Parallelizing Maximal Clique Enumeration on GPUs

Mohammad Almasri[*†], **Yen-Hsiang Chang**[*†], Izzat El Hajj[‡]
Rakesh Nagi[†], Jinjun Xiong[†§], and Wen-mei Hwu[†¶]

[*]Both authors contributed equally to this research
[†]University of Illinois at Urbana-Champaign, Urbana, IL, USA
[‡]American University of Beirut, Beirut, Lebanon
[§]University at Buffalo, Buffalo, NY, USA
[¶]Nvidia Corporation, Santa Clara, CA, USA

BACKGROUND

OUR SOLUTIONS

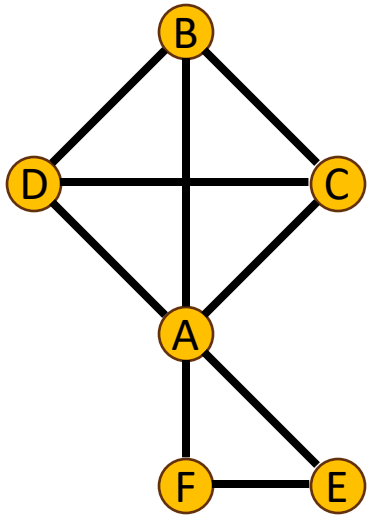
EVALUATION

Maximal Clique Enumeration (MCE)

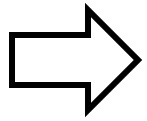
Clique: A complete subgraph

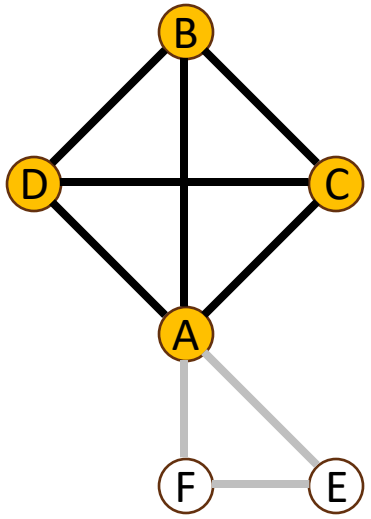
Maximal Clique: A clique that is not contained in a larger clique

Maximal Clique Enumeration: Find all maximal cliques in an undirected graph G

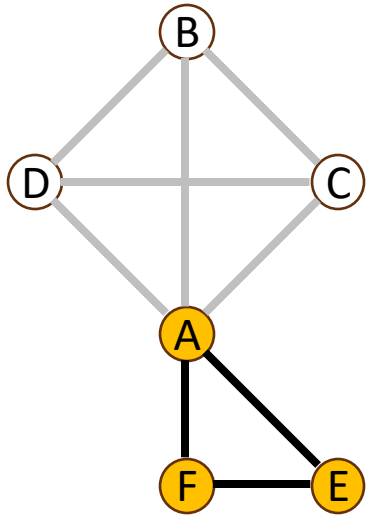


Input G





Maximal clique
$\{A, B, C, D\}$



Maximal clique
$\{A, E, F\}$

MCE Applications

Wide range of applications:

- Community Detection: *Community detection in complex networks via clique conductance* [1]
- Bioinformatics: *Predicting interactions in protein networks by completing defective cliques* [2]
- Electronic Design Automation: *Fast, Nearly Optimal ISE Identification With I/O Serialization Through Maximal Clique Enumeration* [3]
- Epidemiology: *Networks and the epidemiology of infectious disease* [4]
- Graph Compression: *Compact structure for sparse undirected graphs based on a clique graph partition* [5]
- …

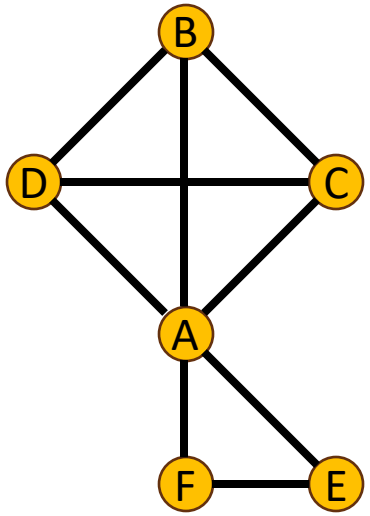
Bron-Kerbosch (BK) Algorithm [6]

A backtracking algorithm that traverses search trees

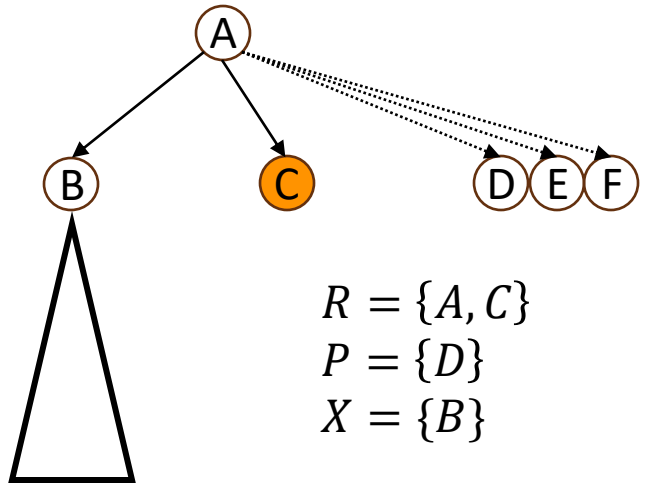
Maintain three disjoint vertex sets R, P and X:

- R (result): Current clique being explored
- P (possible): Common neighbors of R that have not been considered, which can expand the clique
- X (exclude): Common neighbors of R that have already been considered, for checking maximality

Input G



Search tree

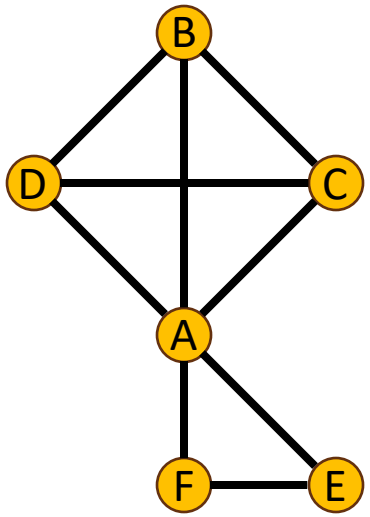


$R = \{A, C\}$
$P = \{D\}$
$X = \{B\}$

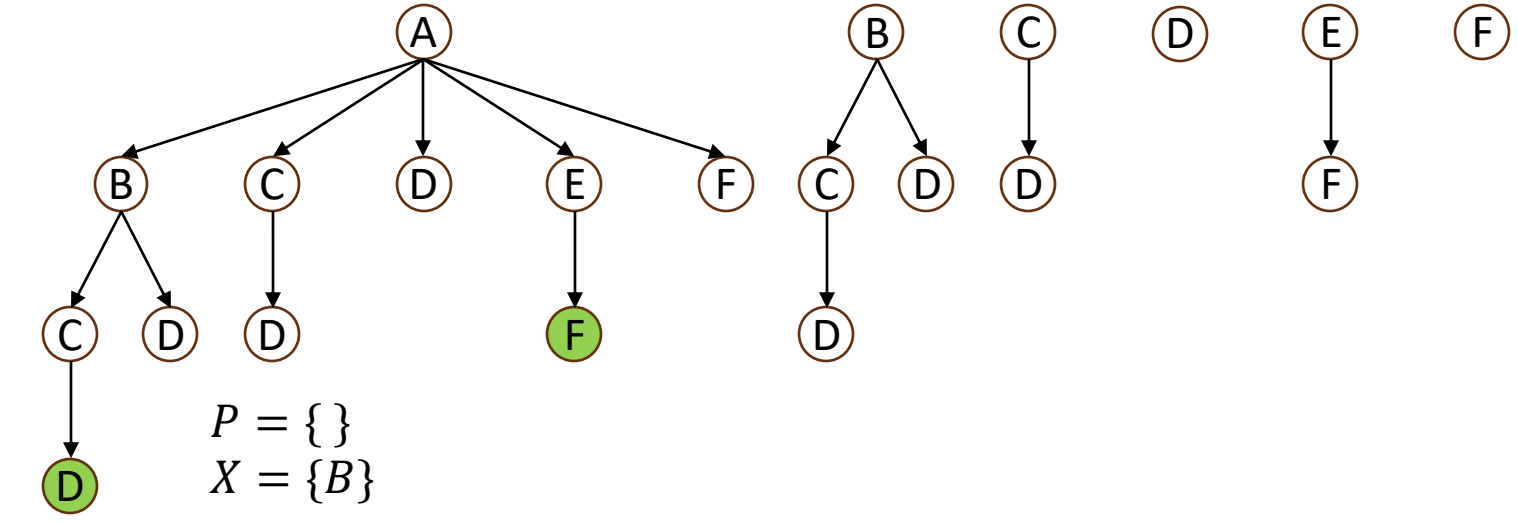
[6] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," Communications of the ACM, vol. 16, no. 9, pp. 575–577, 1973.

Bron-Kerbosch (BK) Algorithm

Input G



Search tree



Maximal clique

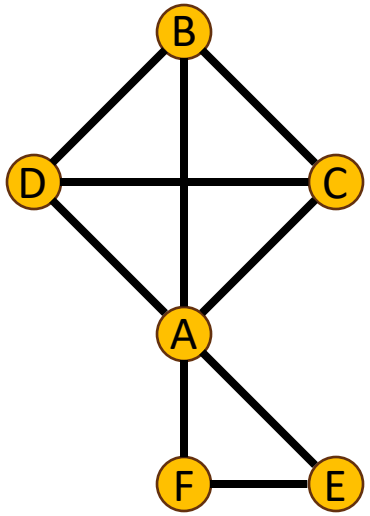
BK + pivoting [6]

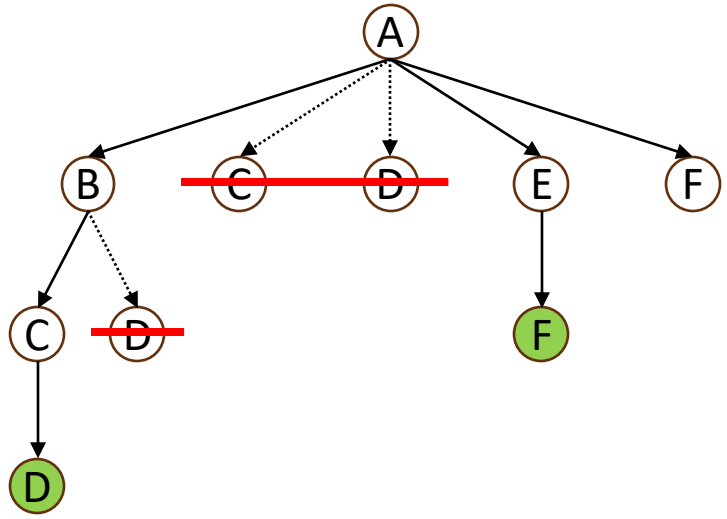
Issue in BK: Many branches are useless

If a pivot v_{pivot} is chosen from P, we only need to search from $P - N(v_{pivot})$

Typically select pivot having the largest number of neighbors also in P

Pivot can also be chosen from X [7]

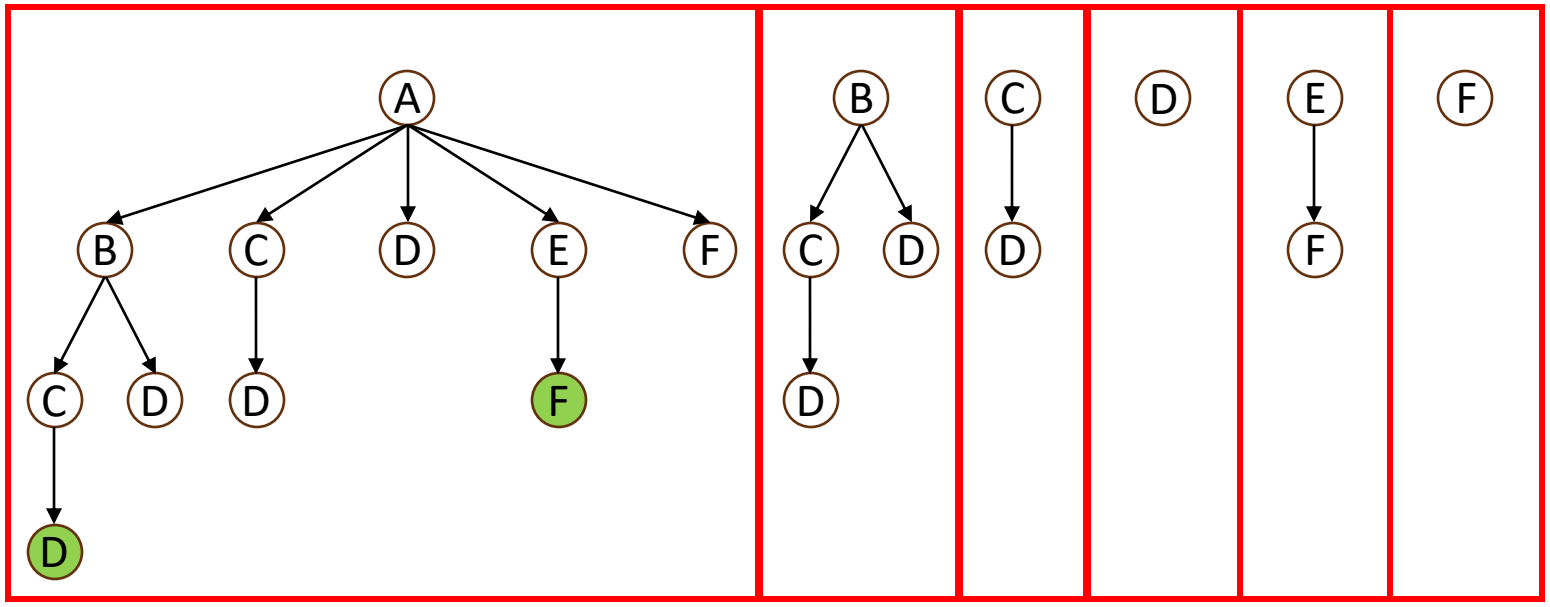




[6] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," Communications of the ACM, vol. 16, no. 9, pp. 575–577, 1973.
[7] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," Theoretical computer science, vol. 363, no. 1, pp. 28–42, 2006.

BK with other optimizations [8]

Independent first-level subtrees: Each search tree at the first level determine its P and X independently



[8] Eppstein, D., Löffler, M., & Strash, D. (2010). Listing all maximal cliques in sparse graphs in near-optimal time. Algorithms and Computation: 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I 21, 403–414.

BK with other optimizations [8]

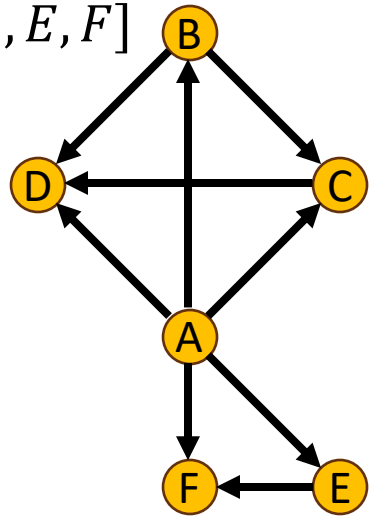
Degeneracy ordering:

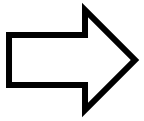
The search tree grows based on the size of P at each level

Reorder vertices by minimizing the maximum number of neighbors being ordered afterward.

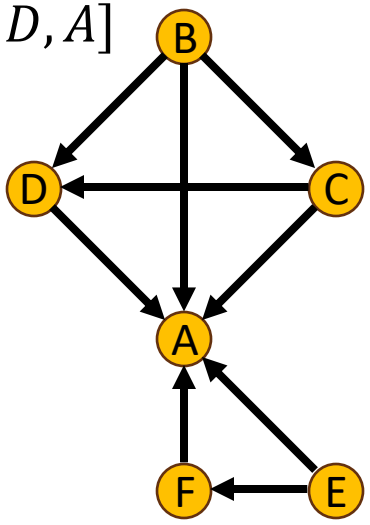
In real-world graphs, degeneracy d is way less than maximum degree Δ

$[A, B, C, D, E, F]$





$[E, F, B, C, D, A]$



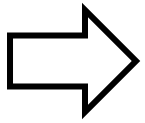
[8] Eppstein, D., Löffler, M., & Strash, D. (2010). Listing all maximal cliques in sparse graphs in near-optimal time. Algorithms and Computation: 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I 21, 403–414.

BK with other optimizations [8]

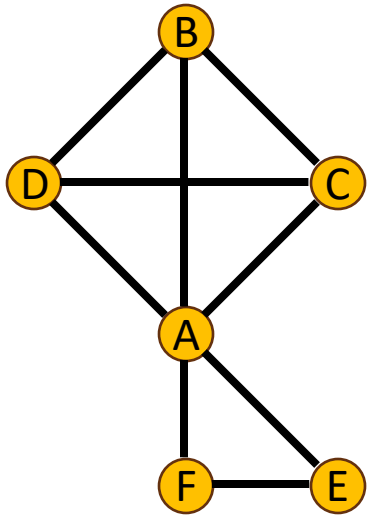
Induced subgraphs:

For each independent first level subtree, build a subgraph recording connections between $P \cup X$ and P

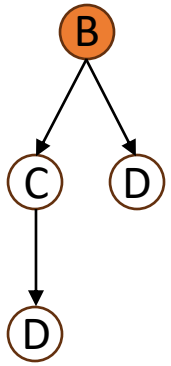
	A	B	C	D	E	F
A	0	1	1	1	1	1
B	1	0	1	1	0	0
C	1	1	0	1	0	0
D	1	1	1	0	0	0
E	1	0	0	0	0	1
F	1	0	0	0	1	0



	C	D
A	1	1
C	0	1
D	1	0



$P = \{C, D\}$
$X = \{A\}$



[8] Eppstein, D., Löffler, M., & Strash, D. (2010). Listing all maximal cliques in sparse graphs in near-optimal time. Algorithms and Computation: 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I 21, 403–414.

Recent CPU solution

“Manycore Clique Enumeration with Fast Set Intersections” [9]

Strength:

- SIMD-accelerated hash-join-based set intersections
- Use Threading Building Blocks (TBB) to exploit parallelism
- Recreate induced subgraphs at each recursive call to improve locality
- Open source and best performance reported

Weakness:

- Building hash adjacency lists of the input graph takes a long time, but it is only recorded in the read graph time and not reported in the results

[9] J. Blanuša, R. Stoica, P. Ienne, and K. Atasu, “Manycore clique enumeration with fast set intersections,” Proceedings of the VLDB Endowment, vol. 13, no. 12, pp. 2676–2690, 2020.

Recent GPU solution

“Accelerating the Bron-Kerbosch algorithm for maximal clique enumeration using GPUs” [10]

Strength:

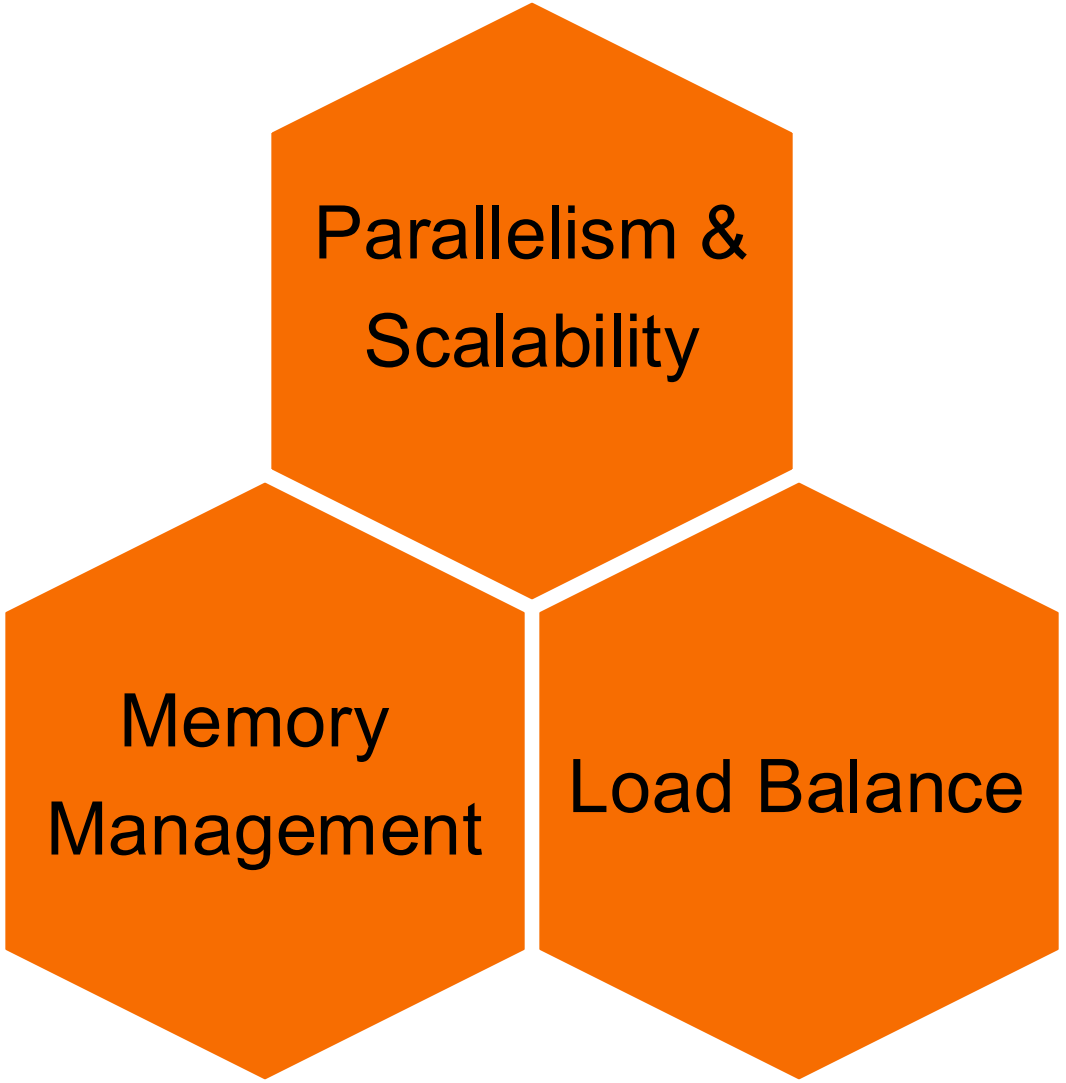
- Fine-grained parallelism by building primitives
- BFS approach favorable on GPU

Weakness:

- Not scalable to large graphs due to the nature of BFS and the space complexity
- High overheads due to moving data between CPU and GPU
- Not open source and only evaluated on small graphs on old GPU

[10] Y.-W. Wei, W.-M. Chen, and H.-H. Tsai, “Accelerating the Bron-Kerbosch algorithm for maximal clique enumeration using GPUs,” IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 9, pp. 2352–2366, 2021.

Main Challenges of MCE on GPUs



BACKGROUND

OUR SOLUTIONS

EVALUATION

Parallelism and Scalability

Adopt the backbone of previous paper of k-clique enumeration (KCE) on GPU [11]

- For each SM, launch max number of concurrent blocks
- Assign first-level subtrees to idle blocks and have each block perform DFS
- Threads within the block collaborate to build induced subgraphs (binary encoded), perform set operations and find pivots (divide into subwarp granularity if applicable)

[11] M. Almasri, I. E. Hajj, R. Nagi, J. Xiong, and W. Hwu, “Parallel k-clique counting on gpus,” in Proceedings of the 36th ACM International Conference on Supercomputing, 2022, pp. 1–14.

Difference between KCE and MCE

	KCE	MCE
Search level	$O(d)$ with early termination	$O(d)$
Induced subgraphs	Edges between P and P	Edge between $P \cup X$ and P
Storing R, P and X	R and P only	R, P and X

Memory management

- CB: # concurrent blocks on GPU
- w: word size
- d: degeneracy
- Δ: maximum degree
- Recall that $d \ll \Delta$ in real-world graphs

	KCE	MCE
Induced subgraphs	$O\left(\frac{CB \times d^2}{w}\right)$	$O\left(\frac{CB \times d \times \Delta}{w}\right)$
R, P and X	$O(CB \times d^2)$	$O(CB \times d \times \Delta)$

Induced subgraphs

IPX: induced subgraphs between $P \cup X$ and P

- More memory needed $O\left(\frac{CB \times d \times \Delta}{w}\right)$
- Set operations using bit-wise operations

IP: induced subgraphs only between P and P

- Less memory needed $O\left(\frac{CB \times d^2}{w}\right)$
- Updating X needs to access original graph

IPX

	C	D
A	1	1
C	0	1
D	1	0

IP

	C	D
C	0	1
D	1	0

Splitting X

Key idea: Only those vertices in P at the first level can be added into X

X_P: those from the first level of P

- vertices can be added and be removed
- but $|X_P| = O(d)$
- If binary encoded and stored for each level, we only need $O\left(\frac{CB \times d^2}{w}\right)$ memory

X_X: those from the first level of X

- Only shrinks when advancing to a deeper level
- Maintain a single array and use level pointers to indicate boundaries between levels
- Only need $O(CB \times \Delta)$ memory in total

Memory management -- Summary

- CB: number of concurrent blocks on GPU
- d: degeneracy
- Δ: maximum degree
- w: word size
- Recall that $d \ll \Delta$ in real-world graphs

	MCE	MCE-IPX	MCE-IP
Induced subgraphs	$O\left(\frac{CB \times d \times \Delta}{w}\right)$	$O\left(\frac{CB \times d \times \Delta}{w}\right)$	$O\left(\frac{CB \times d^2}{w}\right)$
R, P and X	$O(CB \times d \times \Delta)$	$O\left(CB \times \left(\frac{d^2}{w} + \Delta\right)\right)$	

Load balance

In k-clique [11], load imbalance is addressed by extracting subtrees from the second level

We try the same idea for MCE, but it is not enough

- Search trees only start to branch at very deep levels

Dynamic load balancing using work list is not suitable

- Number of subtasks grows exponentially
- Data needed to represent a subtask is huge

[11] M. Almasri, I. E. Hajj, R. Nagi, J. Xiong, and W. Hwu, “Parallel k-clique counting on gpus,” in Proceedings of the 36th ACM International Conference on Supercomputing, 2022, pp. 1–14.

Worker list instead of work list

We design a worker list instead

- When a block is idle, put its ID into the list
- Whenever a block is branching, get some idle blocks from the list and offload branches

Benefits:

- Number of active tasks is bounded by the number of blocks
- No need to tune the capacity of the list, but only tune the offload policy

Policy

- The branch to be offloaded should not be small
  - Require that $|P| \geq 10$ for the root node of the branch
  - Not sensitive to the constant chosen
- The block that is offloading tasks should have a substantial amount of other work to do after offloading the branch
  - Offload a branch if there are other branches at the same level and other branches in previous levels that have not yet been explored

BACKGROUND

OUR SOLUTIONS

EVALUATION

Performance Evaluation

Compare against *“Manycore Clique Enumeration with Fast Set Intersections”* [9]

- A dual-socket Intel Xeon Skylake platform with 48c/96t and 360 GB of main memory

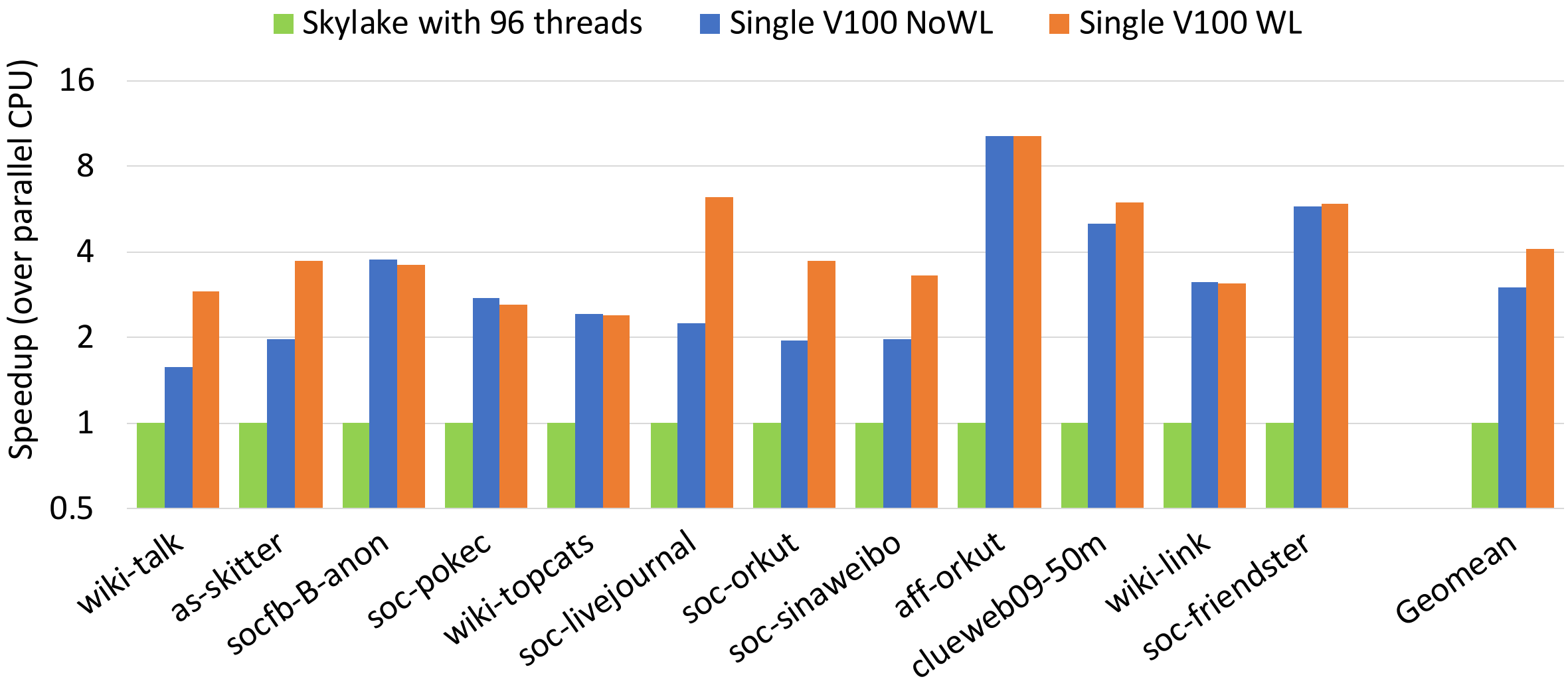
We evaluate our GPU kernels on V100 GPU with 32 GB DRAM

Both include pre-processing time but not read graph time

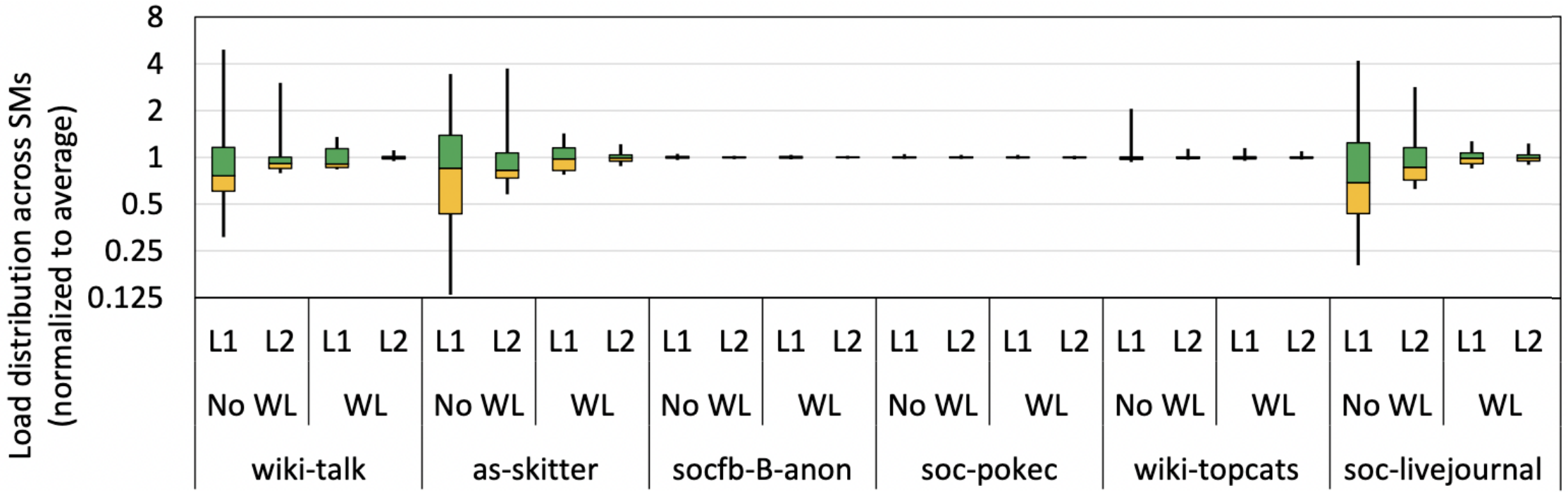
[9] J. Blanuša, R. Stoica, P. Ienne, and K. Atasu, “Manycore clique enumeration with fast set intersections,” Proceedings of the VLDB Endowment, vol. 13, no. 12, pp. 2676–2690, 2020.

Graph	$\|V\|$	$\|E\|$	Δ	d	# of maximal cliques
wiki-talk	2,394,385	4,659,565	100,029	131	86,333,306
as-skitter	1,696,415	11,095,298	35,455	111	37,322,355
socfb-B-anon	2,937,613	20,959,854	4,356	63	27,593,398
soc-pokec	1,632,804	22,301,964	14,854	47	19,376,873
wiki-topcats	1,791,489	25,444,207	238,342	99	27,229,873
soc-livejournal	4,033,138	27,933,062	2,651	213	38,413,665
soc-orkut	3,072,442	117,185,083	33,313	253	2,269,631,973
soc-sinaweibo	58,655,850	261,321,033	278,489	193	1,117,416,174
aff-orkut	8,730,858	327,036,486	318,268	471	417,032,363
clueweb09-50m	428,136,613	446,766,953	308,477	192	1,001,323,679
wiki-link	27,154,799	543,183,611	4,271,341	1,120	568,730,123
soc-friendster	65,608,367	1,806,067,135	5,214	304	3,364,773,700

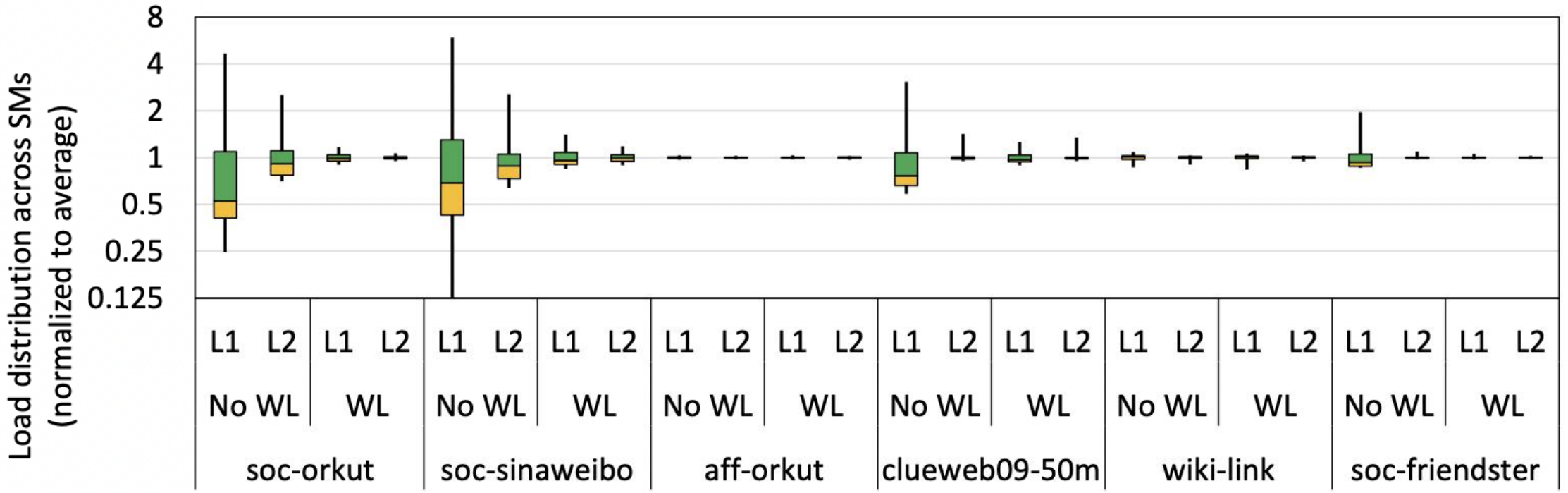
Evaluation – Speedup



Evaluation – Load balance



Evaluation – Load balance

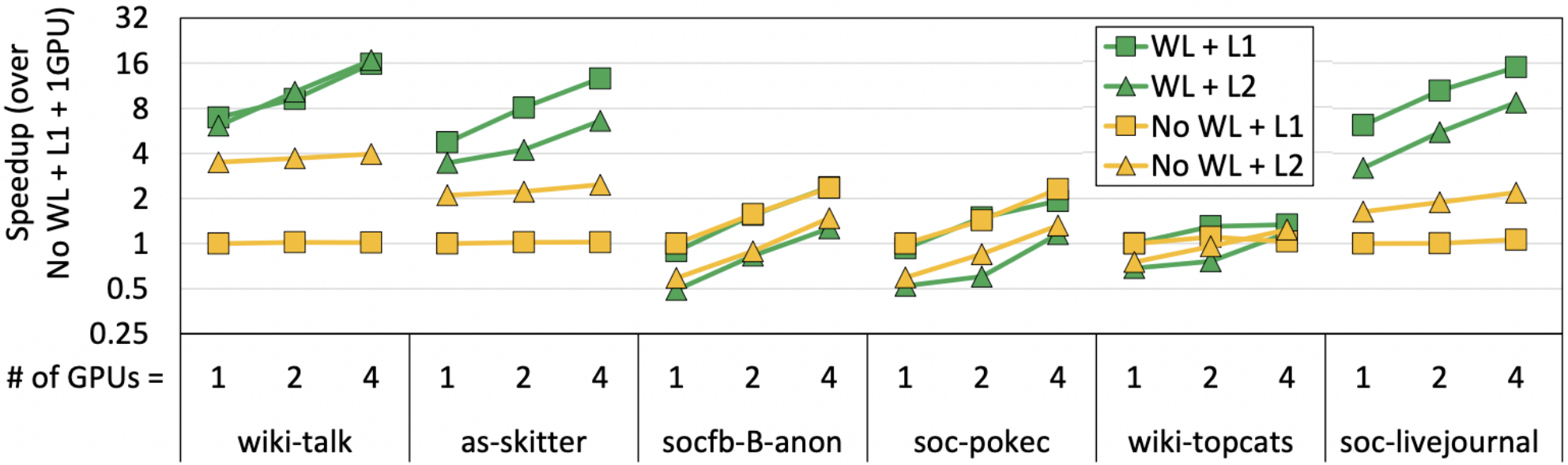


Scaling to multi-GPUs

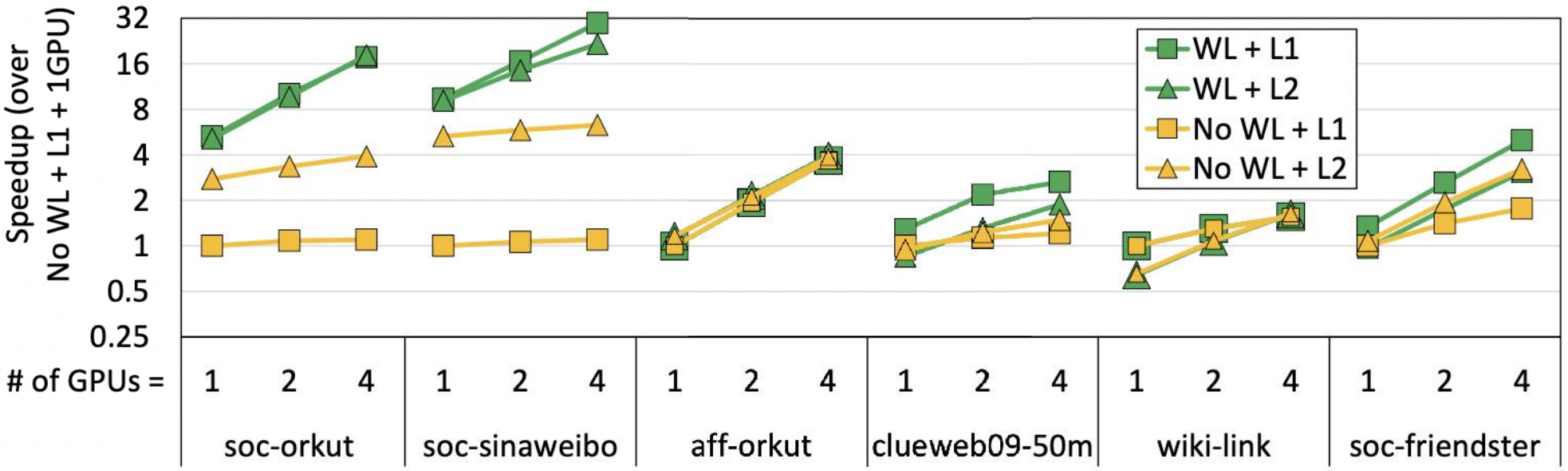
Static division of independent first- or second-level subtrees to each GPU

- Round-robin fashion to avoid vertices sharing similar properties
- No communication between GPUs when running kernels
- Do not include pre-processing time, since we only scale MCE kernels to multi-GPUs

Evaluation – Scalability



Evaluation – Scalability



Evaluation – Choice of optimizations

COMPARING HEURISTIC (UNDERLINED) AND OPTIMAL (BOLD) SELECTION OF OPTIMIZATION COMBINATIONS

		Execution time (s)				Heuristic
Graph	Δ / d	L1 + IP	L1 + IPX	L2 + IP	L2 + IPX	slowdown
wiki-talk	763.58	1.14	1.74	**1.10**	1.48	1.03
as-skitter	319.41	**0.58**	0.70	0.80	0.66	1.00
socfb-B-anon	69.14	0.25	**0.22**	0.42	0.36	1.00
soc-pokec	316.04	0.21	**0.21**	0.30	0.31	1.00
wiki-topcats	2,407.49	**0.60**	1.85	0.74	1.99	1.00
soc-livejournal	12.45	0.52	**0.44**	1.10	0.65	1.00
soc-orkut	131.67	22.10	18.68	29.29	**16.98**	1.10
soc-sinaweibo	1,442.95	12.20	13.05	15.80	**11.39**	1.07
aff-orkut	675.73	**8.16**	11.82	9.31	12.18	1.00
clueweb09-50m	1,606.65	**6.70**	9.77	15.10	11.35	1.00
wiki-link	3,813.70	**24.67**	-	29.76	-	1.00
soc-friendster	17.15	44.64	**33.41**	58.16	39.17	1.00
					Geomean	1.02

Summary

- Highly efficient GPU solution to MCE
  - Evaluate trade-offs between different paradigms of induced subgraphs
  - Maintain only one copy of excluding set X across levels to reduce memory footprint
  - Design a worker list to deal with load imbalance
- Single GPU solution outperforms the state-of-the-art parallel CPU implementation by a geometric mean of $4.1\times$ (up to $10.2\times$) on V100 and $4.9\times$ (up to $16.7\times$) on A100
- Scalable to multiple GPUs
- Open-sourced to enable further research on accelerating MCE
  - Available, functional and reproduced

Future directions

- Solve load imbalance for building induced subgraphs
- Investigate distributed setup for solving larger graphs
- Extend the functionality of worker list to other problems suffering from load imbalance

Acknowledgments

- We thank Zaid Qureshi and Amir Nassereldine for their insights and technical assistance.
- This work is supported in part by the IBM-Illinois Center for Cognitive Computing Systems Research (C3SR).
- Izzat El Hajj acknowledges the support of the University Research Board of the American University of Beirut (AUB-URB-104391-26749).
- Jinjun Xiong acknowledges the support of NSF (FuSe-TG 2235364) and the joint support of NSF and IES through AI4ExceptionalEd (2229873).
- We are also grateful to NVIDIA's Applied Research Accelerator Program for donating A100 GPUs that were helpful in the final testing stages of this work.

References

[1] Z. Lu, J. Wahlström, and A. Nehorai, “Community detection in complex networks via clique conductance,” Scientific reports, vol. 8, no. 1, p. 5982, 2018.

[2] H. Yu, A. Paccanaro, V. Trifonov, and M. Gerstein, “Predicting interactions in protein networks by completing defective cliques,” Bioinformatics, vol. 22, no. 7, pp. 823–829, 2006.

[3] A. K. Verma, P. Brisk, and P. Ienne, “Fast, nearly optimal ISE identification with I/O serialization through maximal clique enumeration,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 3, pp. 341–354, 2010.

[4] L. Danon, A. Ford, T. House, C. Jewell, M. Keeling, G. Roberts, J. Ross, and M. Vernon, “Networks and the epidemiology of infectious disease,” Interdisciplinary Perspectives on Infectious Diseases, vol. 2011, p. 284909, 2011.

[5] F. Glaria, C. Hernández, S. Ladra, G. Navarro, and L. Salinas, “Compact structure for sparse undirected graphs based on a clique graph partition,” *Information Sciences*, vol. 544, pp. 485–499, 2021.

[6] C. Bron and J. Kerbosch, “Algorithm 457: finding all cliques of an undirected graph,” Communications of the ACM, vol. 16, no. 9, pp. 575–577, 1973.

[7] E. Tomita, A. Tanaka, and H. Takahashi, “The worst-case time complexity for generating all maximal cliques and computational experiments,” Theoretical computer science, vol. 363, no. 1, pp. 28–42, 2006.

[8] Eppstein, D., Löffler, M., & Strash, D. (2010). Listing all maximal cliques in sparse graphs in near-optimal time. Algorithms and Computation: 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I 21, 403–414.

[9] J. Blanuša, R. Stoica, P. Ienne, and K. Atasu, “Manycore clique enumeration with fast set intersections,” Proceedings of the VLDB Endowment, vol. 13, no. 12, pp. 2676–2690, 2020.

[10] Y.-W. Wei, W.-M. Chen, and H.-H. Tsai, “Accelerating the Bron-Kerbosch algorithm for maximal clique enumeration using GPUs,” IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 9, pp. 2352–2366, 2021.

[11] M. Almasri, I. E. Hajj, R. Nagi, J. Xiong, and W. Hwu, “Parallel k-clique counting on gpus,” in Proceedings of the 36th ACM International Conference on Supercomputing, 2022, pp. 1–14.

THANK YOU!

QUESTIONS?

Parallelizing Maximal Clique Enumeration on GPUs

Mohammad Almasri[*†], **Yen-Hsiang Chang**[*†], Izzat El Hajj[‡]
Rakesh Nagi[†], Jinjun Xiong[†§], and Wen-mei Hwu[†¶]