Accelerating Dynamic Graph Analytics on GPUs

Mo Sha[†], Yuchen Li[‡], Bingsheng He[†], Kian-Lee Tan[†]

[†]School of Computing, National University of Singapore
[‡]School of Information Systems, Singapore Management University

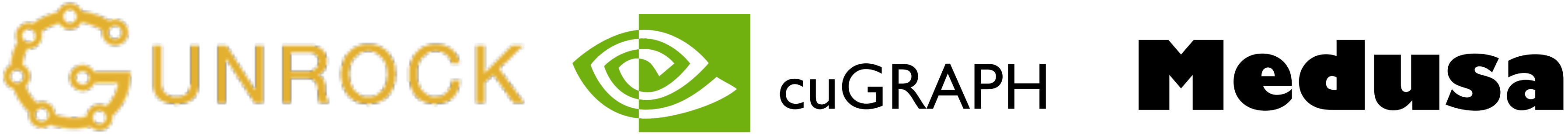
Presenter: Mo Sha
Aug 29th, 2018

Outline

- Introduction
- GPMA Dynamic Graph Processing
- GPMA+ Optimization
- Evaluation
- Conclusion

Introduction

GPU Accelerated Graph Analytics

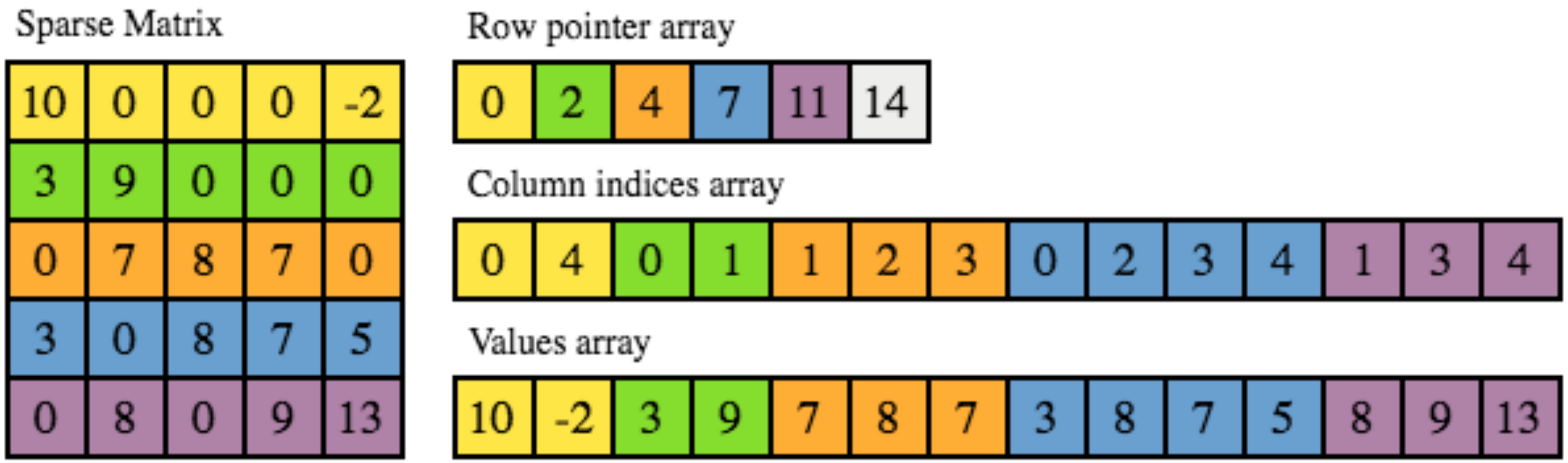


[1] https://gunrock.github.io
[2] https://developer.nvidia.com/nvgraph
[3] https://github.com/pymedusa/Medusa

Introduction

GPU Accelerated Graph Analytics

- Existing GPU Accelerated Graph Analytics focus on conducting graph primitives on static graphs.
- CSR(-like) graph format is widely used due to GPU computational architecture for better parallelism scheduling.



Sparse Matrix

10	0	0	0	-2
3	9	0	0	0
0	7	8	7	0
3	0	8	7	5
0	8	0	9	13

Row pointer array

0	2	4	7	11	14

Column indices array

0	4	0	1	1	2	3	0	2	3	4	1	3	4

Values array

10	-2	3	9	7	8	7	3	8	7	5	8	9	13

Introduction

GPU Accelerated Graph Analytics

- Existing GPU Accelerated Graph Analytics focus on conducting graph primitives on static graphs.
- CSR(-like) graph format is widely used due to GPU computational architecture for better parallelism scheduling.
- For evolving graphs, an entire rebuilding is required for each single update, which significantly limits the performance on evolving graphs.

Introduction

Dynamic Graph Format on GPUs requires …

- To support edge insertion, deletion, and edge value modification.
- To be compatible with existing GPU-based graph algorithms.
- An acceptable trade-off between update efficiency and the overhead introduced.

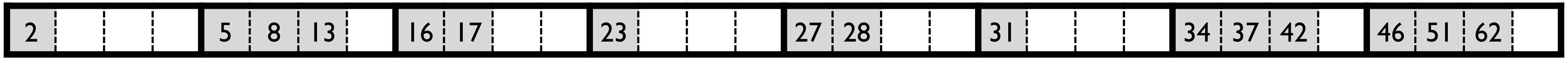
Outline

- Introduction
- GPMA Dynamic Graph Processing
- GPMA+ Optimization
- Evaluation
- Conclusion

GPMA Dynamic Graph Processing

Introduce Packed Memory Array into GPU Graph Analytics

- Sort elements in a partially continuous fashion
- Leave gaps on each segment with a bounded ratio
- Self-balancing binary tree structure
- The amortized insertion complexity of PMA is proved to be $O(\log^2 N)$ in the worst case and $O(\log N)$ in the average case.

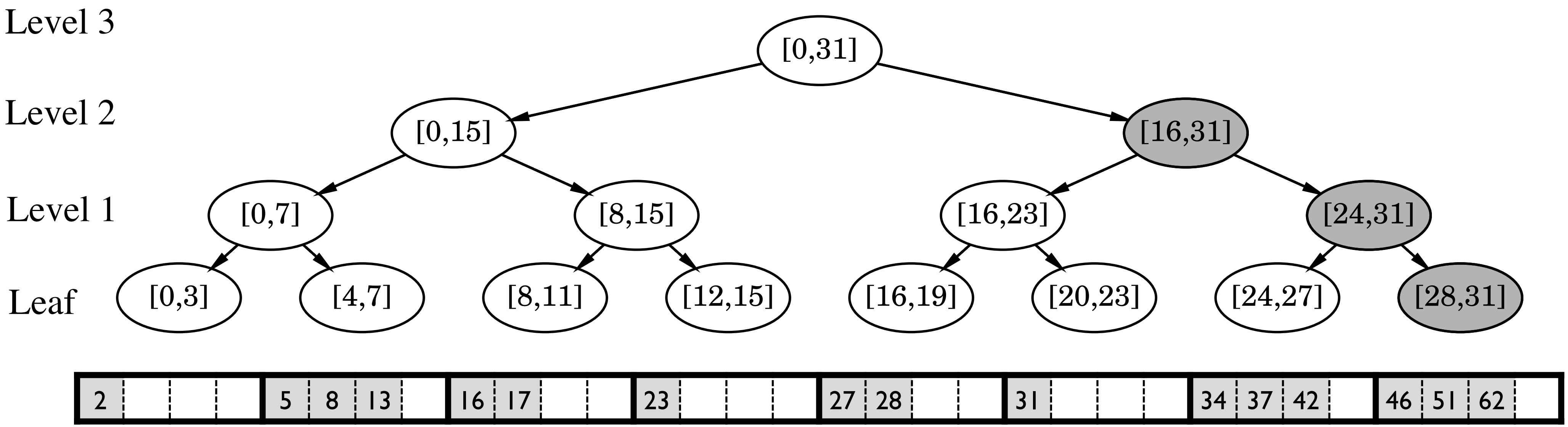


2				5	8	13		16	17			23				27	28			31				34	37	42		46	51	62	

[1] Bender, M. A., Demaine, E. D., & Farach-Colton, M. Cache-oblivious b-trees. SIAM J. Comput., 35(2):341–358, 2005.
[2] M. A. Bender and H. Hu. An adaptive packed-memory array. ACM Trans. Database Syst., 32(4), 2007.

GPMA Dynamic Graph Processing

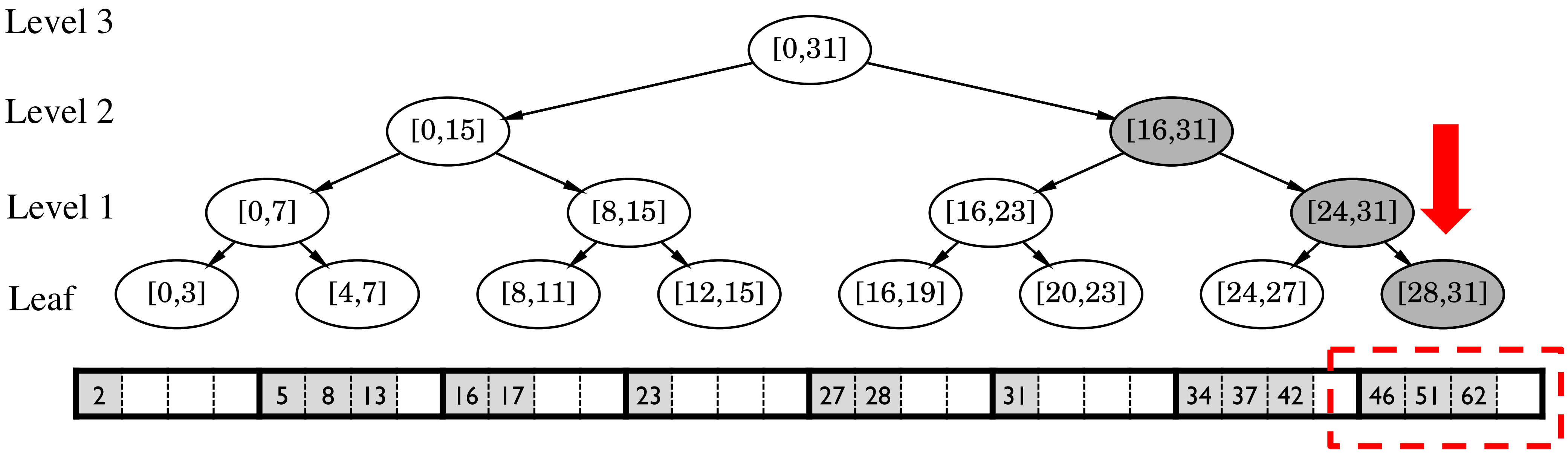
	Leaf	Level 1	Level 2	Level 3
segment size	4	8	16	32
density lower bound ρ	0.08	0.19	0.29	0.40
density upper bound τ	0.92	0.88	0.84	0.80
min # of entries	1	2	4	8
max # of entries	3	6	12	24



[1] Bender, M. A., Demaine, E. D., & Farach-Colton, M. Cache-oblivious b-trees. SIAM J. Comput., 35(2):341–358, 2005.
[2] M. A. Bender and H. Hu. An adaptive packed-memory array. ACM Trans. Database Syst., 32(4), 2007.

GPMA Dynamic Graph Processing

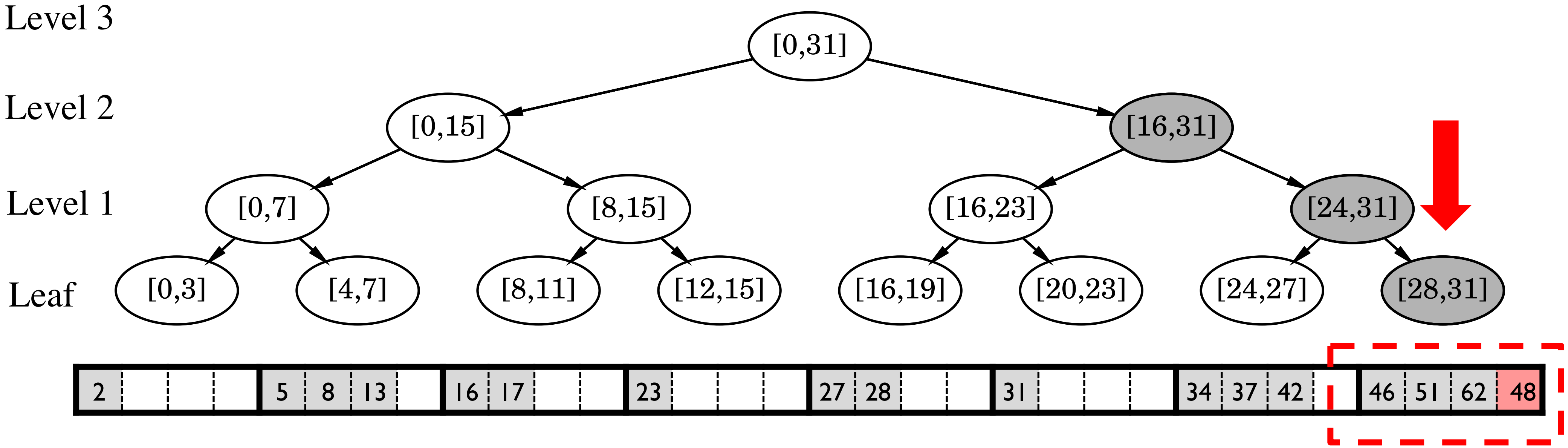
	Leaf	Level 1	Level 2	Level 3
segment size	4	8	16	32
density lower bound ρ	0.08	0.19	0.29	0.40
density upper bound τ	0.92	0.88	0.84	0.80
min # of entries	1	2	4	8
max # of entries	3	6	12	24



[1] Bender, M. A., Demaine, E. D., & Farach-Colton, M. Cache-oblivious b-trees. SIAM J. Comput., 35(2):341–358, 2005.
[2] M. A. Bender and H. Hu. An adaptive packed-memory array. ACM Trans. Database Syst., 32(4), 2007.

GPMA Dynamic Graph Processing

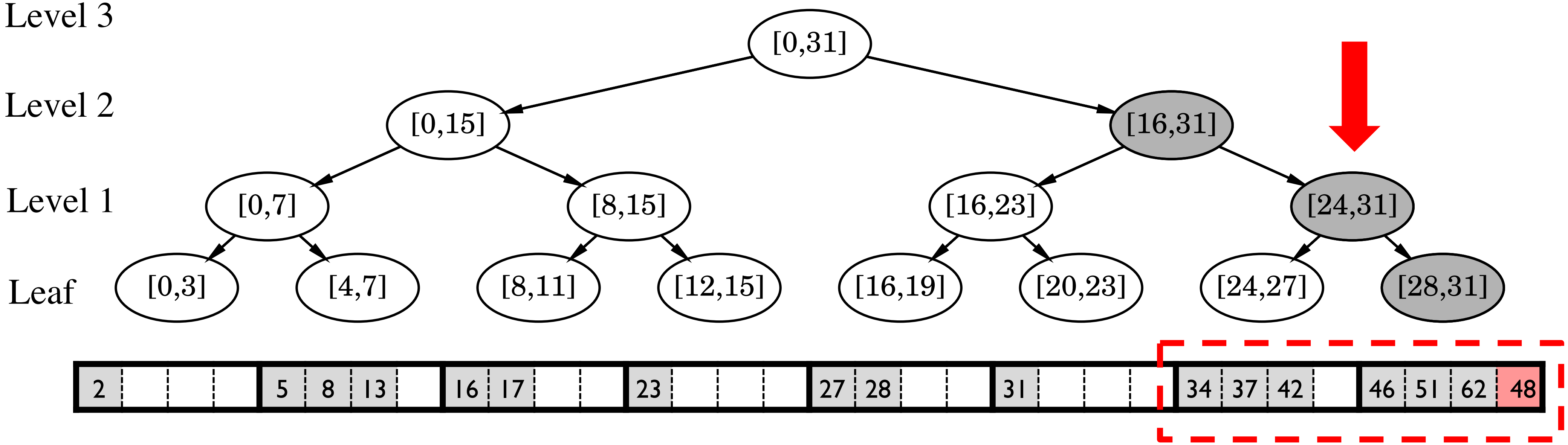
	Leaf	Level 1	Level 2	Level 3
segment size	4	8	16	32
density lower bound ρ	0.08	0.19	0.29	0.40
density upper bound τ	0.92	0.88	0.84	0.80
min # of entries	1	2	4	8
max # of entries	3	6	12	24



[1] Bender, M. A., Demaine, E. D., & Farach-Colton, M. Cache-oblivious b-trees. SIAM J. Comput., 35(2):341–358, 2005.
[2] M. A. Bender and H. Hu. An adaptive packed-memory array. ACM Trans. Database Syst., 32(4), 2007.

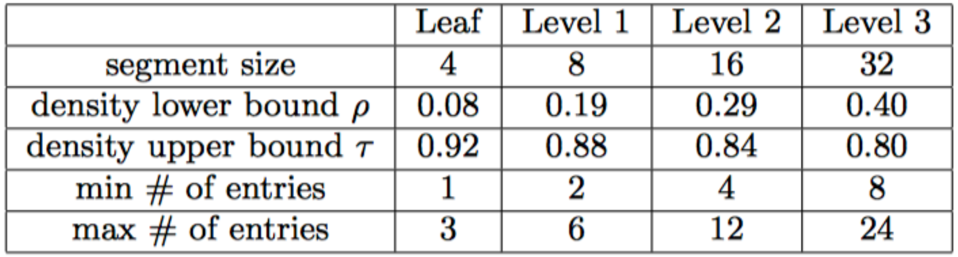
GPMA Dynamic Graph Processing

	Leaf	Level 1	Level 2	Level 3
segment size	4	8	16	32
density lower bound ρ	0.08	0.19	0.29	0.40
density upper bound τ	0.92	0.88	0.84	0.80
min # of entries	1	2	4	8
max # of entries	3	6	12	24

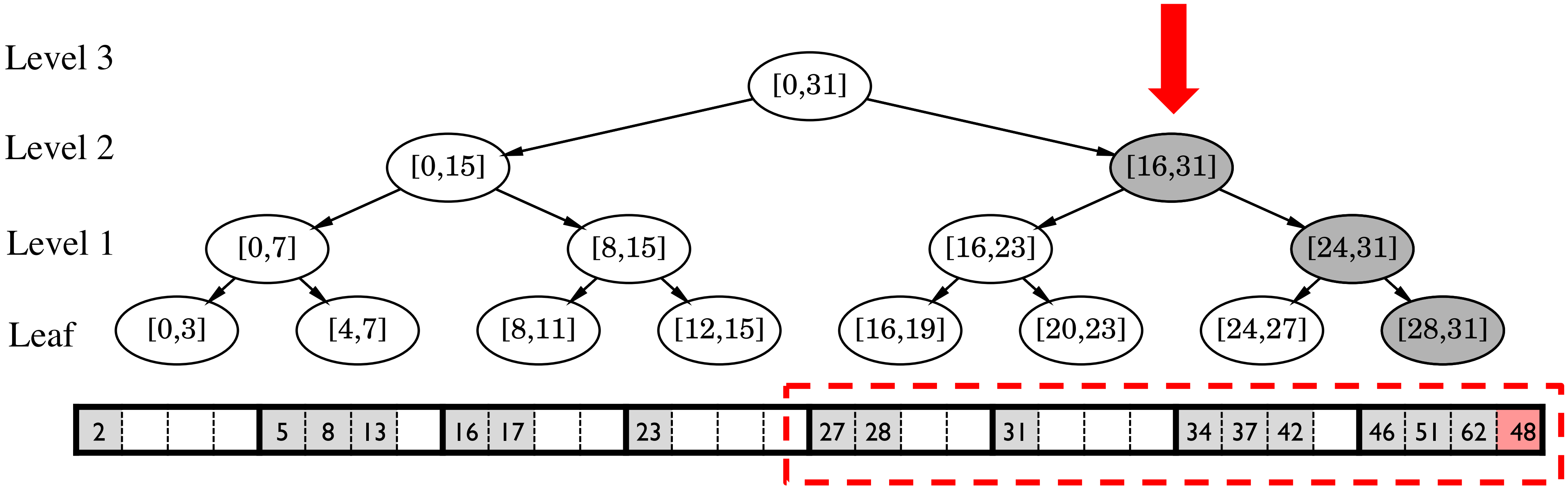


[1] Bender, M. A., Demaine, E. D., & Farach-Colton, M. Cache-oblivious b-trees. SIAM J. Comput., 35(2):341–358, 2005.
[2] M. A. Bender and H. Hu. An adaptive packed-memory array. ACM Trans. Database Syst., 32(4), 2007.

GPMA Dynamic Graph Processing



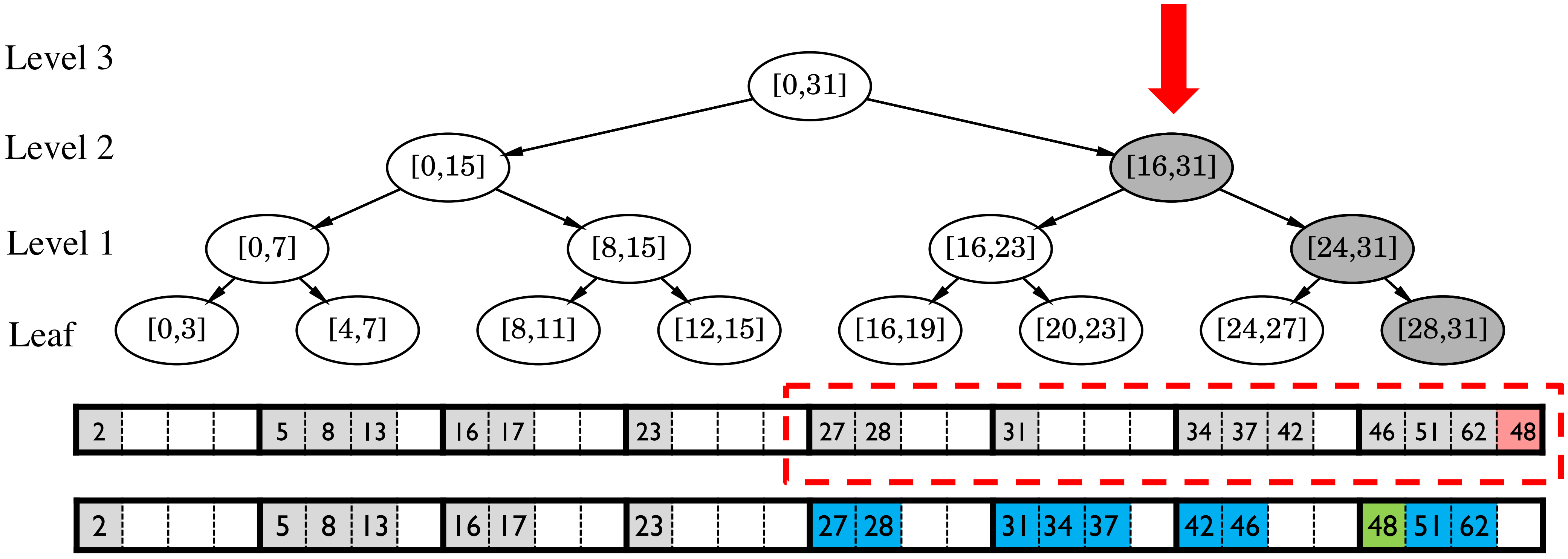
	Leaf	Level 1	Level 2	Level 3
segment size	4	8	16	32
density lower bound ρ	0.08	0.19	0.29	0.40
density upper bound τ	0.92	0.88	0.84	0.80
min # of entries	1	2	4	8
max # of entries	3	6	12	24



[1] Bender, M. A., Demaine, E. D., & Farach-Colton, M. Cache-oblivious b-trees. SIAM J. Comput., 35(2):341–358, 2005.
[2] M. A. Bender and H. Hu. An adaptive packed-memory array. ACM Trans. Database Syst., 32(4), 2007.

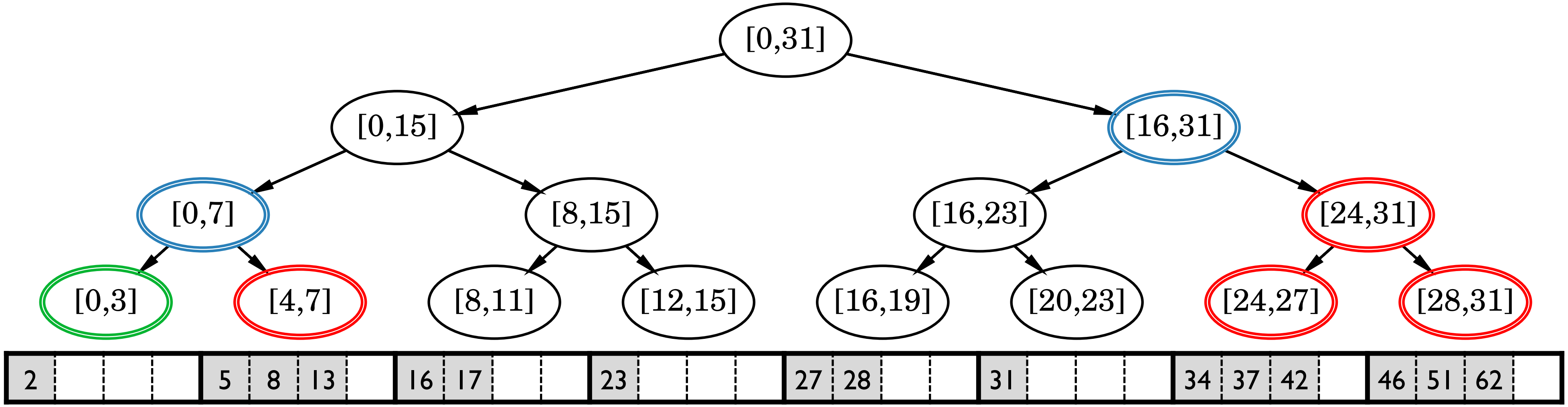
GPMA Dynamic Graph Processing

	Leaf	Level 1	Level 2	Level 3
segment size	4	8	16	32
density lower bound ρ	0.08	0.19	0.29	0.40
density upper bound τ	0.92	0.88	0.84	0.80
min # of entries	1	2	4	8
max # of entries	3	6	12	24



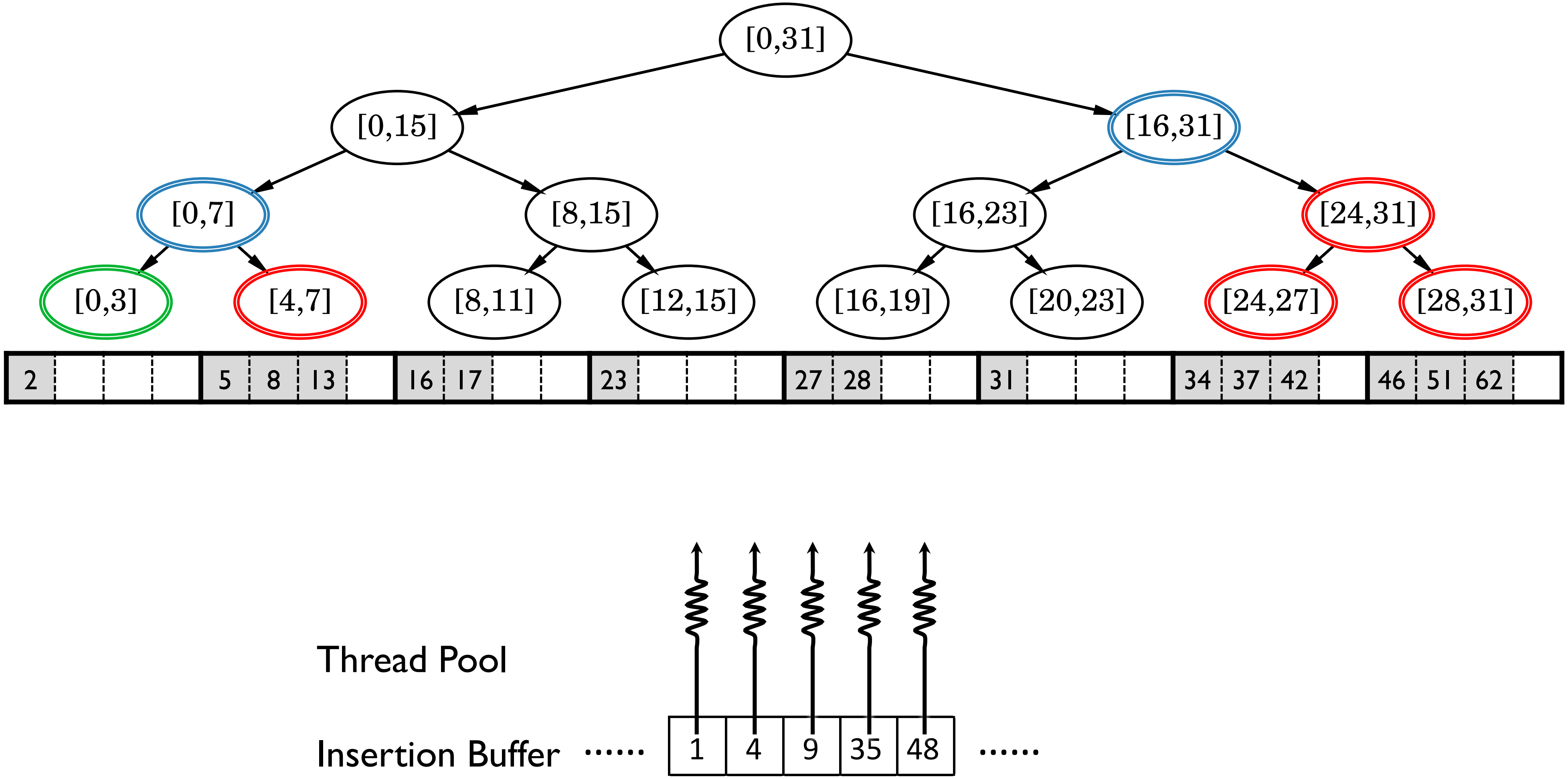
[1] Bender, M. A., Demaine, E. D., & Farach-Colton, M. Cache-oblivious b-trees. SIAM J. Comput., 35(2):341–358, 2005.
[2] M. A. Bender and H. Hu. An adaptive packed-memory array. ACM Trans. Database Syst., 32(4), 2007.

GPMA Dynamic Graph Processing

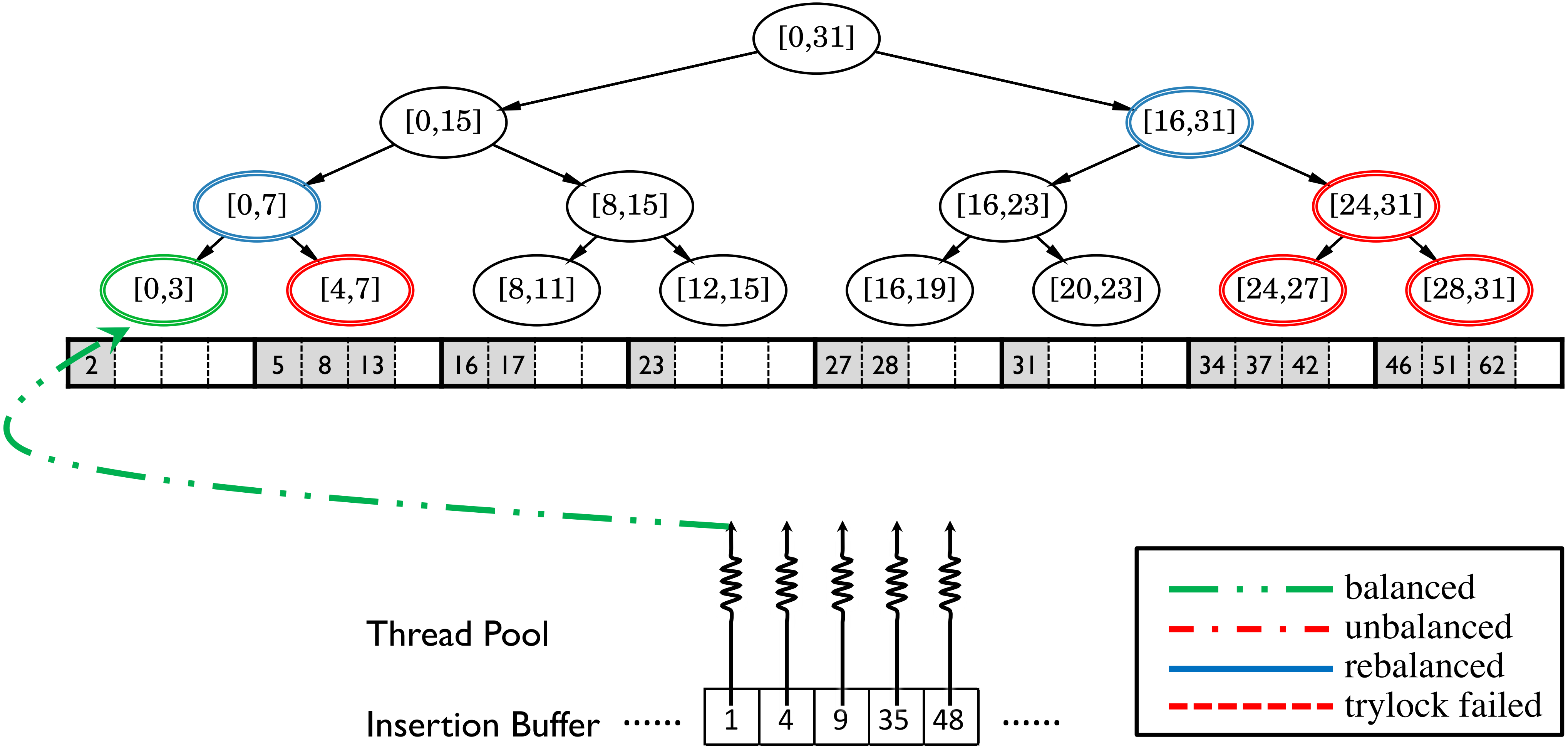


2				5	8	13		16	17			23				27	28			31				34	37	42		46	51	62	

GPMA Dynamic Graph Processing



GPMA Dynamic Graph Processing

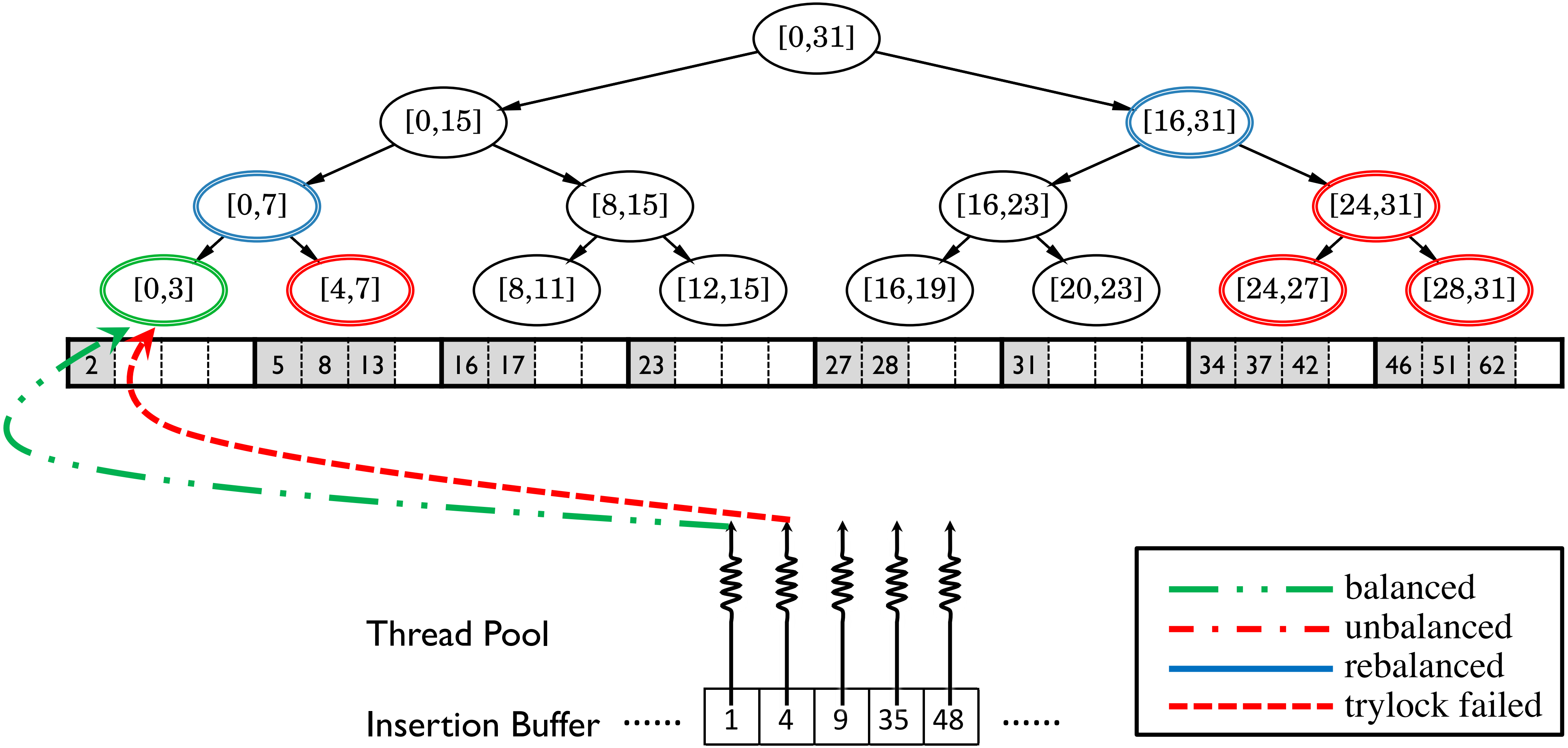


Thread Pool

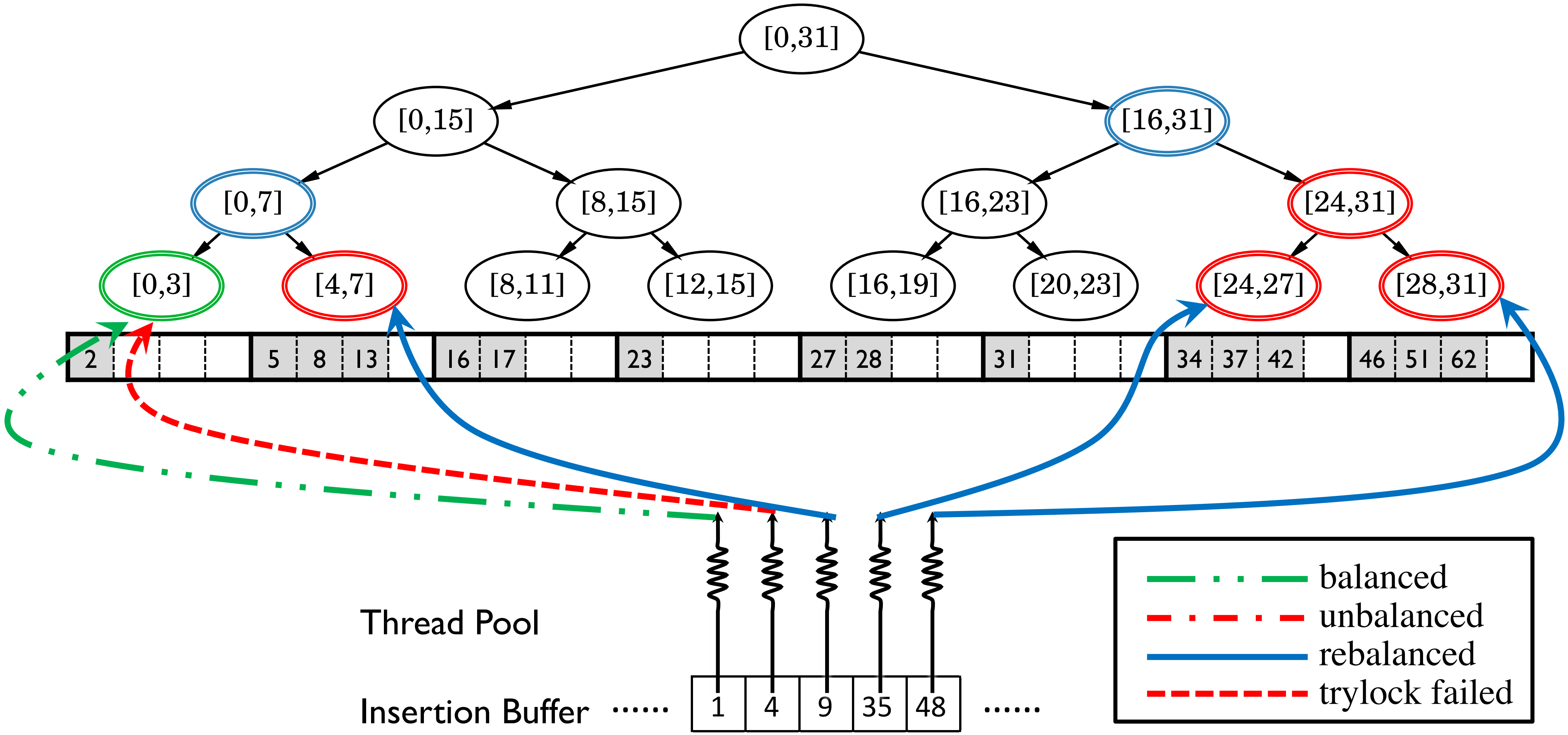
Insertion Buffer

balanced
unbalanced
rebalanced
trylock failed

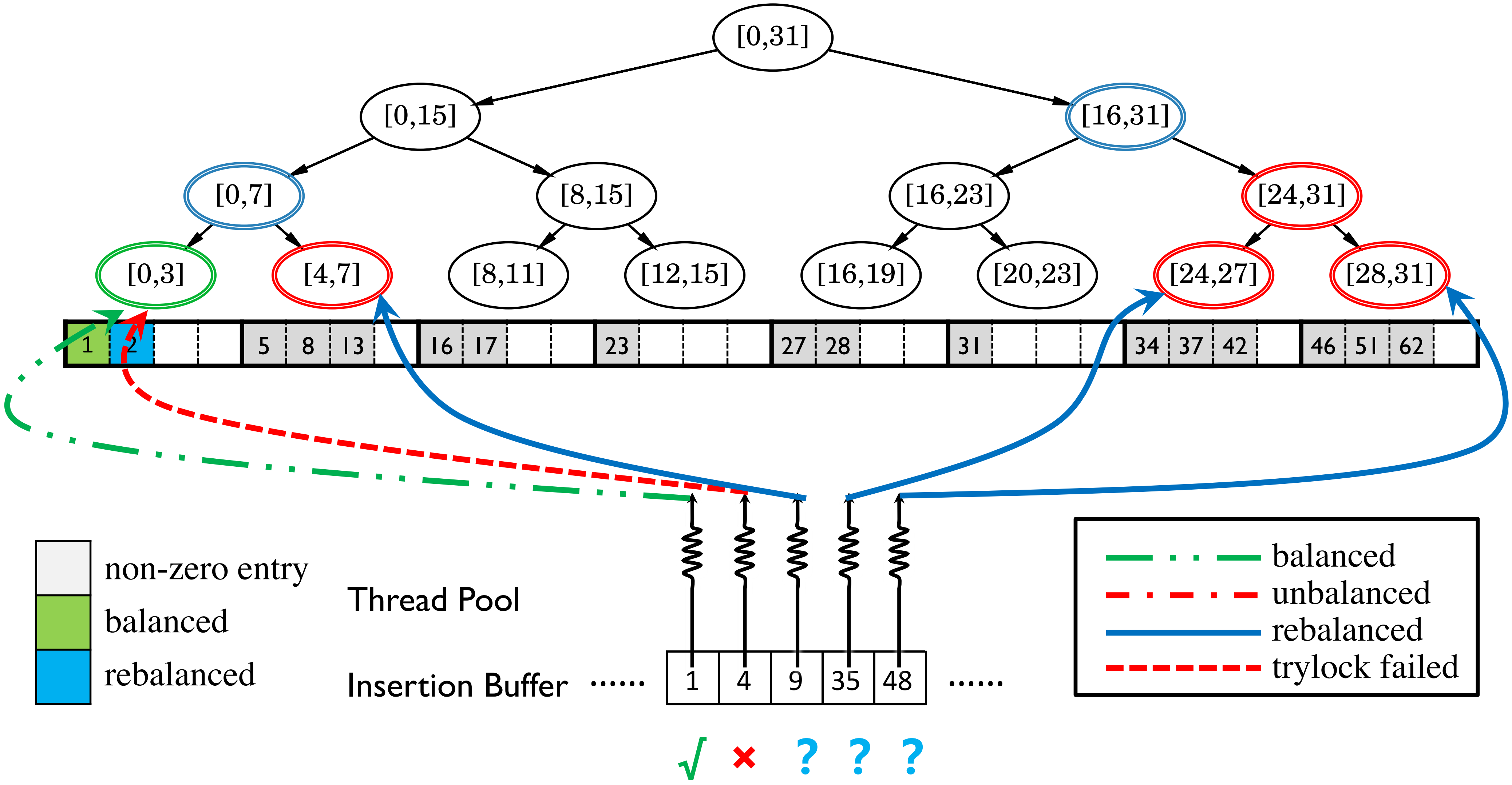
GPMA Dynamic Graph Processing



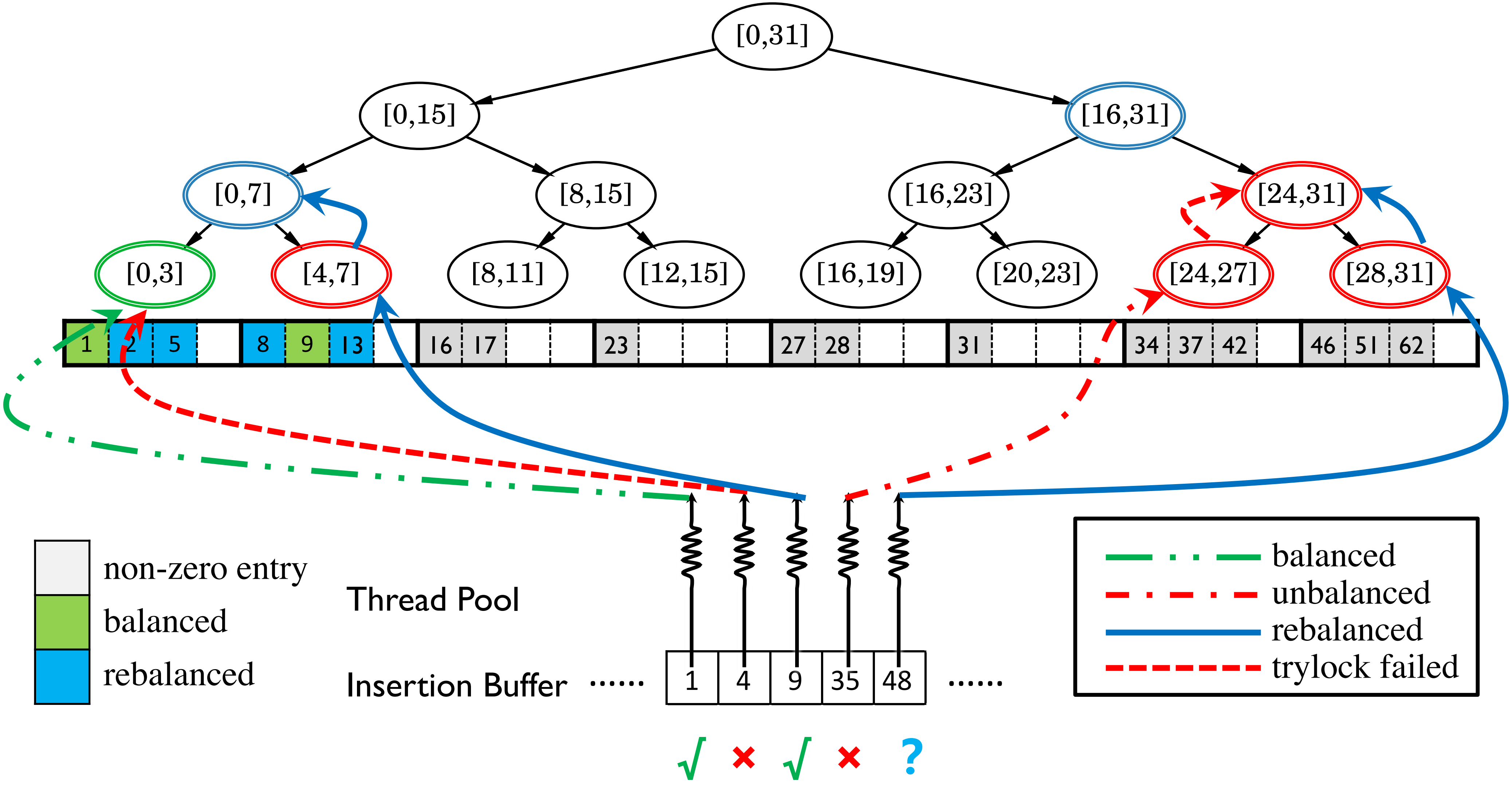
GPMA Dynamic Graph Processing



GPMA Dynamic Graph Processing



GPMA Dynamic Graph Processing



⬜ non-zero entry

🟩 balanced

🟦 rebalanced

Thread Pool

Insertion Buffer ······ 1 | 4 | 9 | 35 | 48 ······

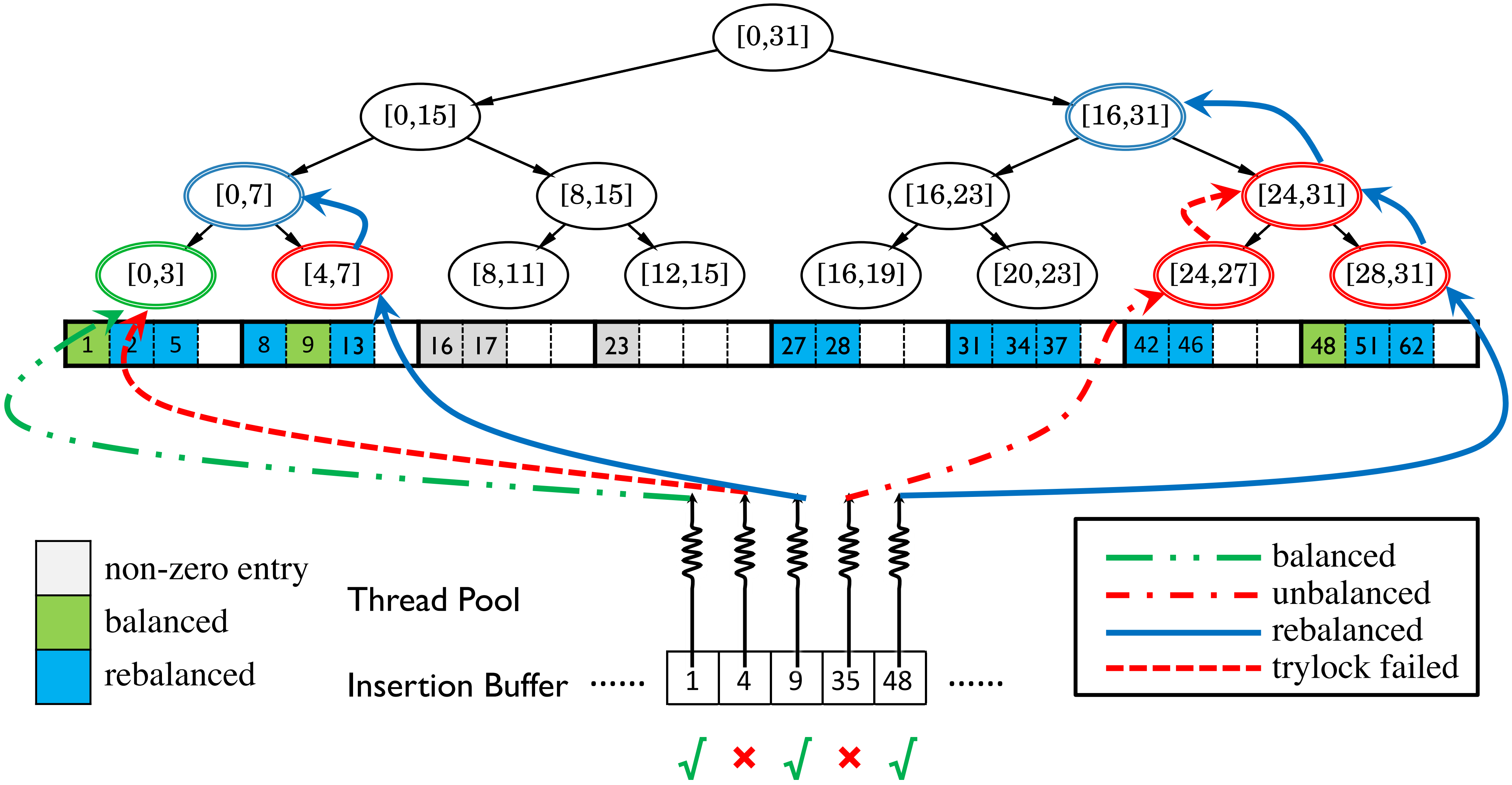
— · · — balanced

— · — unbalanced

—— rebalanced

- - - trylock failed

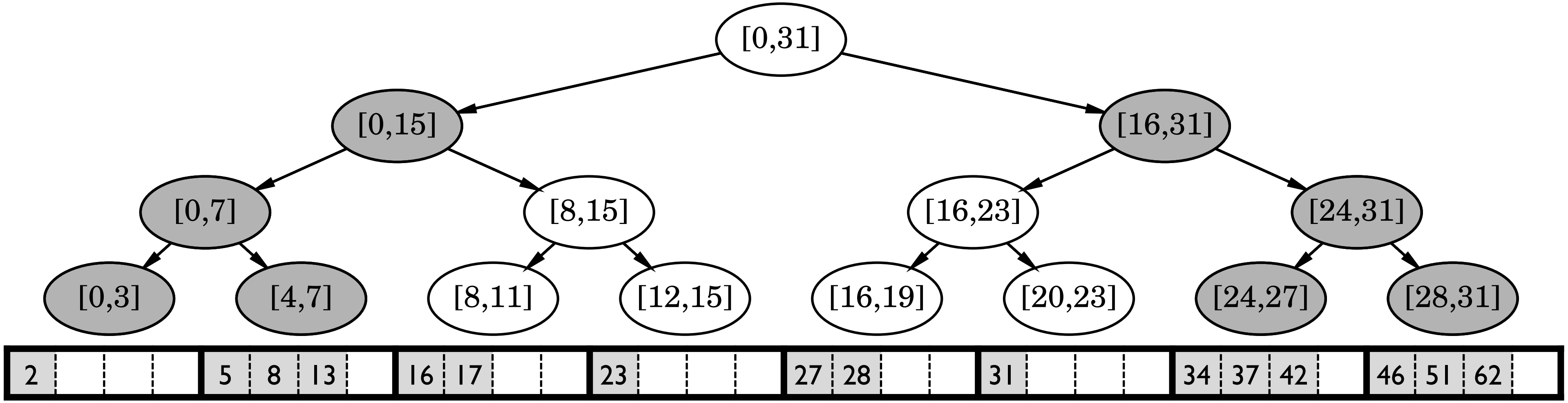
GPMA Dynamic Graph Processing



Outline

- Introduction
- GPMA Dynamic Graph Processing
- GPMA+ Optimization
- Evaluation
- Conclusion

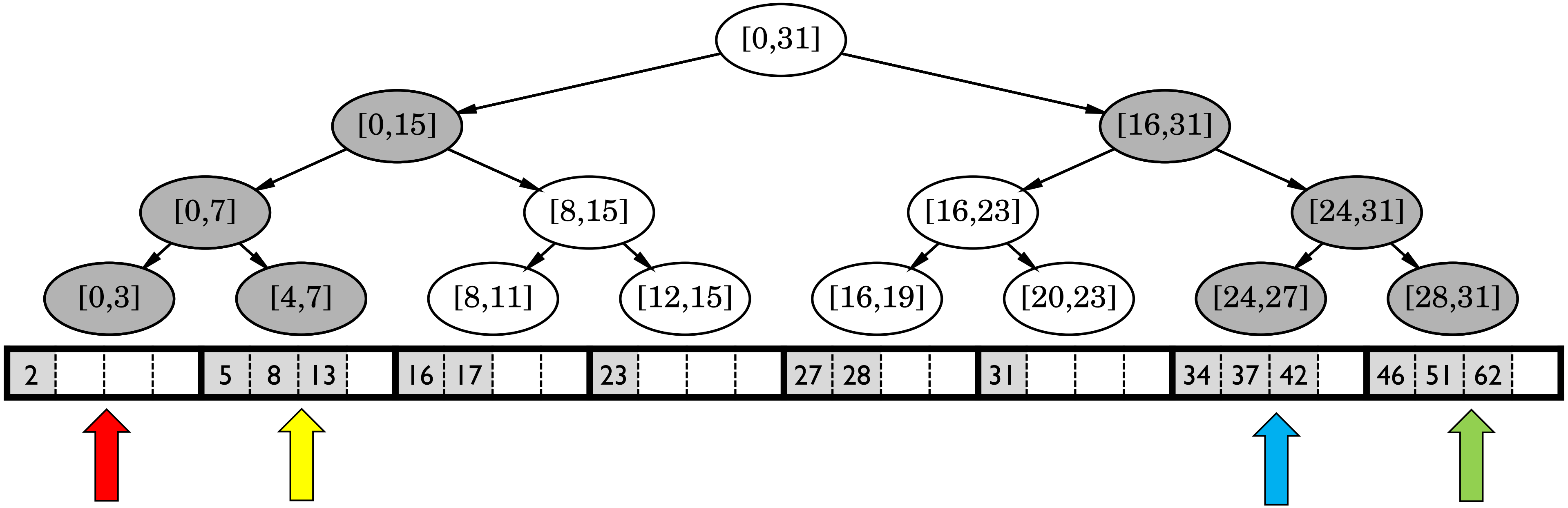
GPMA+ Optimization



Insertion Keys:

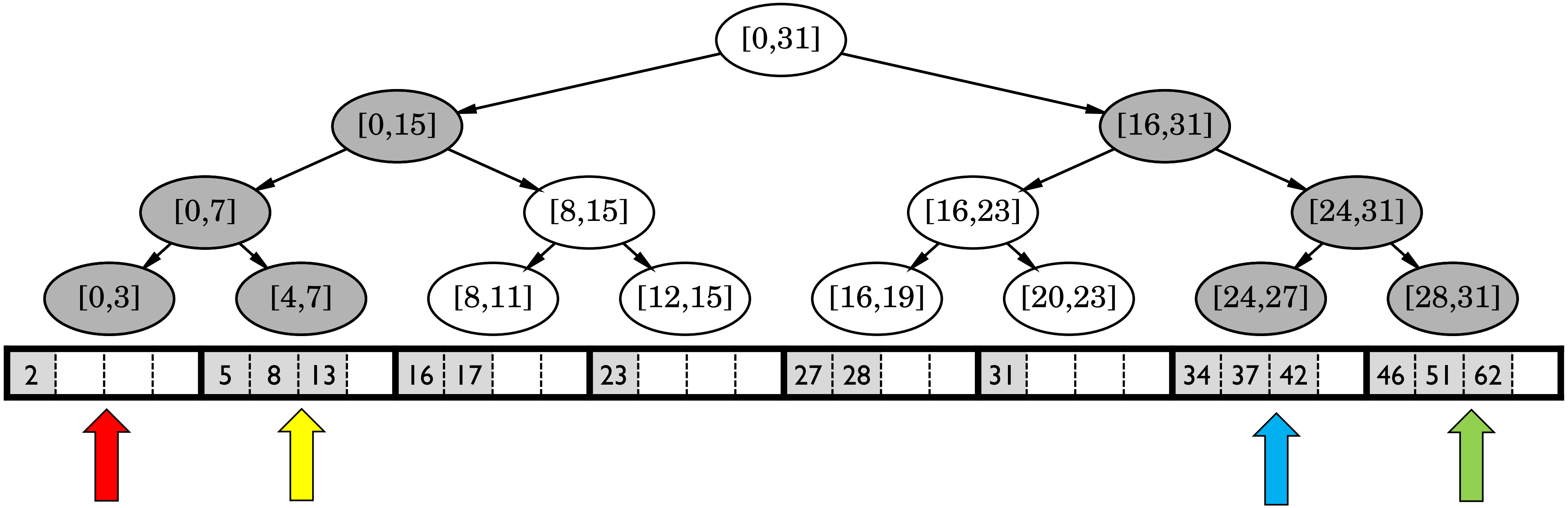
1	4	9	35	48

GPMA+ Optimization



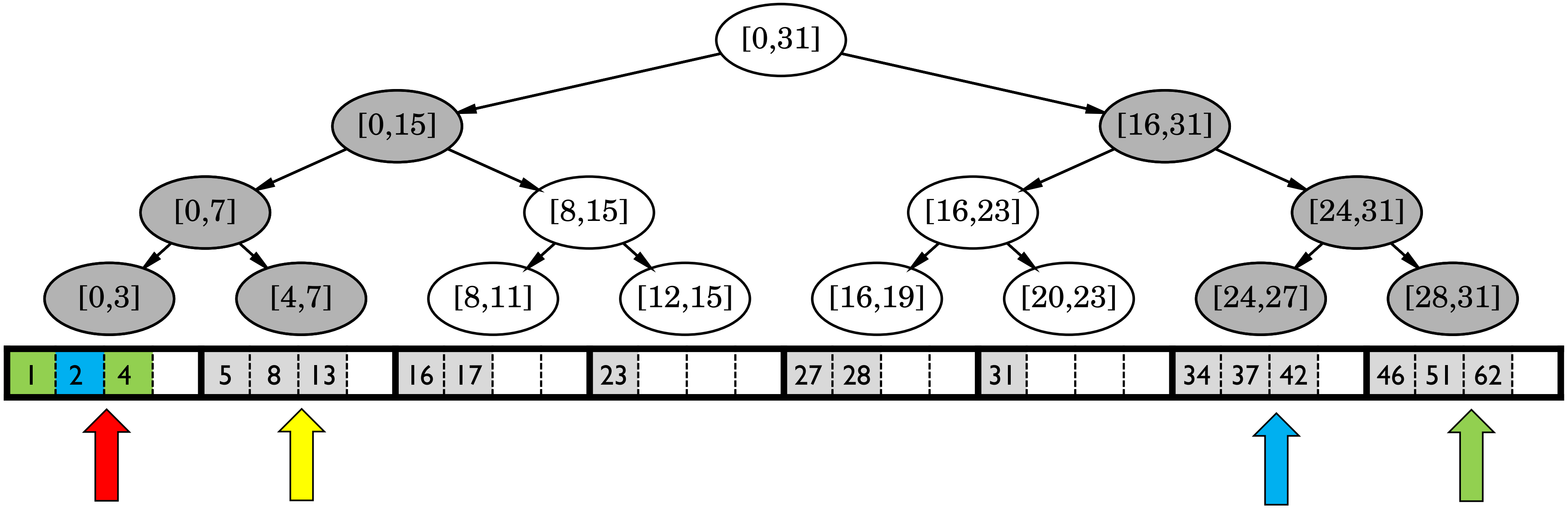
Insertion Keys:	1	4	9	35	48
Leaf Segments:	0	0	4	24	28

GPMA+ Optimization



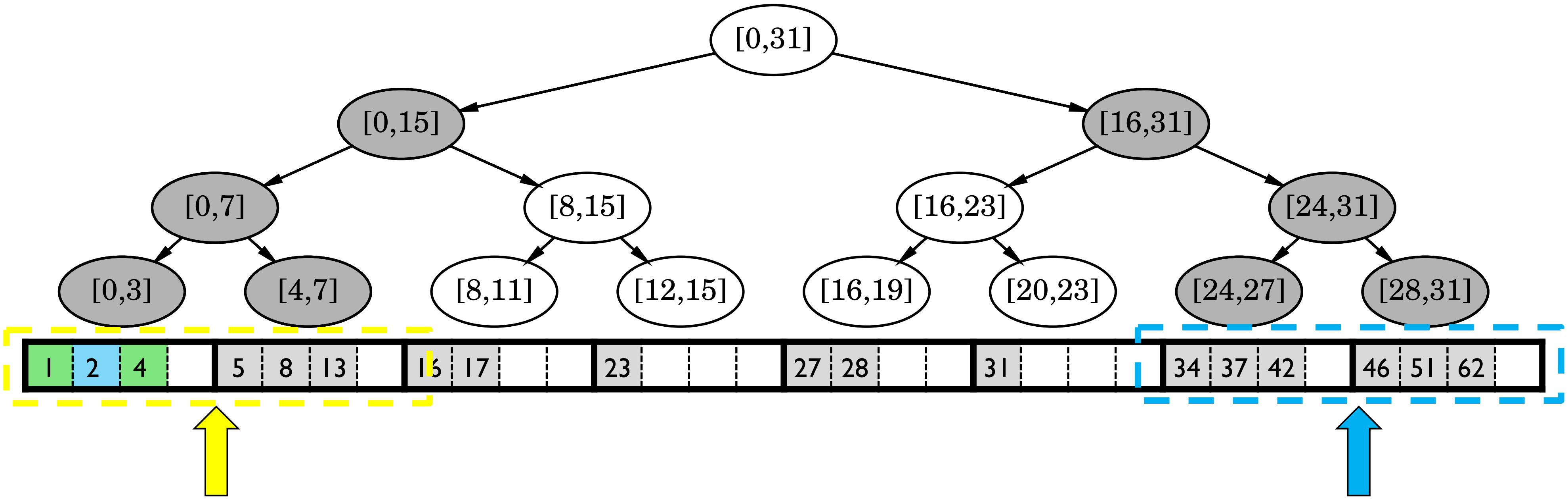
Insertion Keys:	1	4	9	35	48
Leaf Segments:	0	0	4	24	28
Update Segments:	0	4	24	28	
Update Offsets:	2	3	4	5	

GPMA+ Optimization



Insertion Keys:	1	4	9	35	48
Leaf Segments:	0	0	4	24	28
Update Segments:	0	4	24	28	
Update Offsets:	2	3	4	5	
Successful Flag:	Y	N	N	N	

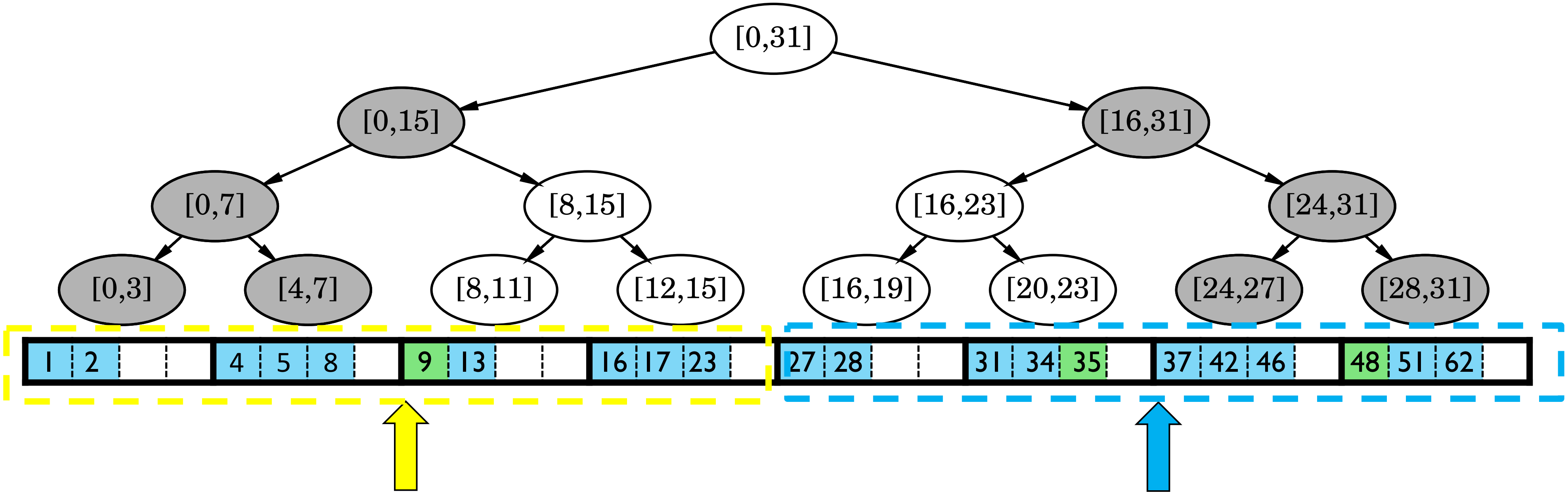
GPMA+ Optimization



Insertion Keys:	1	4	9	35	48
Leaf Segments:	0	0	4	24	28
Update Segments:	0	4	24	28	
Update Offsets:	2	3	4	5	
Successful Flag:	Y	N	N	N	

Insertion Keys:	9	35	48
Update Segments:	0	24	
Update Offsets:	1	3	
Successful Flag:	N	N	

GPMA+ Optimization



Insertion Keys:	1	4	9	35	48
Leaf Segments:	0	0	4	24	28
Update Segments:	0	4	24	28	
Update Offsets:	2	3	4	5	
Successful Flag:	Y	N	N	N	

Insertion Keys:	9	35	48
Update Segments:	0	24	
Update Offsets:	1	3	
Successful Flag:	N	N	

Insertion Keys:	9	35	48
Update Segments:	0	16	
Update Offsets:	1	3	
Successful Flag:	Y	Y	

Outline

- Introduction
- GPMA Dynamic Graph Processing
- GPMA+ Optimization
- Evaluation
- Conclusion

Evaluation

Datasets

	|V|	|E|
Reddit	2.61M	34.4M
Pokec	1.6M	30.6M
Graph500	1M	200M
Random	1M	200M

Specification

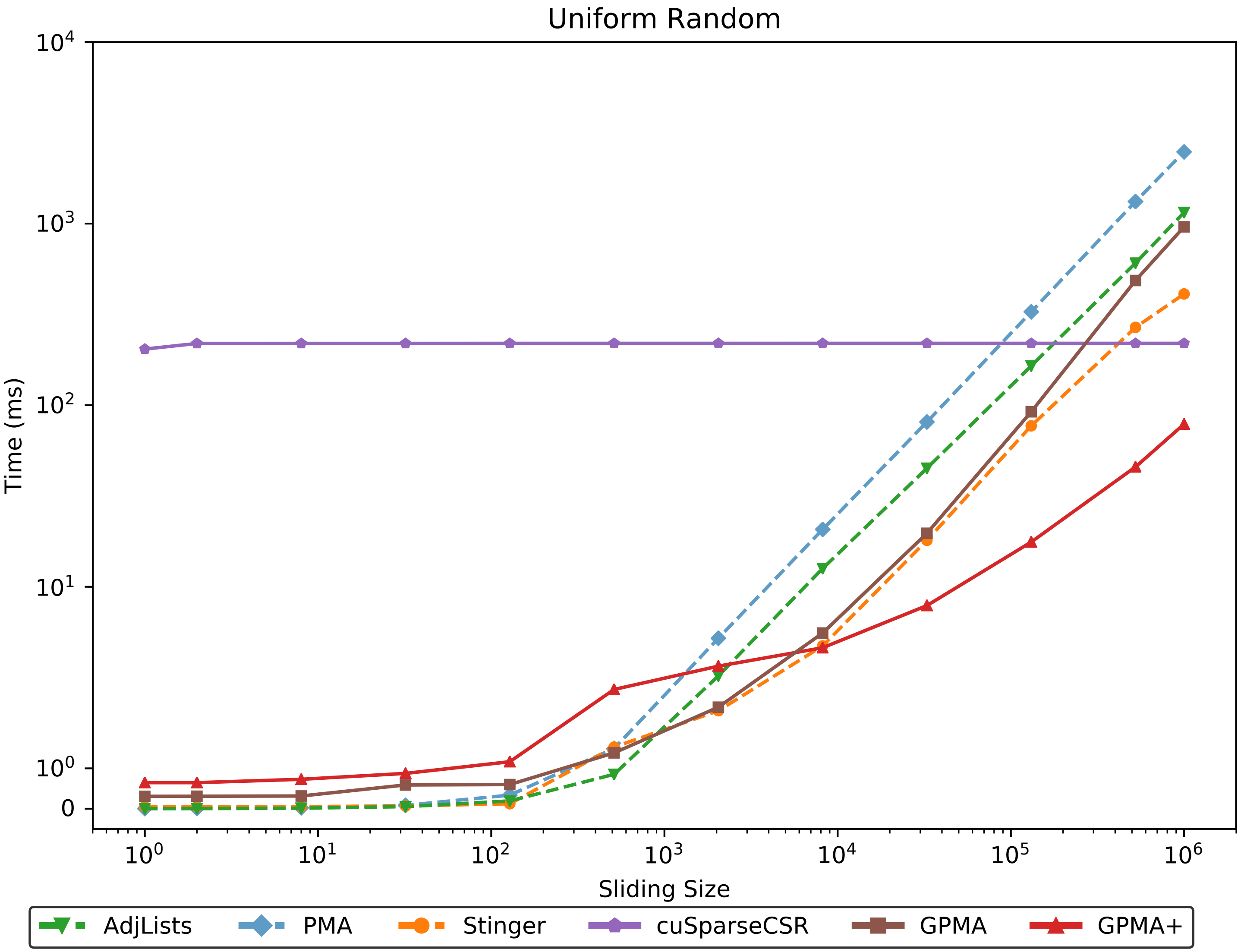
- CPU-based
  4-way Intel(R) Xeon(R) CPU E7-4820 v3
  40-cores, 1.90GHz
  128GB memory
- GPU-based
  Intel(R) Core i7-5820k
  6-cores, 3.30GHz
  64GB memory
  Nvidia GeForce Titan X 12GB

Baselines

	Container	BFS	Connect Component	PageRank
CPU Approaches	AdjLists	Standard Single Thread Algorithms		
	PMA			
	Stinger	Stinger built-in Parallel Algorithms		
GPU Approaches	cuSparseCSR	D. Merrill et al.	J. Soman et al.	CUSP SpMV
	GPMA+			

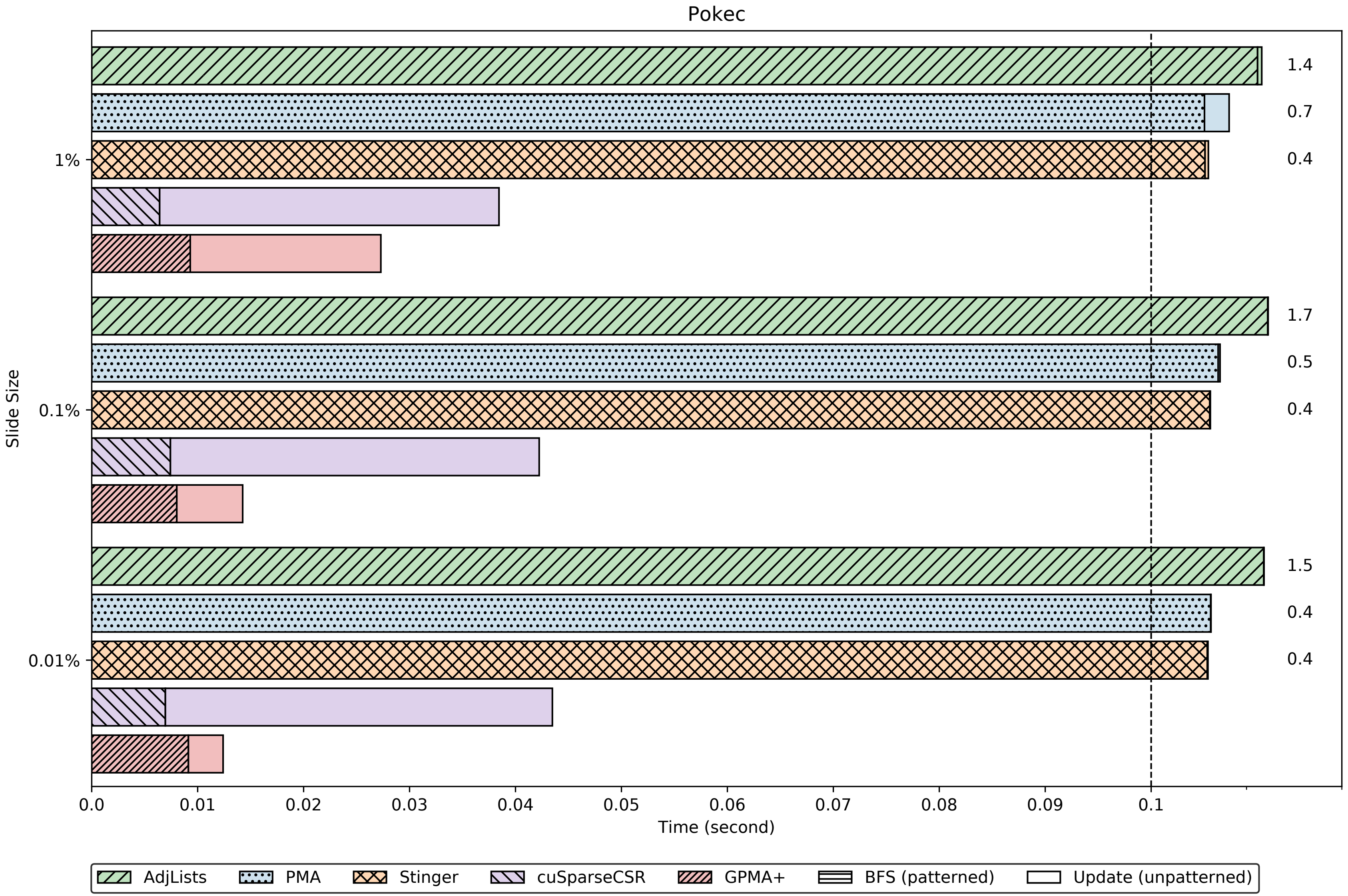
Evaluation

Insertion



Evaluation

BFS



Outline

- Introduction
- GPMA Dynamic Graph Processing
- GPMA+ Optimization
- Evaluation
- Conclusion

Conclusion

I. *We introduce a framework for GPU dynamic graph analytics and propose, the first of its kind, a GPU dynamic graph storage scheme to pave the way for real-time dynamic graph analytics on GPUs.*

II. *We devise two GPU-oriented parallel algorithms: GPMA and GPMA+, to support efficient updates against high-speed graph streams.*

III. *We conduct extensive experiments to show the performance superiority of GPMA and GPMA+.*

Thank you!

Open Source: https://github.com/desert0616/gpma_bfs_demo