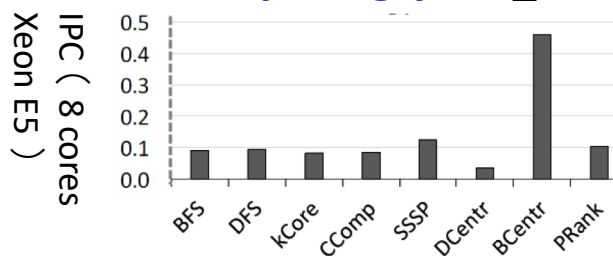
### Heavy traffics[Zhu\_OSDI\_2016]

System	PowerG.	PowerL.
Mem. Ref.	95.8G	87.2G
Comm. (GB)	115	38.1

Lower  
traffics

Data  
Driven

### Memory-hungry[Nai\_HPCA\_2017]



Higher  
bandwidth

Poor  
Locality

### Cache miss[Zhu\_OSDI\_2016]

System	Ligra	Galois	PowerG.	PowerL.
IPC	0.408	0.414	0.500	0.655
LLC Miss	43.9%	49.7%	71.0%	54.9%

Sequential  
access

Efficient data  
transferring required!



# Memristor & ReRAM

- **Memristor**

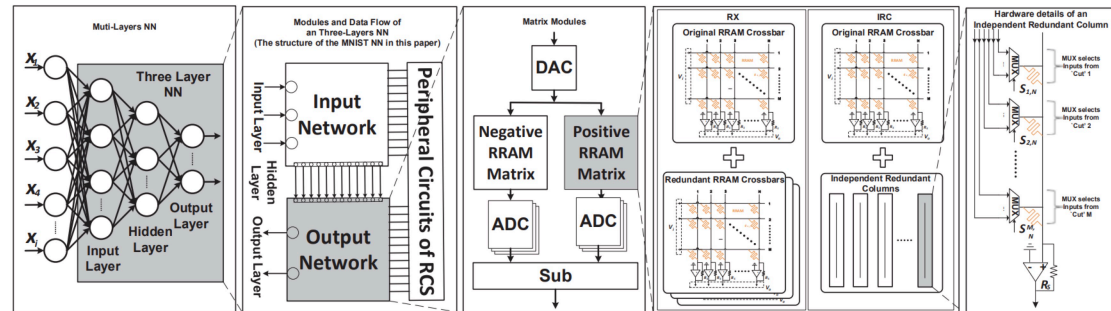
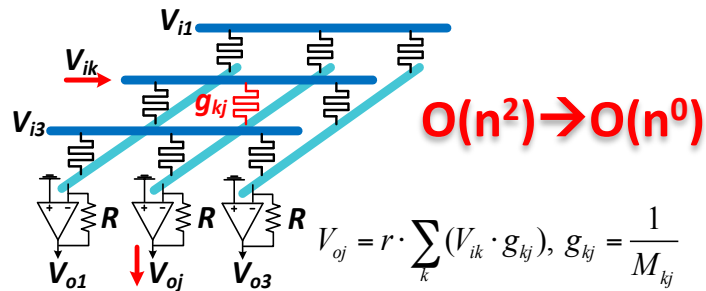
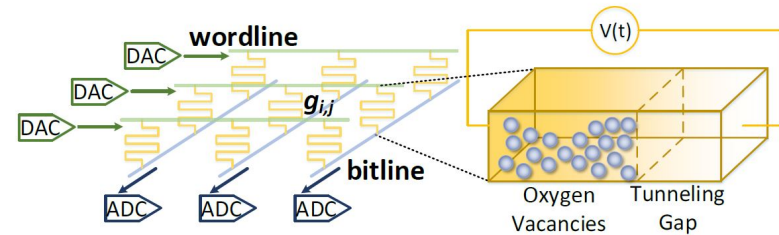
- Resistance can be changed by voltage

- **Storage**

- Memristor crossbar
- Using changeable resistance to store information

- **Computation**

- Processing-in-memory, 10x ~ 100x energy efficiency improvement compared with conventional von Neumann architecture



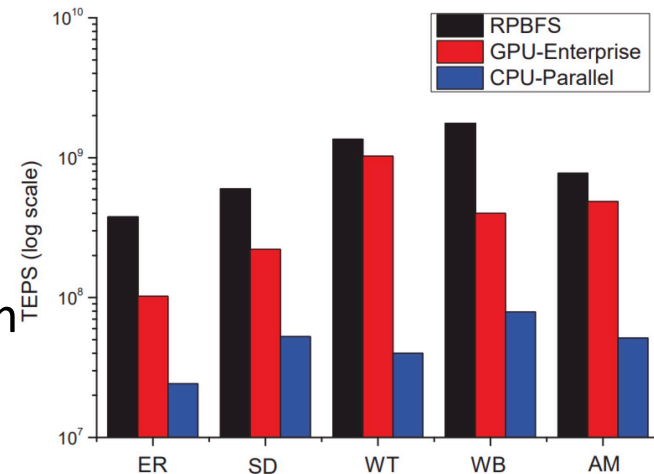
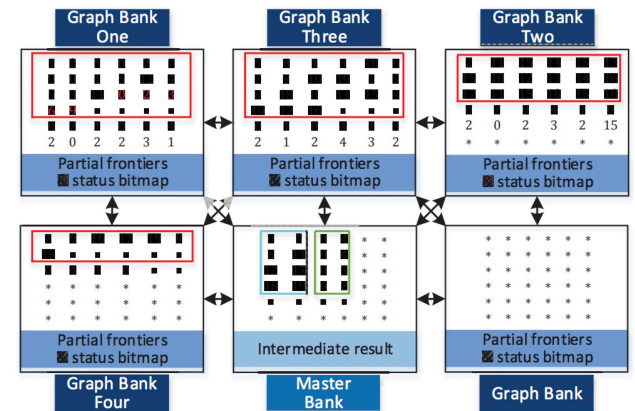


# Content



- Background
- Motivation
- **Related Work**
- GraphSAR design
- Experiment Results
- Conclusion

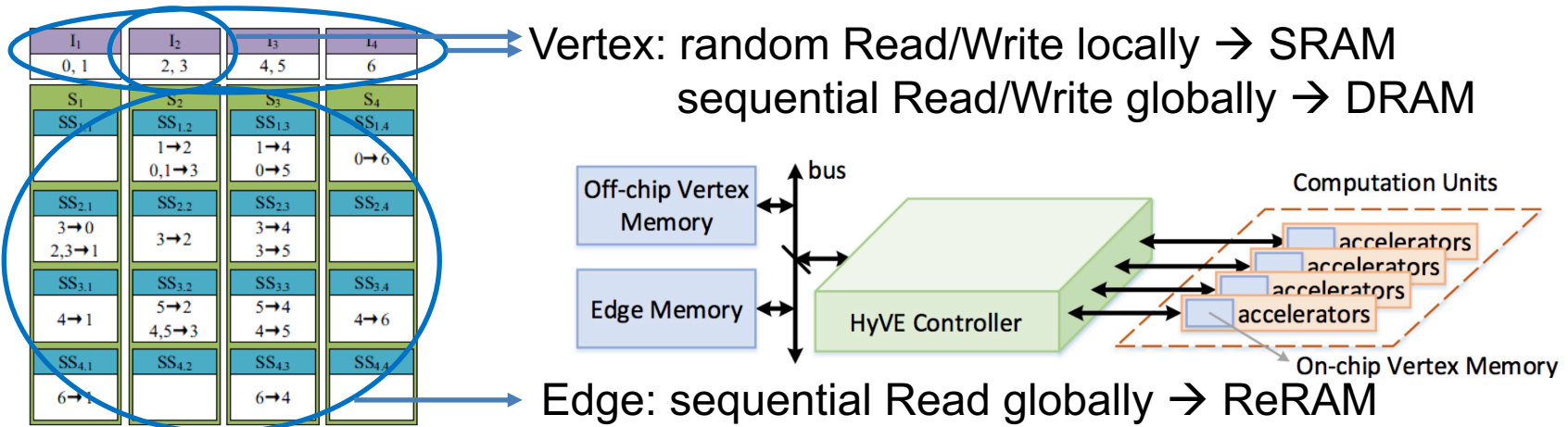
- Designed for Breadth-First Search
  - Using 1 bit to represent status of a vertex
  - Graph Bank: edge storage
    - Parallel processing
  - Master Bank: vertex storage
    - **Centralized processing**
- Performance
  - 33.8x speedup against CPU
  - 16.0x speedup against GPU
- However...
  - Only for BFS
  - Centralization scheme → Scalability problem





# HyVE [Ours, DATE 18]

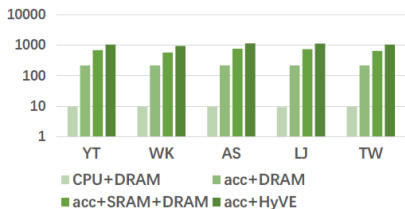
- Memory energy efficiency difference due to patterns
  - Different patterns → corresponding memories → hybrid memory



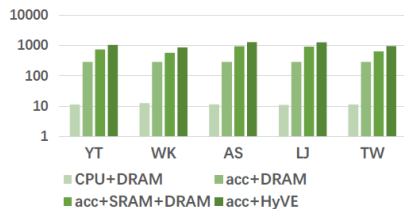
## Performance

- 114x energy efficiency improvement against CPU+DRAM
- Memory subsystem energy consumption < 50%

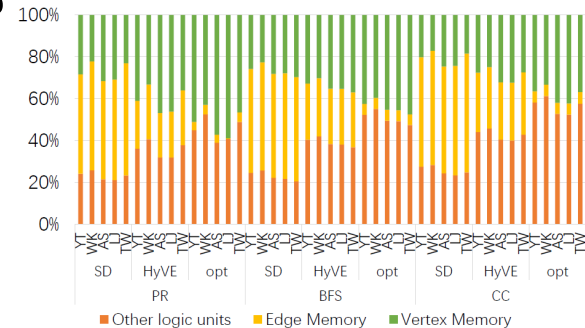
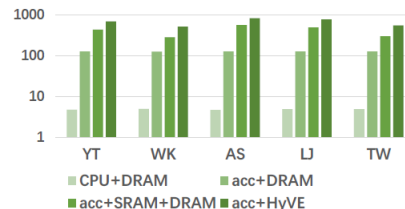
Energy efficiency running BFS (MTEPS/W)



Energy efficiency running CC (MTEPS/W)

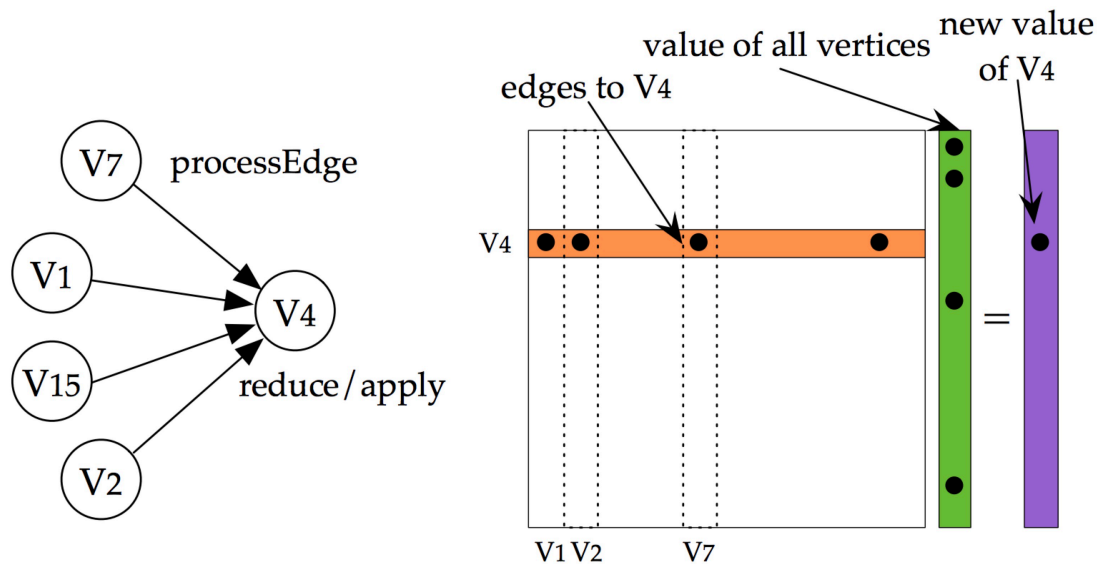


Energy efficiency running PR (MTEPS/W)



# GraphR [Duke, HPCA 18]

- Matrix-vector representation for graph processing
  - Src vertex vector, adjacency matrix  $\rightarrow$  Dst vertex vector
  - MVM (e.g., PageRank): direct mapping
  - Non-MVM (e.g., BFS): activating each row sequentially
- Divide a large adjacency matrix into small **blocks**



(a) Vertex Program in Graph View

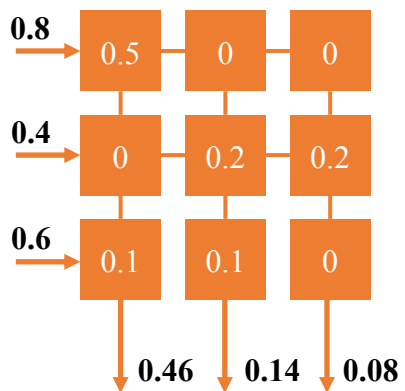
(b) Vertex Program in Matrix View



# GraphR [Duke, HPCA 18]



- MVM (e.g., PageRank): direct mapping

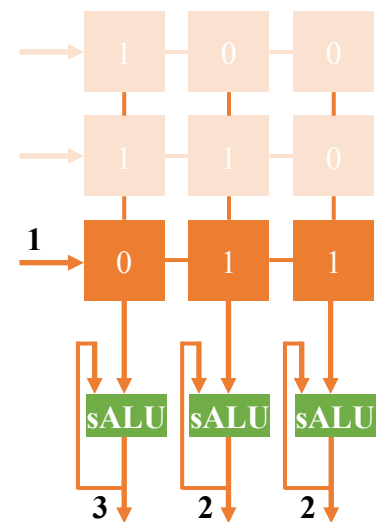
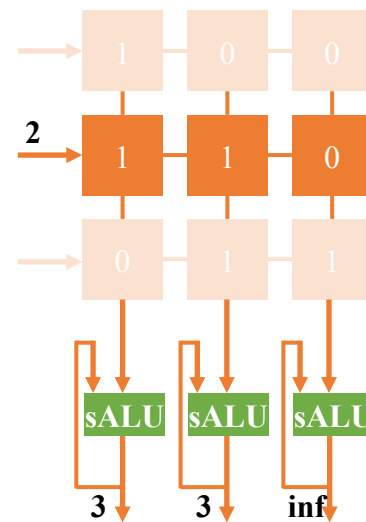
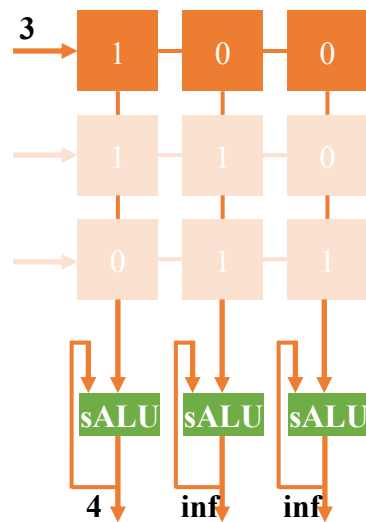




# GraphR [Duke, HPCA 18]



- Non-MVM (e.g., BFS): activating each row sequentially

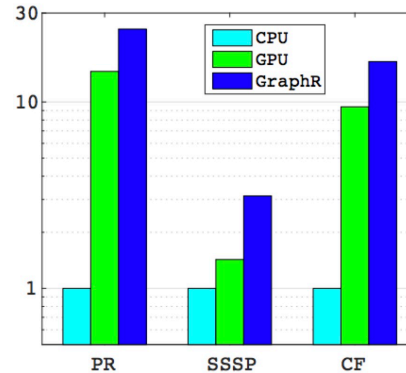




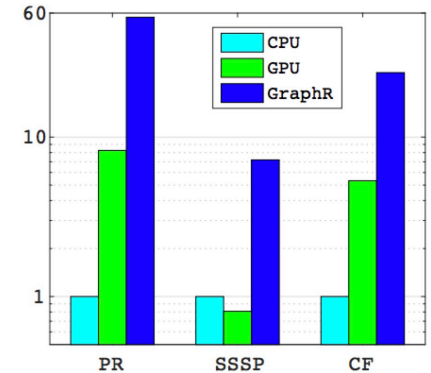


# GraphR [Duke, HPCA 18]

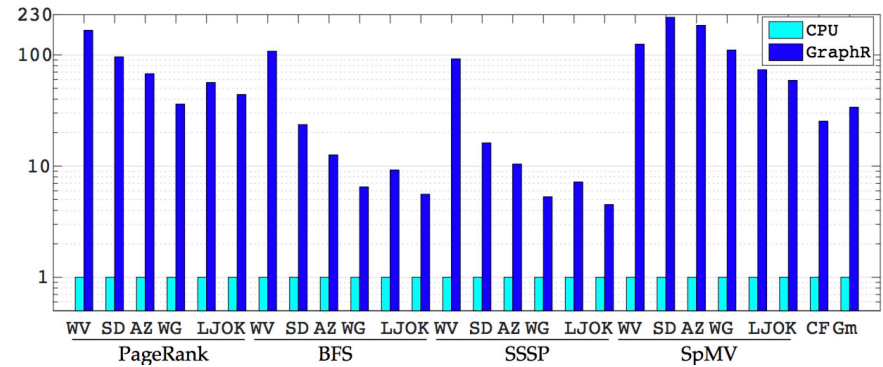
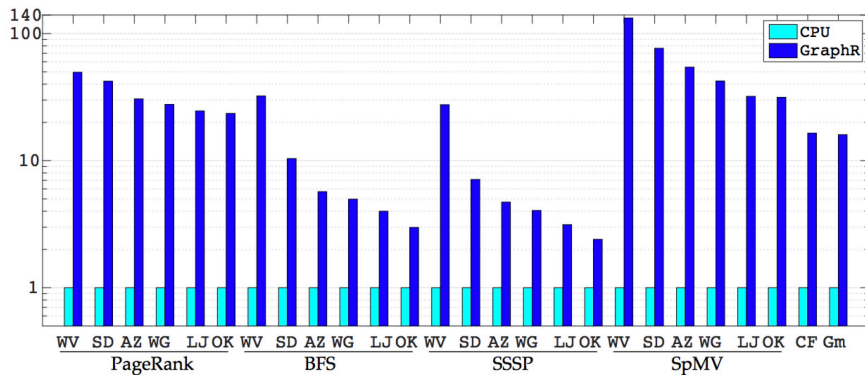
- Compared with CPU
  - Speedup: 16.01x
  - Energy efficiency: 33.82x
- Compared with GPU
  - Speedup: 1.69x ~ 2.19x
  - Energy efficiency: 4.77x ~ 8.91x



(a) Performance



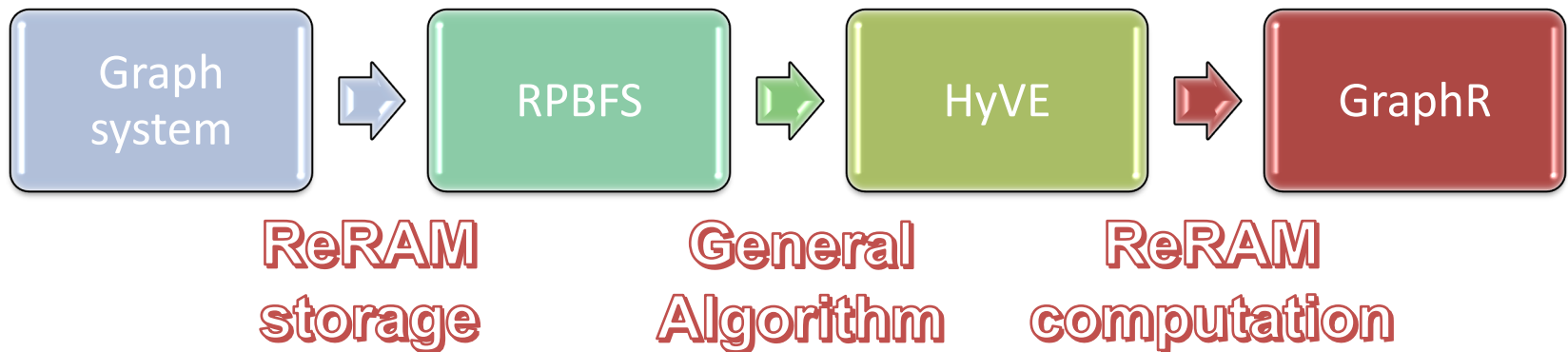
(b) Energy Saving





# Related works – conclusion

- Energy efficiency of graph processing can be improved by using ReRAM



	RPBFS	HyVE	GraphR
Algorithm	Only BFS	General purposed	General purposed
Storage	RRAM	Hybrid	RRAM
Computation	CMOS	CMOS	RRAM/CMOS



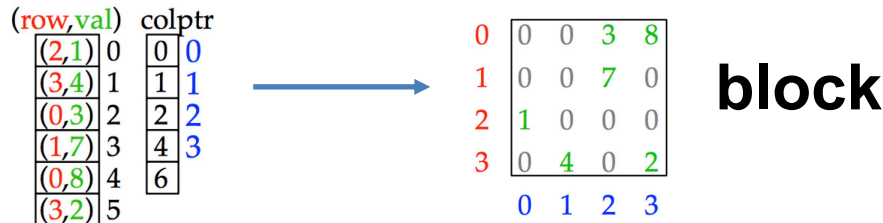
# Content



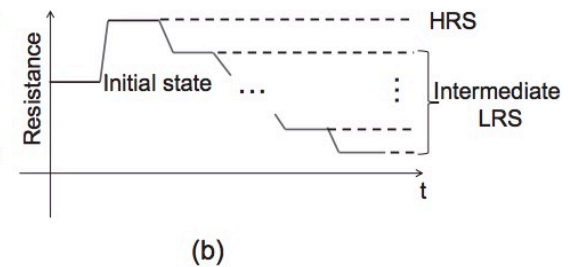
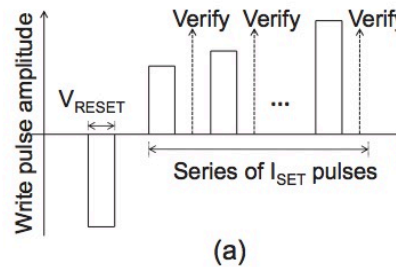
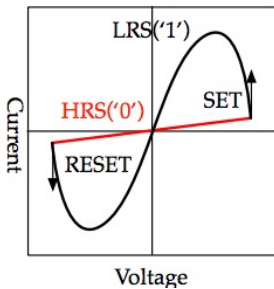
- Background
- Motivation
- Related Work
- **GraphSAR design**
- Experiment Results
- Conclusion

# However...

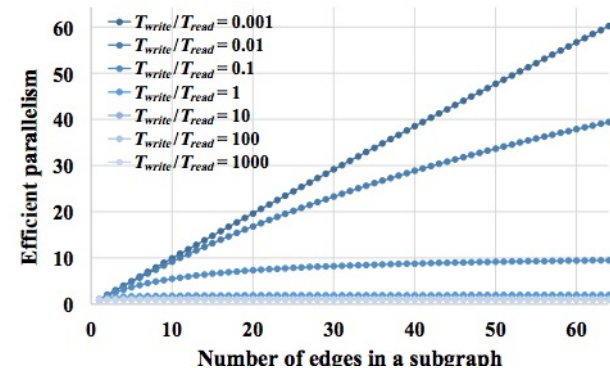
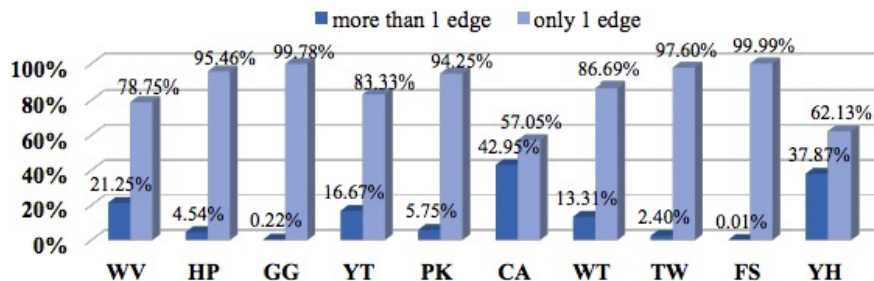
- GraphR: Writing ReRAM (adjacency list  $\rightarrow$  block)



- Write-and-verify scheme of ReRAM: **Heavy writing overheads**



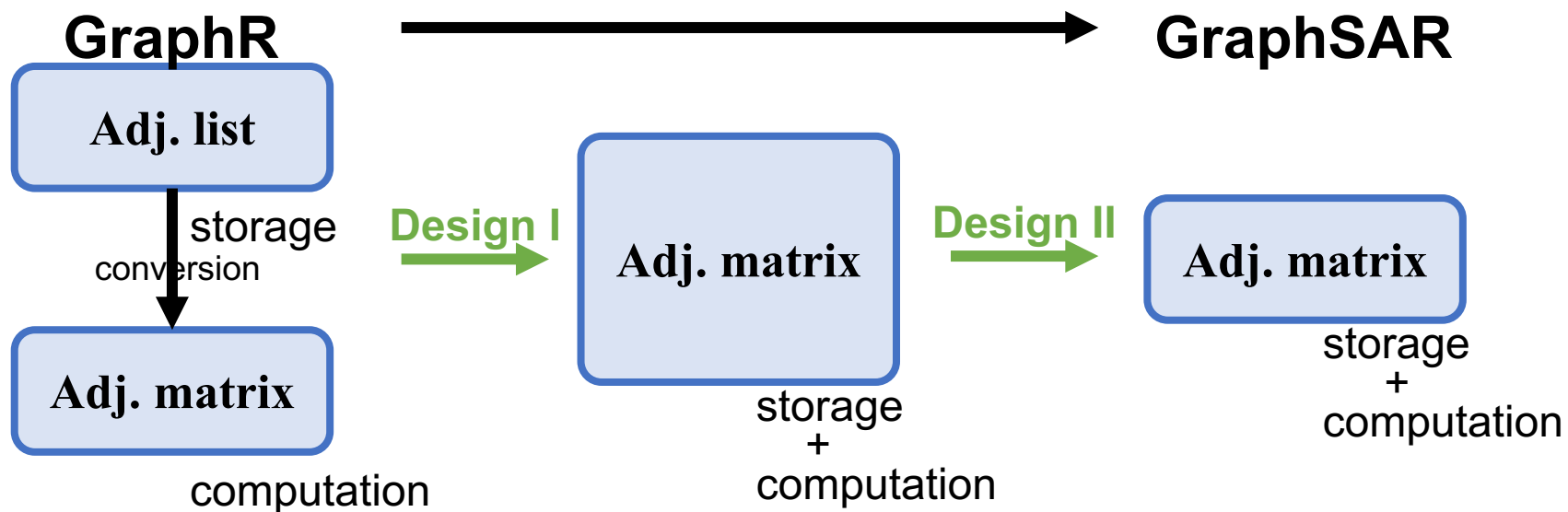
- Sparsity of graphs: **Low parallelism**





# GraphSAR Design

- Design I: Processing-in-memory
- Design II: Sparsity-aware partitioning



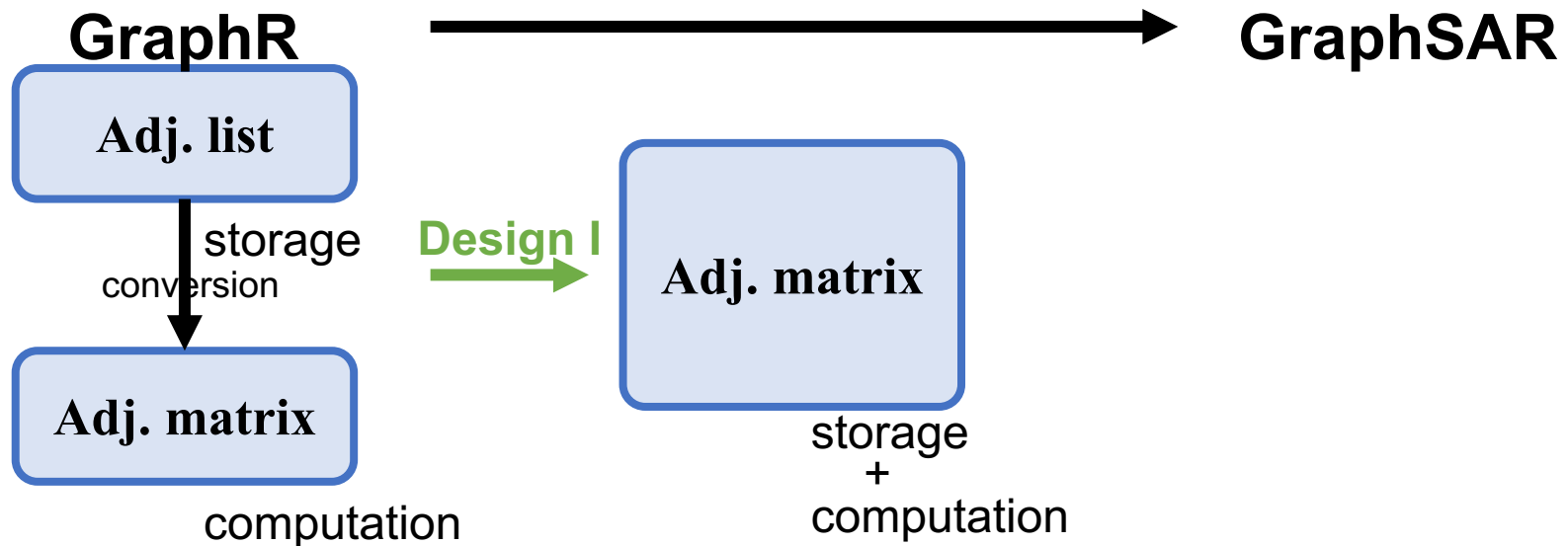


# Design I: Processing-in-memory

- Conversion leads to low parallelism/heavy writing overheads



- Directly storing blocks on ReRAM



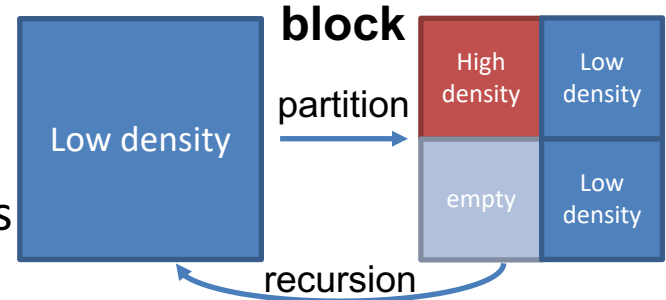
## Problem I:

- Heavy writing overheads
- Low parallelism



# Design II: Sparsity-aware partitioning

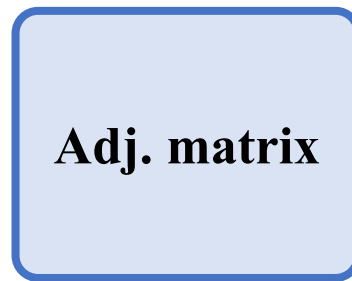
- Storing  $8 \times 8$  blocks leads to memory space overheads
- Sparsity-aware partitioning
  - Divide large sparse block into small ones
  - Drop empty blocks



**GraphR**

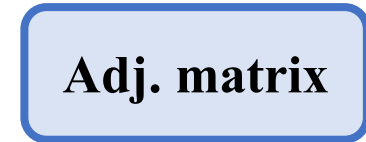


**GraphSAR**



storage  
+  
computation

**Design II**



storage  
+  
computation

**Problem II:**

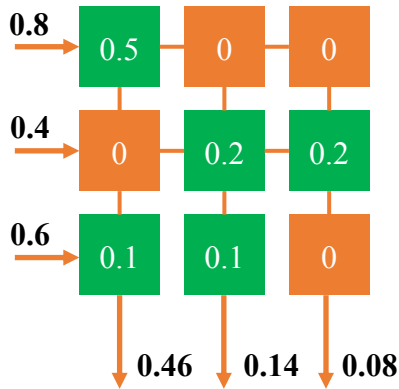
- **Memory space overheads**



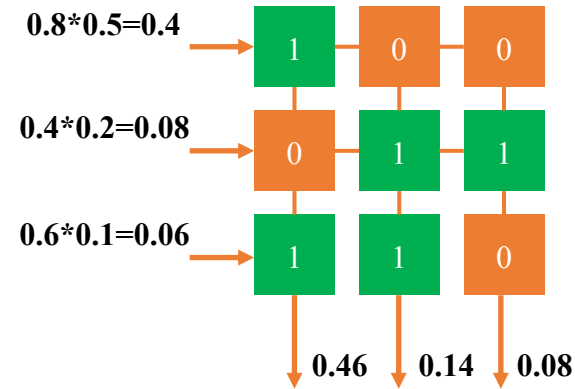


# Opt. I: Single bit implementation

- For algorithms on unweighted graphs
  - e.g., PageRank, BFS, etc.
  - Scatter the same value to neighbors
  - One bit of an edge to represent connectivity



**GraphR**



**GraphSAR**



# Opt. II: lightweight clustering



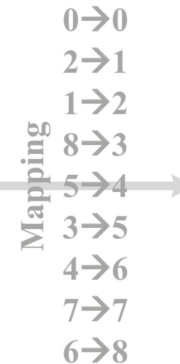
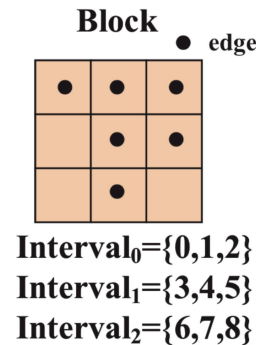
- Vertex clustering  $\rightarrow$  less blocks to be processed
  - Consecutive vertices in the original adjacency list tend to gather
  - Original indices are not continuous
  - Assign new indices to vertices when reading edges
  - Only  $O(n)$  complexity

Original edge list

src	dst	src	dst
3	28	3	371
3	30	3	567
3	39	3	581
3	54	3	584
3	108	3	586
3	152	3	590
3	178	3	604
3	182	3	611
3	214	3	8283
3	271	25	3
3	286	25	6
3	300	25	8
3	348	25	19
3	349	25	23
		25	28

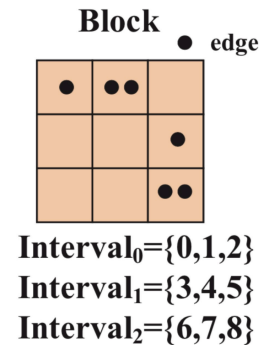
Edge list

- 0 $\rightarrow$ 2
- 1 $\rightarrow$ 8
- 2 $\rightarrow$ 5
- 3 $\rightarrow$ 4
- 4 $\rightarrow$ 7
- 6 $\rightarrow$ 4



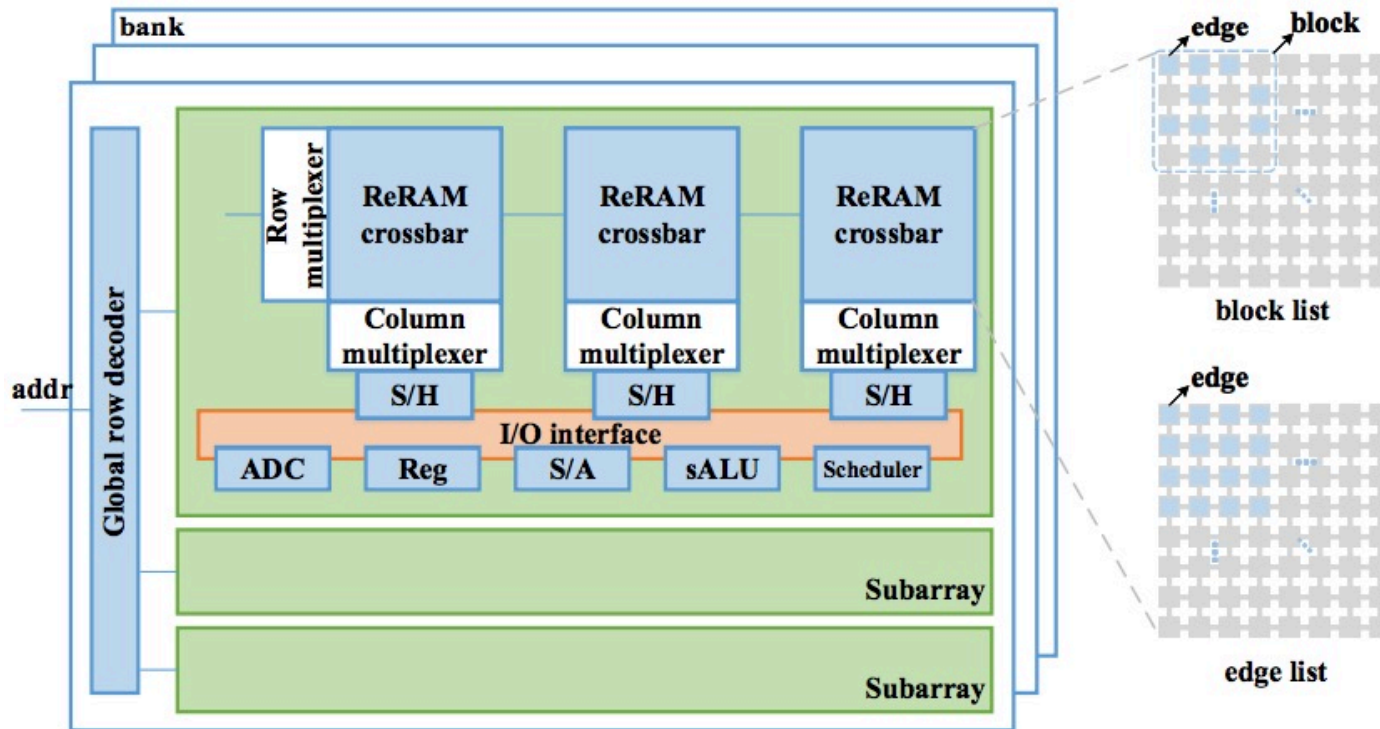
Edge list

- 0 $\rightarrow$ 1
- 2 $\rightarrow$ 3
- 1 $\rightarrow$ 4
- 5 $\rightarrow$ 6
- 6 $\rightarrow$ 7
- 8 $\rightarrow$ 6



# Architecture

- According to the sparsity-aware partitioning
  - Edges are stored into edge lists and block lists
  - Edge lists and block lists are stored into different banks for the alignment purpose

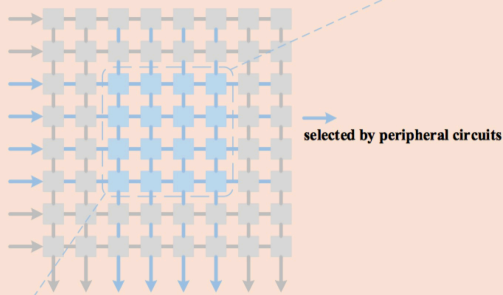




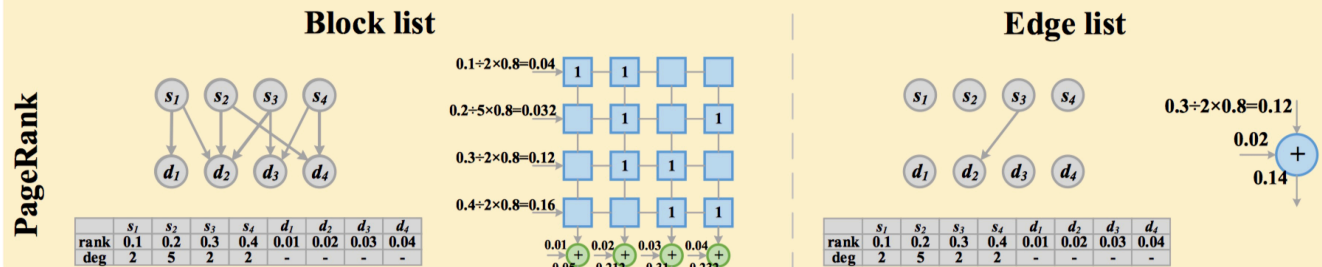
# Working flow

- Selecting: activate a block for processing
  - GraphR: activate a row
- Processing: process edge list and block list separately
  - GraphR: treat a block with one edge as a block

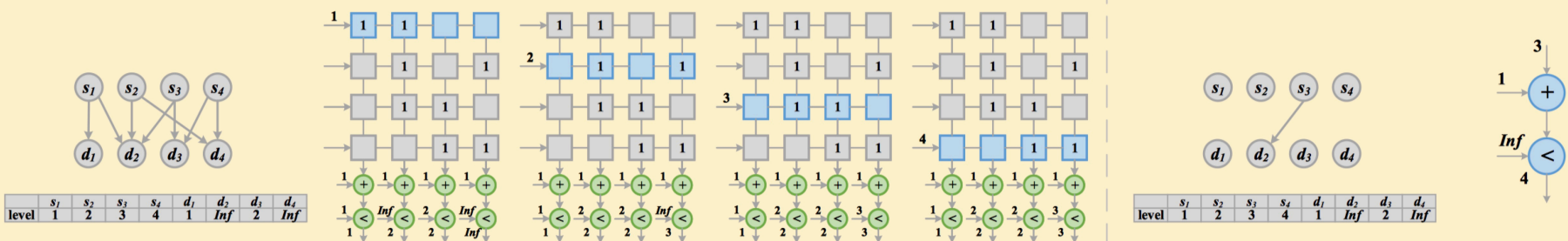
## Selecting Flow



## Processing Flow



## Breadth-First Search





# Content



- Background
- Motivation
- Related Work
- Architecture and Detailed Implementation
- **Experiment Results**
- Conclusion and Future Work



# Configuration

- Datasets

	WV	HP	GG	YT	PK	CA	WT	TW	FS	YH
# V	7.12k	34.5k	0.88m	1.13m	1.63m	1.97m	2.39m	41.7m	65.6m	1.41b
# E	0.10m	0.42m	5.11m	2.99m	30.6m	2.77m	5.02m	1.47b	1.81b	6.64b
type	social	citation	web	community	social	road	communication	social	community	web

- Algorithms

- PageRank, Breadth-first Search, Connected Components

- Configuration

- ReRAM simulator: NVSim

- read/write energy consumption: 1.08pJ/7.4pJ
- read/write latency: 29.31ns/50.88ns
- HRS/LRS resistance: 25M $\Omega$ /50K $\Omega$
- read/write voltage: 0.7V/2V
- current of LRS/HRS: 40 $\mu$ A/2 $\mu$ A

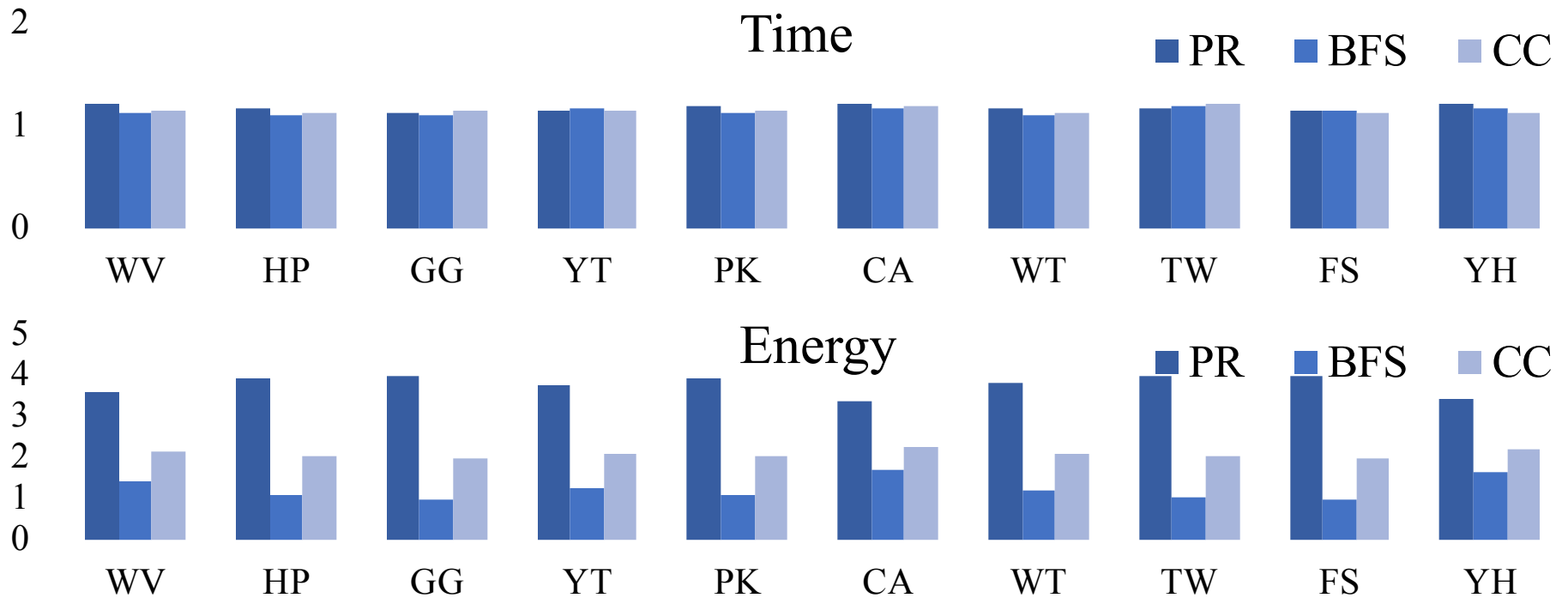
Jure Leskovec et al. SNAP Datasets: Stanford large network dataset collection.  
Haewoon Kwak et al. What is twitter, a social network or a news media?  
Yahoo WebScope. Yahoo! altavista web page hyper-link connectivity graph, circa 2002.



# Results – Opt. I



- Single bit implementation
  - Speedup: 1.15x
  - Energy efficiency improvement: 2.37x
  - Energy-Delay Product reduction: 2.73x



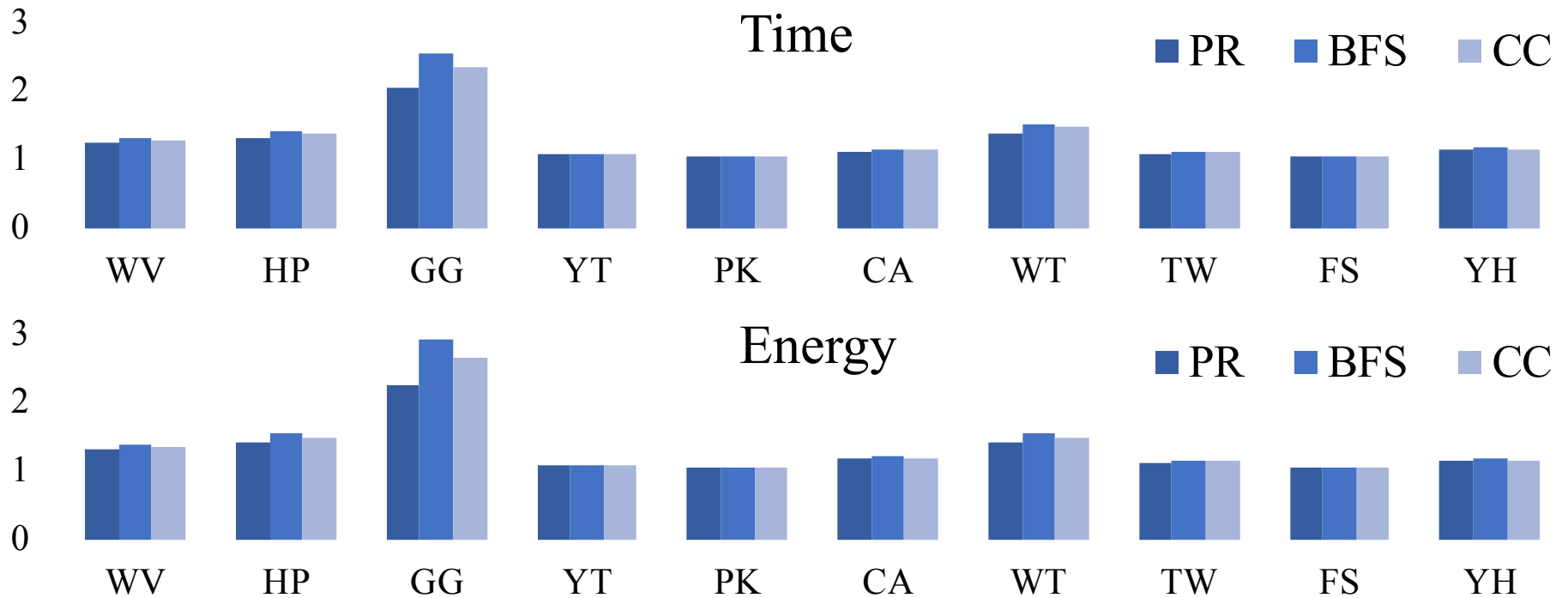




# Results – Opt. II



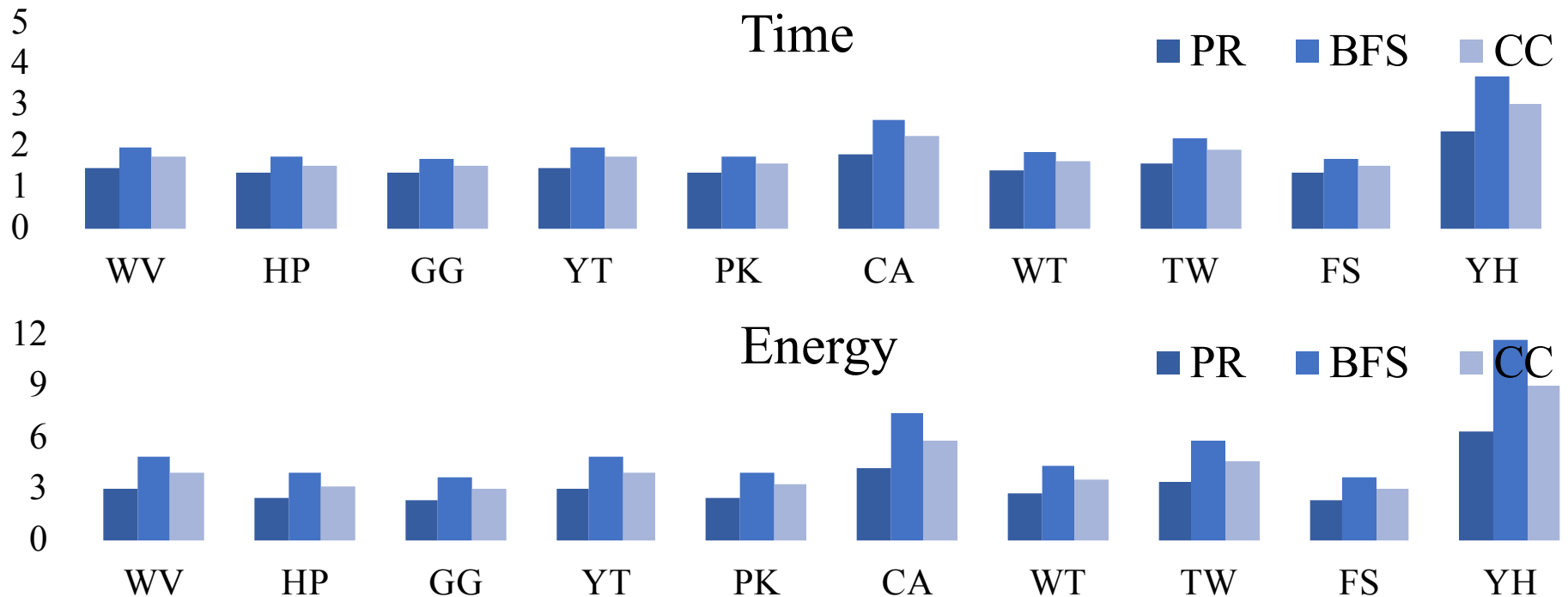
- Lightweight clustering
  - Speedup: 1.30x
  - Energy efficiency improvement: 1.37x
  - Energy-Delay Product reduction: 1.78x





# Results – Overall performance

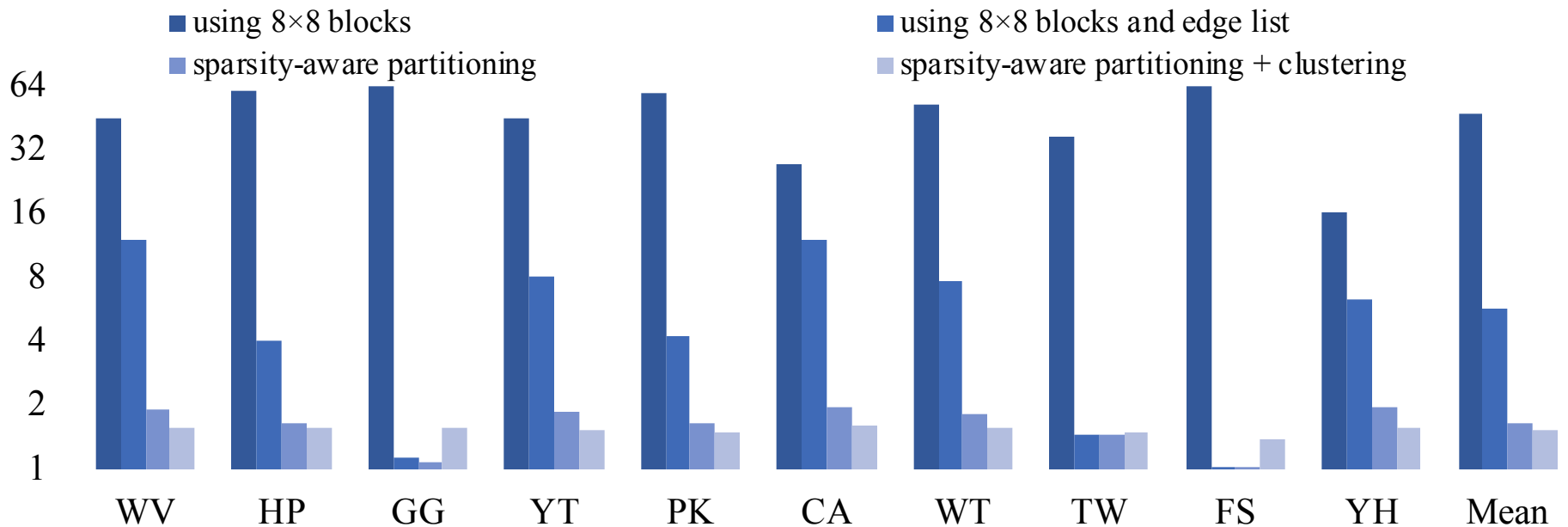
- Compared with GraphR (already used Opt. I & II)
  - Speedup: 1.85x
  - Energy efficiency improvement: 4.43x
  - Energy-Delay Product reduction: 8.19x





# Results – Memory space overhead

- Compared with using adjacency list (need to write ReRAM)
  - Only 1.54x storage overheads
  - 46.87x storage overheads when storing 8\*8 blocks





# Content

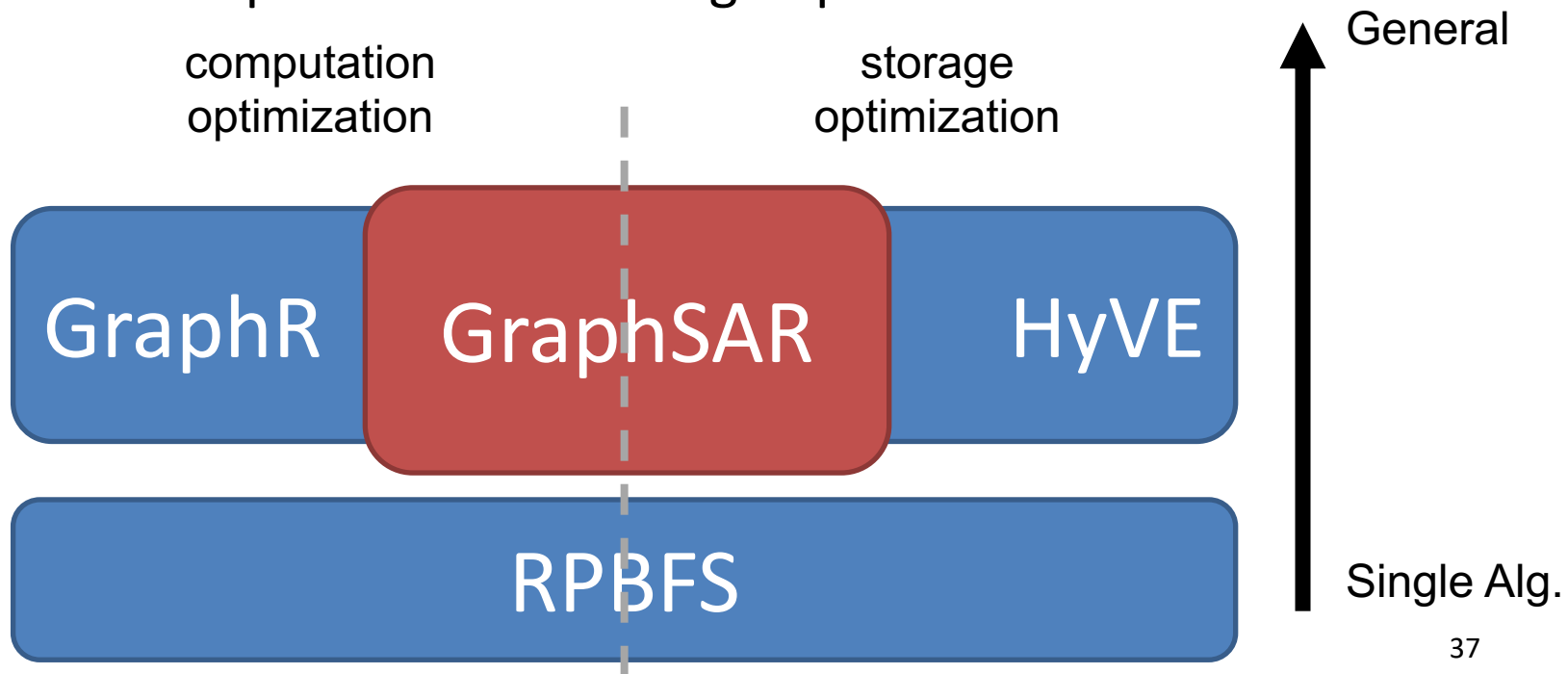


- Background
- Motivation
- Related Work
- Architecture and Detailed Implementation
- Experiment Results
- **Conclusion**



# Conclusion

- GraphSAR
  - Improving energy efficiency/Accelerating graph processing using ReRAM
  - Design for different graph algorithms
  - Both computation and storage optimization





# References

1. Linghao Song et al. GraphE: Accelerating graph processing using reram. In HPCA, pages 531–543. IEEE, 2018.
2. Aapo Kyrola et al. Graphchi: Large-scale graph computation on just a pc. In OSDI, pages 31–46. USENIX, 2012.
3. Amitabha Roy et al. X-stream: Edge-centric graph processing using streaming partitions. In SOSP, pages 472–488. ACM, 2013.
4. Xiaowei Zhu et al. Gridgraph: Large-scale graph processing on a single machine using 2-level hierarchical partitioning. In ATC, pages 375–386. USENIX, 2015.
5. Yuze Chi et al. Nxgraph: An efficient graph processing system on a single machine. In ICDE, pages 409–420. IEEE, 2016.
6. Grzegorz Malewicz et al. Pregel: a system for large-scale graph processing. In SIGMOD, pages 135–146. ACM, 2010.
7. Yucheng Low et al. Distributed GraphLab: a framework for machine learning and data mining in the cloud. Proceedings of the VLDB Endowment, 5(8):716–727, 2012.
8. Tae Jun Ham et al. Graphicionado: A high-performance and energy-efficient accelerator for graph analytics. In MICRO, pages 1–13. IEEE, 2016.
9. Fabien Alibart et al. High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm. Nanotechnology, 23(7):075201, 2012.
10. Cong Xu et al. Overcoming the challenges of crossbar resistive memory architectures. In HPCA, pages 476–488. IEEE, 2015.



# References

11. Lei Han et al. A novel reram-based processing-in-memory architecture for graph computing. In NVMSA, pages 1–6. IEEE, 2017.
12. Tianhao Huang et al. Hyve: Hybrid vertex-edge memory hierarchy for energyefficient graph processing. In DATE. EDA Consortium, 2018.
13. Cong Xu et al. Understanding the trade-offs in multi-level cell reram memory design. In DAC, pages 1–6. IEEE, 2013.
14. Xiaowei Zhu et al. Gemini: A computation-centric distributed graph processing system. In OSDI, pages 301–316. USENIX, 2016.
15. Dimin Niu et al. Design of cross-point metal-oxide reram emphasizing reliability and cost. In ICCAD, pages 17–23. IEEE, 2013.
16. Jure Leskovec et al. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, June 2014.
17. Haewoon Kwak et al. What is twitter, a social network or a news media? In WWW, pages 591–600. ACM, 2010.
18. Yahoo WebScope. Yahoo! altavista web page hyper-link connectivity graph, circa 2002. <http://webscope.sandbox.yahoo.com/>, 2012.
19. Xiangyu Dong et al. Nvsim: A circuit-level performance, energy, and area model for emerging non-volatile memory. In Emerging Memory Technologies, pages 15–50. Springer, 2014.



# References

20. Shimeng Yu et al. Investigating the switching dynamics and multilevel capability of bipolar metal oxide resistive switching memory. *Applied Physics Letters*, 98(10):103514, 2011.
21. Steven JE Wilton and Norman P Jouppi. Cacti: An enhanced cache access and cycle time model. *JSSC*, 31(5):677–688, 1996.
22. Boris Murmann. ADC performance survey 1997-2017. <http://web.stanford.edu/~murmurmann/adcsurvey.html>, August 2017.





# Thank you!

## Q & A