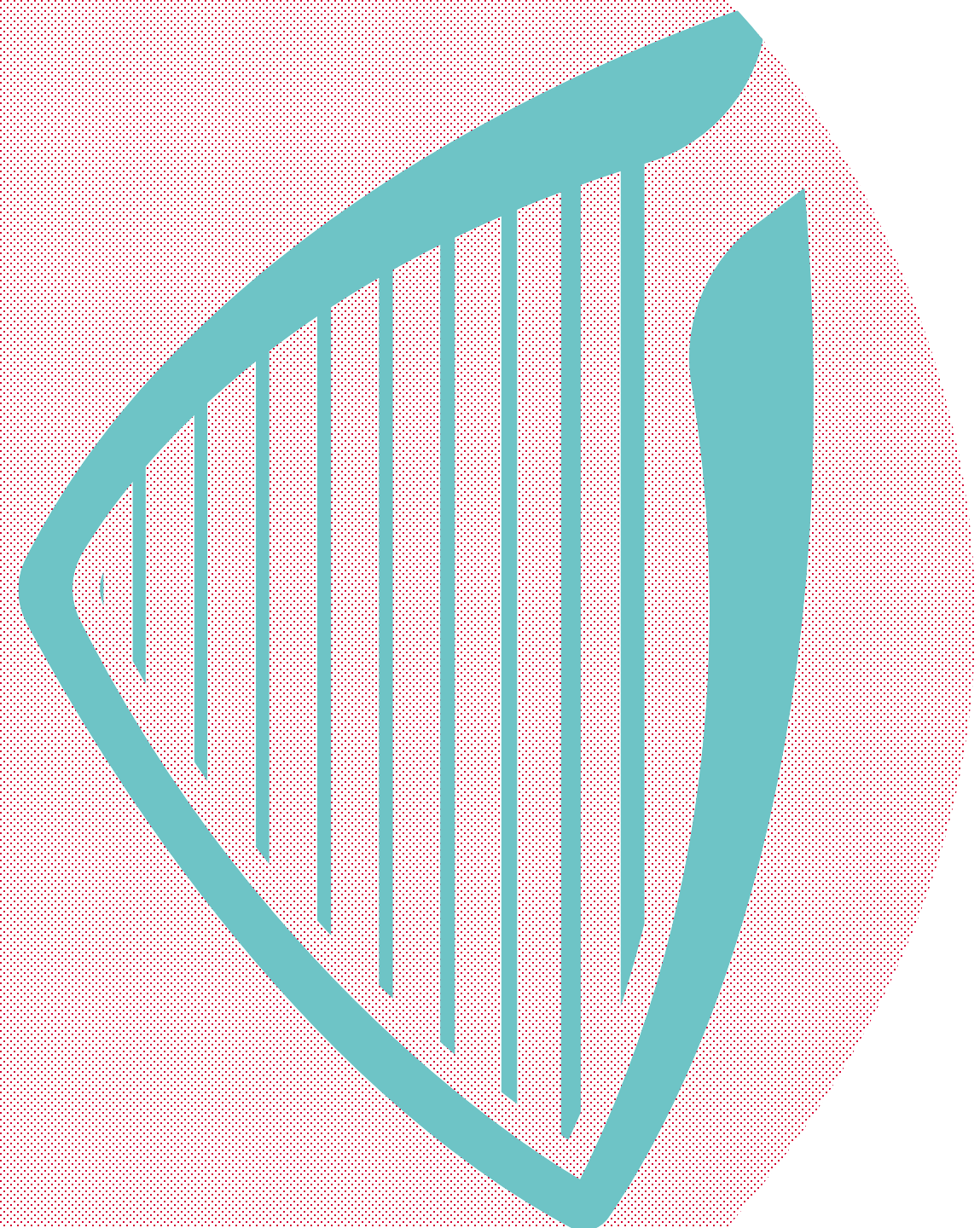


Declarative Analytics on Heterogeneous HPC Systems

Ahmedur Rahman Shovon

Committee:
Dr. Sidharth Kumar, *Chair and Advisor*
Dr. Michael E. Papka
Dr. Zhiling Lan
Dr. Stavros Sintos
Dr. Gopikrishna Deshpande (AU)
Dr. Thomas Gilray (WSU)

JULY 15, 2025

Roadmap

1. **Introduction and motivation**
2. GPU based Datalog engine development
3. Multi-node multi-GPU Datalog engine development
4. GPU portability of Datalog engines
5. Power analysis of Datalog engines
6. Exploration of high-dimensional scientific analytics
7. Future work
8. Conclusion

UIC

Declarative Programming Paradigm

Users expresses what to achieve with the data rather than how to accomplish it

Employees

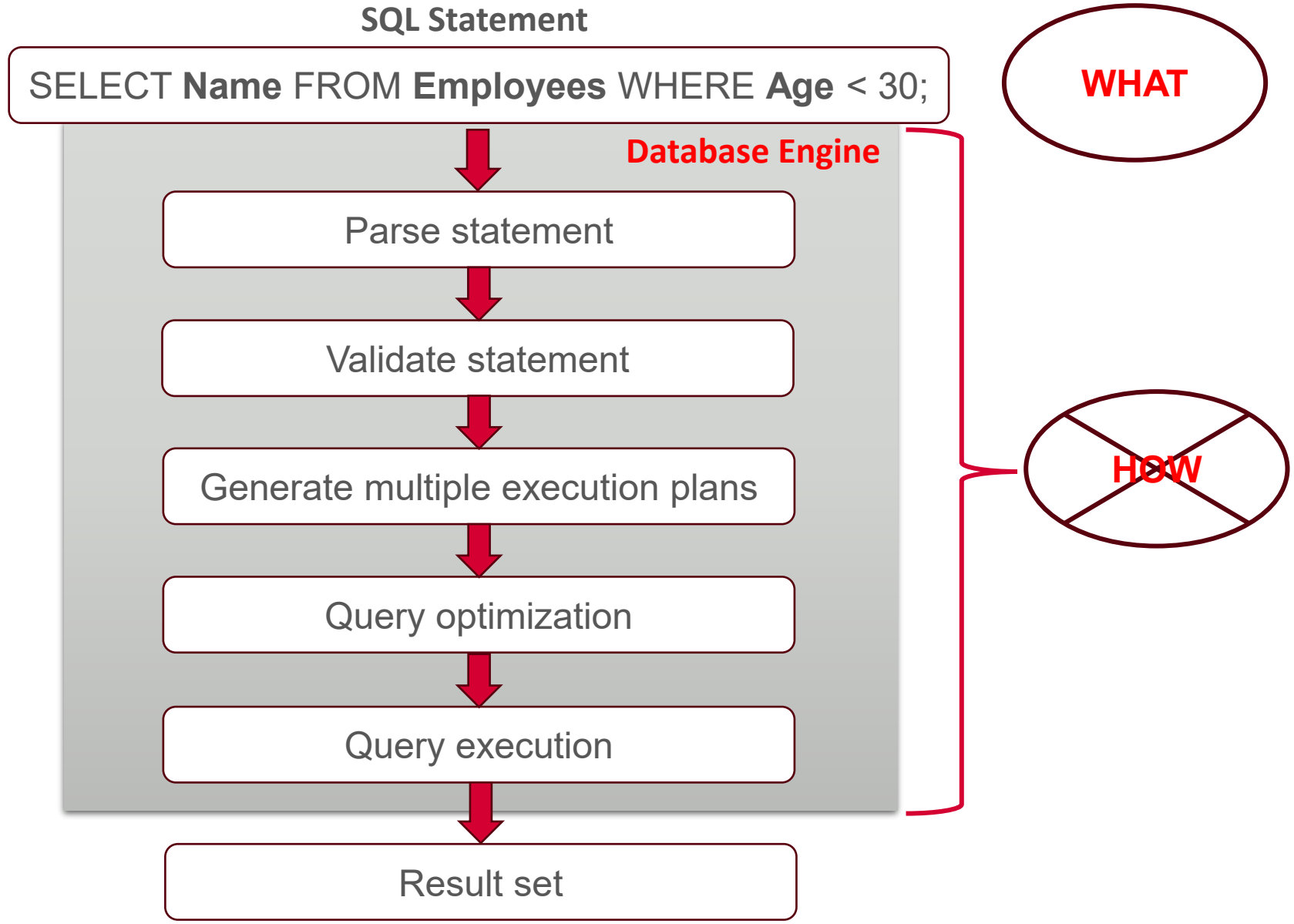
EmpID	Name	Email	Age	ManagerID
0	Alice	alice@example.com	56	Null
1	Bob	bob@example.com	33	0
2	Eve	eve@example.com	26	0
3	Carol	carol@example.com	34	2
4	Dave	dave@example.com	22	3

Find all employees who are less than 30 years old

SELECT **Name** FROM **Employees** WHERE **Age** < 30;

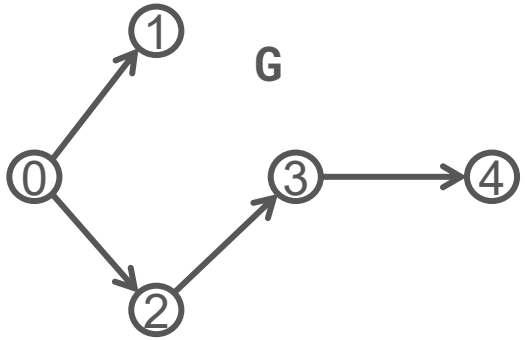
• Lloyd, J. W. (1994, September). Practical Advantages of Declarative Programming. In GULP-PRODE (1) (pp. 18-30).

How is the query processed?



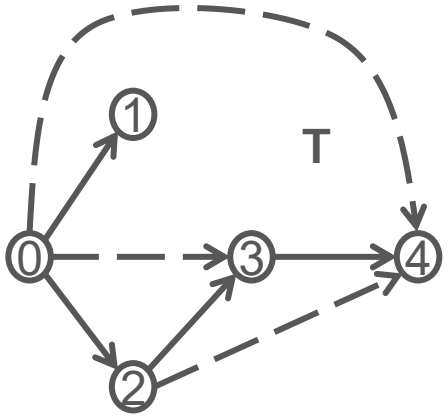
- Processing a SQL statement. Microsoft. https://learn.microsoft.com/en-us/sql/odbc/reference/processing-a-sql-statement?view=sql-server-2017

Recursive Queries using SQL



Employees

EmpID	Name	Email	Age	ManagerID
0	Alice	alice@example.com	56	Null
1	Bob	bob@example.com	33	0
2	Eve	eve@example.com	26	0
3	Carol	carol@example.com	34	2
4	Dave	dave@example.com	22	3

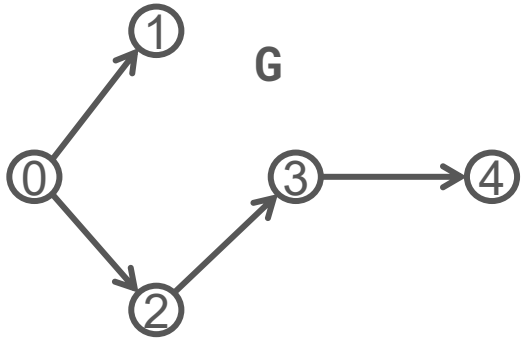


Find all managers, including direct and indirect managers.

```
-- Recursive CTE to find all managers (direct and indirect)
WITH RECURSIVE ManagerHierarchy AS (
-- Base case: direct manager relationship
        SELECT EmpID AS EmployeeID, ManagerID
        FROM Employees WHERE ManagerID IS NOT NULL
        UNION
-- Recursive case: find manager of managers recursively
        SELECT mh.EmployeeID, e.ManagerID
        FROM ManagerHierarchy mh
        JOIN Employees e ON mh.ManagerID = e.EmpID WHERE e.ManagerID IS NOT NULL
)
SELECT DISTINCT ManagerID, Employeeid FROM ManagerHierarchy ORDER BY ManagerID;
```

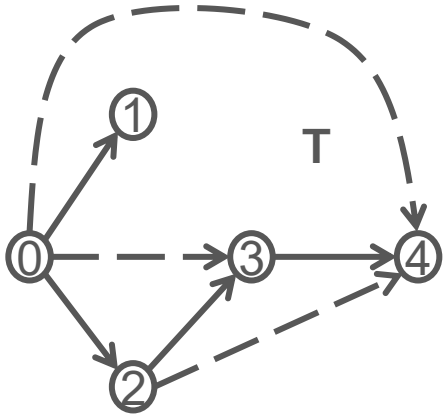
Advanced approach: Datalog

Recursive Queries using Datalog



Employees

EmpID	Name	Email	Age	ManagerID
0	Alice	alice@example.com	56	Null
1	Bob	bob@example.com	33	0
2	Eve	eve@example.com	26	0
3	Carol	carol@example.com	34	2
4	Dave	dave@example.com	22	3



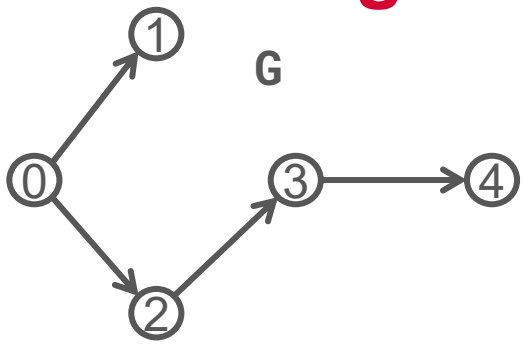
Find all managers, including direct and indirect managers.

Finding recursive relationships using Datalog

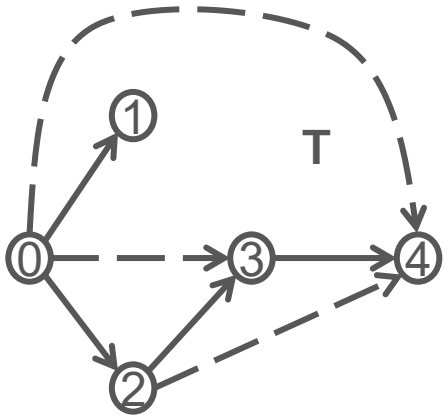
```
// Base case: direct manager manages employee
Manager(mID, eID) :- Employees(eID, mID), mID != NULL.

// Recursive case: indirect managers
Manager(mID, eID) :- Manager(mID, nID), Employees(eID, nID).
```

Advantages of Recursive Queries with Datalog



Find all managers, including direct and indirect managers.



Finding recursive relationships using SQL

```
-- Recursive CTE to find all managers (direct and indirect)
WITH RECURSIVE ManagerHierarchy AS (
-- Base case: d
        SEL
        FRO
        WHERE ManagerID IS NOT NULL
        UN
-- Recursive ca
        SEL
        FRO
        JOI
        WH
)
SELECT DISTINCT ManagerID, EmployeeID FROM ManagerHierarchy
ORDER BY ManagerID;
```

Finding recursive relationships using Datalog

Manager(MID) :- Employees(EID, MID), MID != NULL.

ID).

Datalog offers greater expressiveness and natural syntax for recursive queries

As a logic programming language, it's well-suited for deductive databases

Ideal for declarative analytics to find complex patterns over large datasets

- Shovon, A. R. (2024, April 23). Recursive queries in SQL and Datalog. Ahmedur Rahman Shovon. Retrieved June 22, 2024, from https://arshovon.com/blog/recursive-queries/

Datalog Rules Execution

Datalog Rules

```
// Base case: direct manager manages employee
Manager(mID, eID) :- Employees(eID, mID), mID != NULL.
// Recursive case: indirect managers
Manager(mID, eID) :- Manager(mID, nID), Employees(eID, nID).
```

Datalog Engine

Parse rules
Generate query plan

Iterative Relational Algebra

$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$

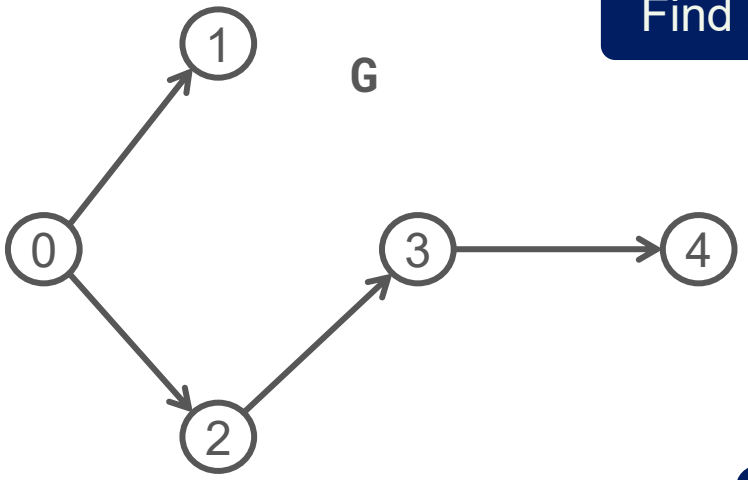
Union | Projection | Join

Relational algebra operations

Loop until $\textbf{manager}_i = \textbf{manager}_{i-1}$

- Gilray, T., & Kumar, S. (2019, December). Distributed relational algebra at scale. In *2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC)* (pp. 12-22). IEEE.

Manager Chains Problem as Transitive Closure



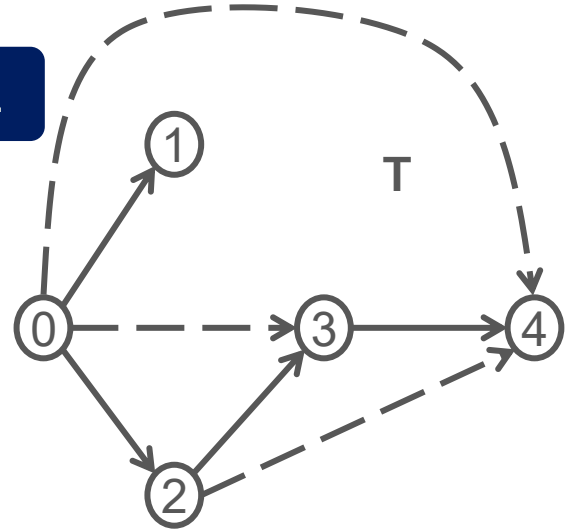
0	1
0	2
2	3
3	4

Find all managers, including direct and indirect managers.

```
// Base case: direct manager manages employee
Manager(mID, eID) :- Employees(eID, mID), mID != NULL.
// Recursive case: indirect managers
Manager(mID, eID) :- Manager(mID, nID), Employees(eID, nID).
```

Transitive closure computation of a graph

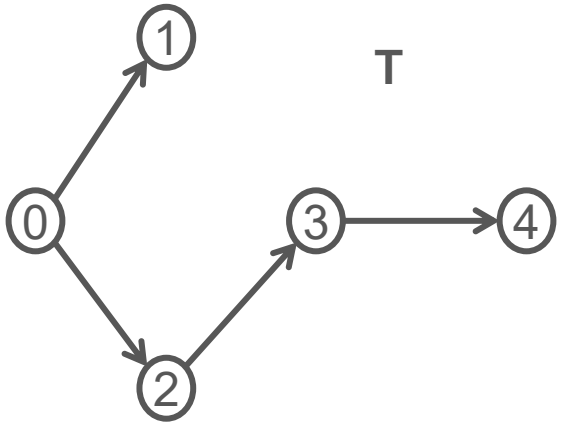
```
tc(X, Y) :- edges(X, Y).
tc(X, Z) :- tc(X, Y), edges(Y, Z).
```



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

Transitive Closure: Iterations 1

$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$



0	1
0	2
2	3
3	4

Transitive Closure: Iterations 1

$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$

$\rho_{0/1}(T)$

1	0
2	0
3	2
4	3

$\bowtie$

G

0	1
0	2
2	3
3	4

Transitive Closure: Iterations 1

$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$

$\rho_{0/1}(T)$

1	0
2	0
3	2
4	3

$\bowtie$

G

0	1
0	2
2	3
3	4

$\rho_{0/1}(T) \bowtie G$

2	0	3

Transitive Closure: Iterations 1

$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$

$\rho_{0/1}(T)$

1	0
2	0
3	2
4	3

$\bowtie$

G

0	1
0	2
2	3
3	4

$\rho_{0/1}(T) \bowtie G$

1	0	3
3	2	4

Transitive Closure: Iterations 1

$$F_G(T) \triangleq G \cup \boxed{\Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)}$$

$\rho_{0/1}(T)$

1	0
2	0
3	2
4	3

$\bowtie$

G

0	1
0	2
2	3
3	4

$\rho_{0/1}(T) \bowtie G$

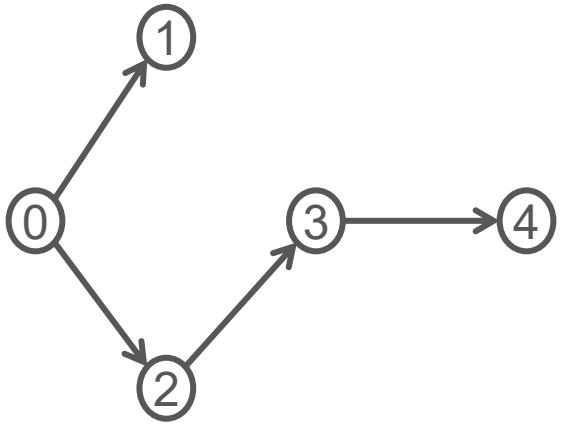
1	0	3
3	2	4

$\Pi_{1,2}(\rho_{0/1}(T) \bowtie G)$

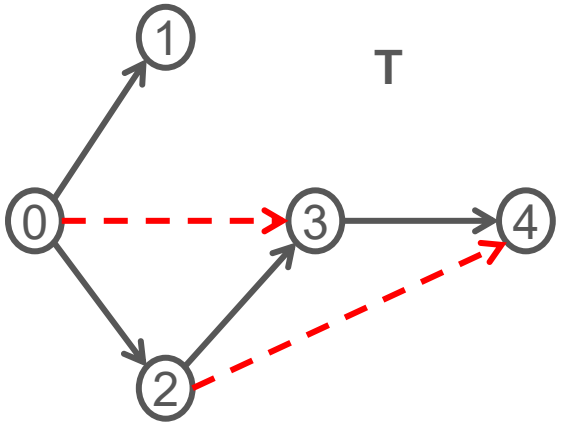
0	3
2	4

Transitive Closure: Iterations 1

$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$



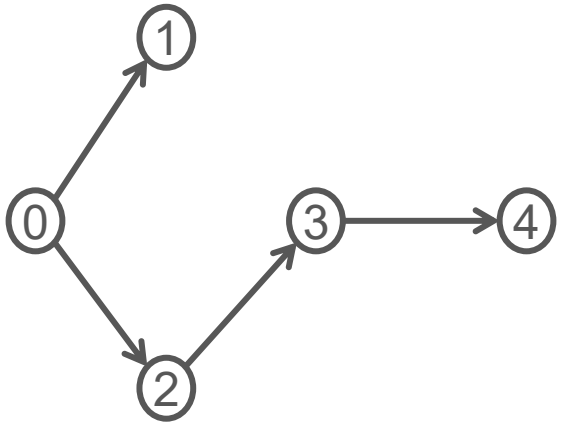
0	1
0	2
2	3
3	4



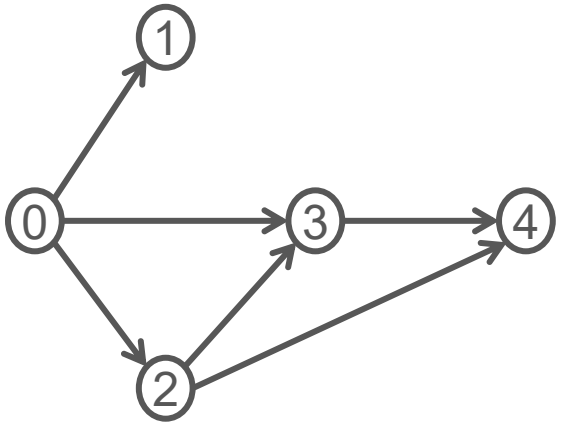
0	1
0	2
2	3
3	4
0	3
2	4

Transitive Closure: Iterations 2

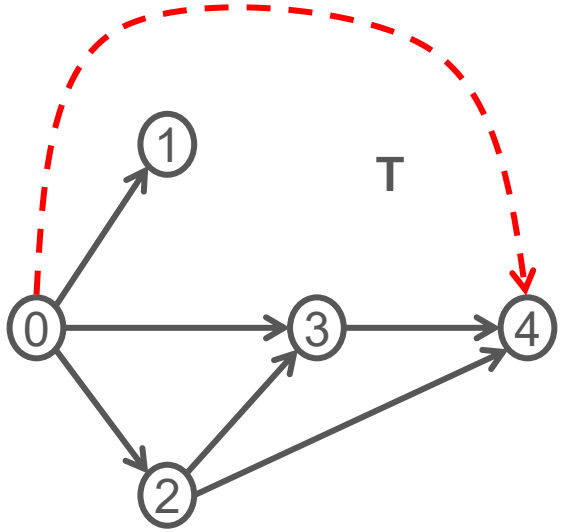
$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$



0	1
0	2
2	3
3	4



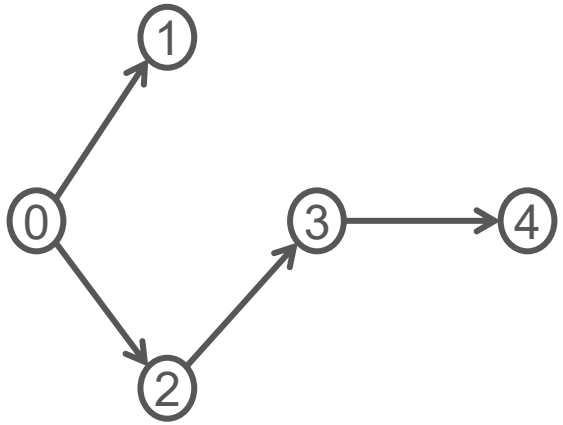
0	1
0	2
2	3
3	4
0	3
2	4



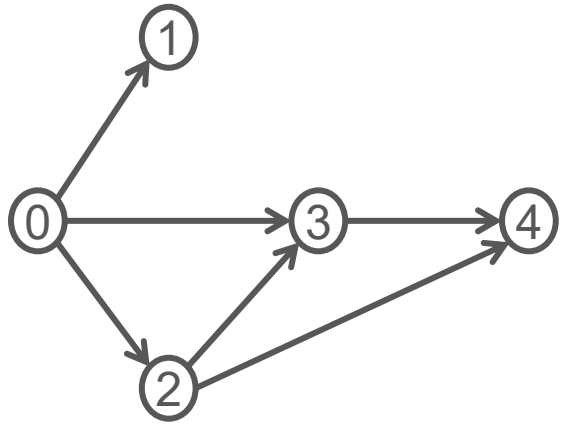
0	1
0	2
2	3
3	4
0	3
2	4
0	4

Transitive Closure: Iterations 3

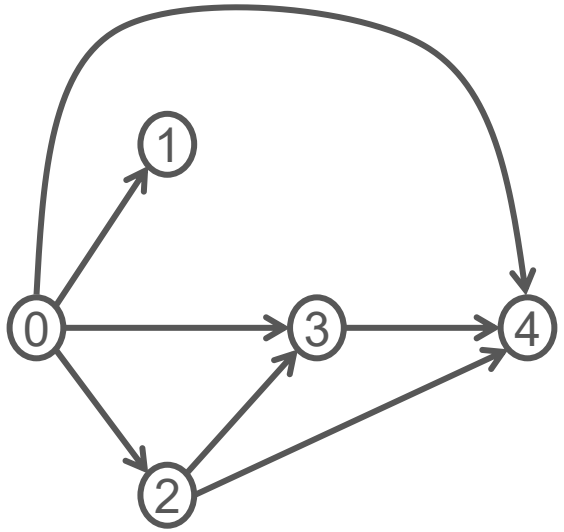
Fixed-point reached



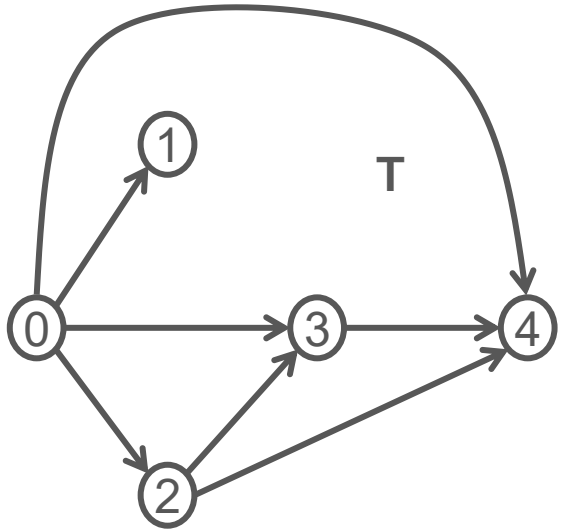
0	1
0	2
2	3
3	4



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



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

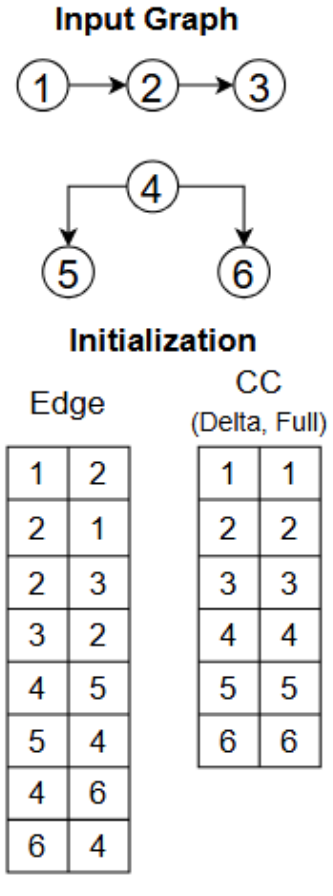


0	1
0	2
2	3
3	4
0	3
2	4
0	4

Graph Mining Applications

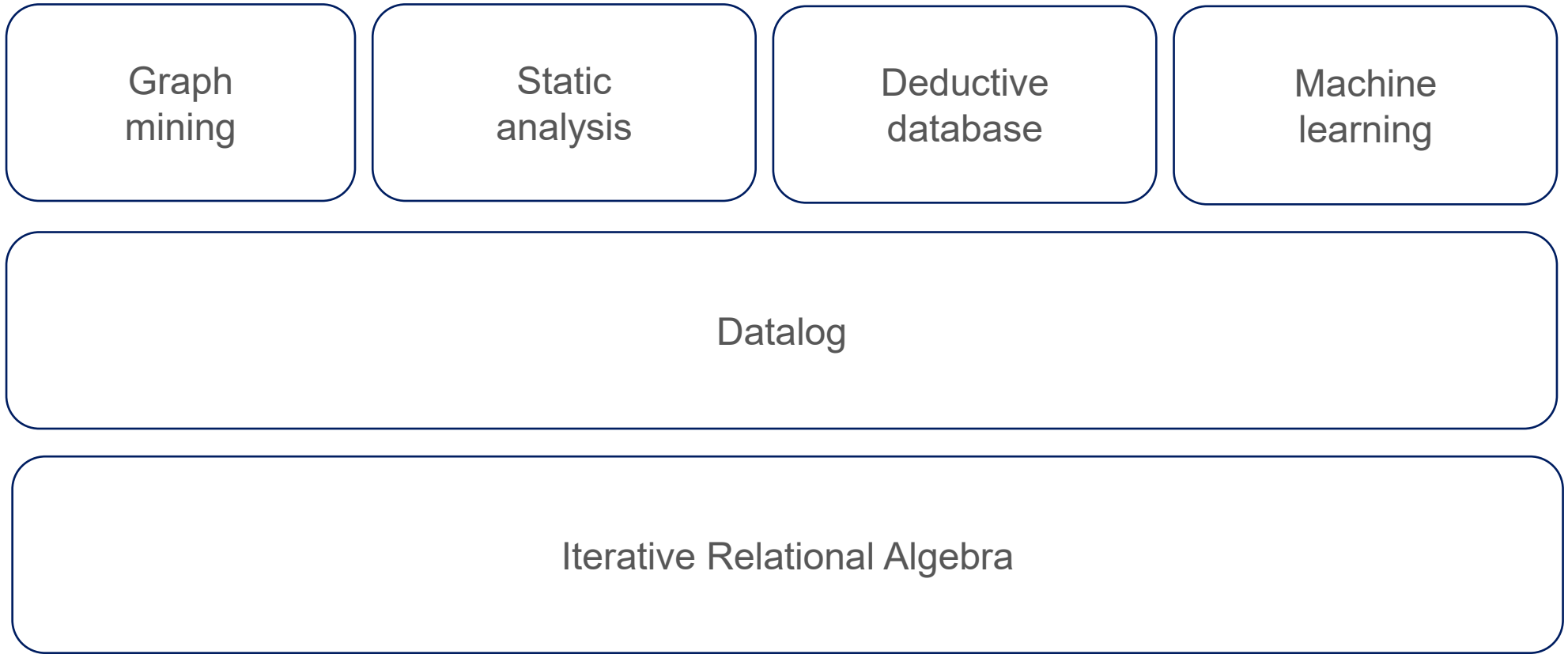
Applications

Transitive closure	tc(X, Y) :- edges(X, Y). tc(X, Z) :- tc(X, Y), edges(Y, Z).
Connected components	cc(X, X) :- edges(X, _). cc(Y, $MIN(Z)) :- cc(Y, Z), edges(X, Y).
Triangle counting	2cl(X, Y) :- edges(X, Y), X < Y. 2cl(X, Y) :- edges(Y, X), X < Y. triangles(X, Y, Z) :- 2cl(X, Y), 2cl(Y, Z), 2cl(X, Z).
Same generation	sg(x, y) :- edges(p, x), edges(p, y), x ≠ y. sg(x, y) :- edges(a, x), sg(a, b), edges(b, y), x ≠ y.



- De Moor, O., Gottlob, G., Furche, T., & Sellers, A. (Eds.). (2012). Datalog Reloaded: First International Workshop, Datalog 2010, Oxford, UK, March 16-19, 2010. Revised Se Springer.
- Huang, S. S., Green, T. J., & Loo, B. T. (2011, June). Datalog and emerging applications: an interactive tutorial. In Proceedings of the 2011 ACM SIGMOD International Conf (pp. 1213-1216).
- Gilray, T., & Kumar, S. (2017). Toward parallel cfa with datalog, mpi, and cuda. In Scheme and Functional Programming Workshop.
- Zomorodian, A. (2012). Topological data analysis. Advances in applied and computational topology, 70, 1-39.

Datalog's Applications



Limitations of Current Datalog Engines

Scalability	Framework	CPU-only	VRAM
Multi-threaded	Distributed (Apache Spark)	Multi-node Multi-threaded	Single-GPU
Soufflé	RDFox	SLOG	GPUJoin
LogicBlox	Radlog	PRAM	GPULog
Nemo	BigDatalog		GPUDatalog

Multi-node Multi-GPU

- Herbert Jordan, Bernhard Scholz, and Pavle Subotić. 2016. Soufflé: On synthesis of program analyzers. In Computer Aided Verification: 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II 28, Swarat Chaudhuri and Azadeh Farzan (Eds.). Springer, Springer International Publishing, Cham, 422–430.
- Boris Motik, Yavor Nenov, Robert Piro, Ian Horrocks, and Dan Olteanu. 2014. Parallel materialisation of datalog programs in centralised, main-memory RDF systems. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 28.
- Shkapsky, A., Yang, M., Interlandi, M., Chiu, H., Condie, T., & Zaniolo, C. (2016, June). Big data analytics with datalog queries on spark. In Proceedings of the 2016 International Conference on Management of Data (pp. 1135-1149).
- Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. 2015. RDFox: A Highly-Scalable RDF Store. In The Semantic Web - ISWC 2015, Marcelo Arenas, Oscar Corcho, Elen Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Steffen Staab (Eds.). Springer International Publishing, Cham, 3–20.
- Thomas Gilray, Arash Sahebolamri, Yihao Sun, Sowmith Kunapaneni, Sidharth Kumar, and Kristopher Micinski. 2024. Datalog with First-Class Facts. Proc. VLDB Endow. 18, 3 (November 2024), 651–665.

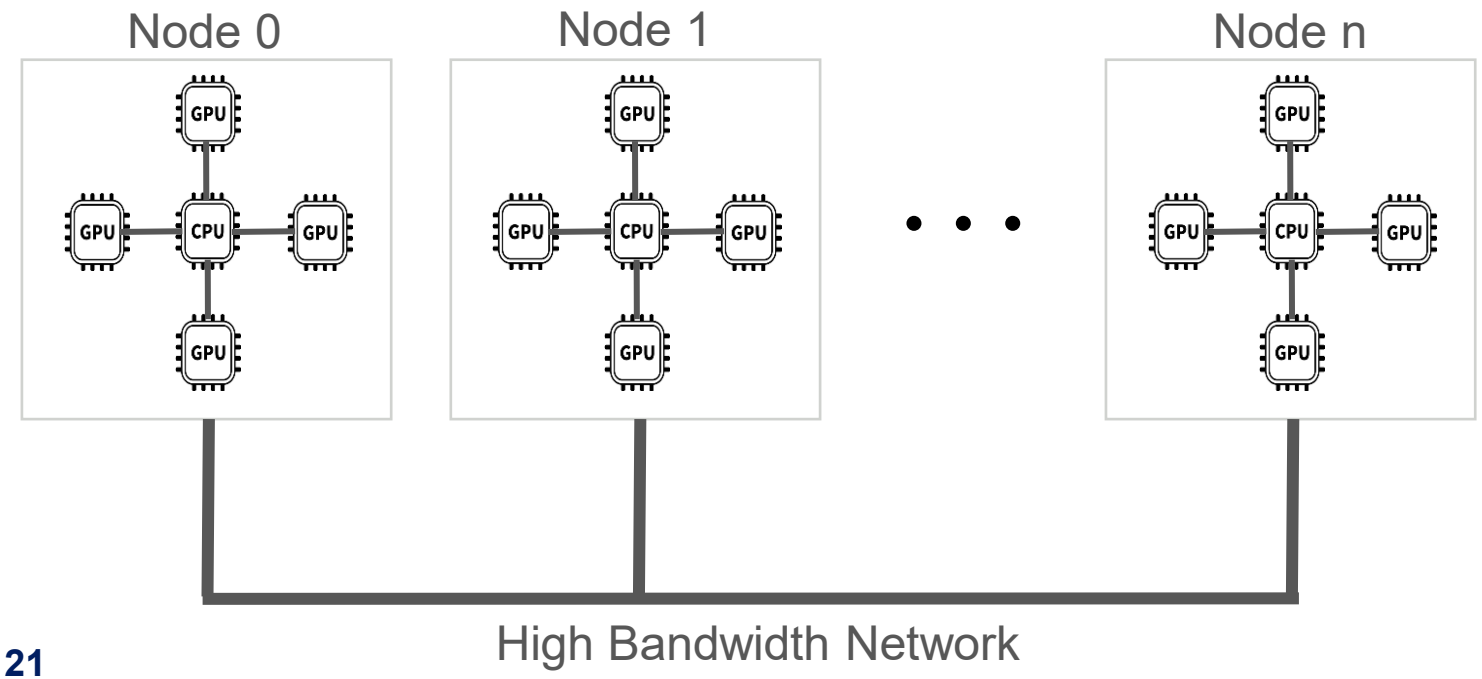
UIC

Motivation: Datalog on Multi-Node Multi-GPU Systems

Datalog

Relational algebra

Multi-node multi-GPU systems



High performance implementation for highly expressive language

Selected publications

Conference Proceedings

Towards iterative relational algebra on the {GPU}

Acceptance rate: 19%

USENIX ATC '23

Bruck Algorithm Performance Analysis for Multi-GPU All-to-All Communication

Acceptance rate: 44%

HPC Asia 2024

Optimizing Datalog for the GPU

Acceptance rate: 15%

ASPLOS 2025

Multi-node multi-GPU Datalog

Acceptance rate: 23%

ACM ICS '25

Journal Articles

The robustness of persistent homology of brain networks to data acquisition-related non-neural variability in resting state fMRI

Impact factor: 5.0

HUMAN BRAIN MAPPING

UALCAN: An update to the integrated cancer data analysis platform

Impact factor: 6.3

NEOPLASIA

Workshop Papers

Accelerating Datalog applications with cuDF

Acceptance rate: 42%

IA^3 2022

Scalable, interactive and hierarchical visualization of virus taxonomic data

VAHC 2023

Submission

Topology assisted clustering of temporal fMRI brain networks with use-case in mitigating non-neural multi-site variability

IEEE Access

Selected publications

Conference Proceedings

Towards iterative relational algebra on the {GPU}

Acceptance rate: 19%

USENIX ATC '23

Bruck Algorithm Performance Analysis for Multi-GPU All-to-All Communication

Acceptance rate: 44%

HPC Asia 2024

Optimizing Datalog for the GPU

Acceptance rate: 15%

ASPLOS 2025

Multi-node multi-GPU Datalog

Acceptance rate: 23%

ACM ICS '25

Journal Articles

The robustness of persistent homology of brain networks to data acquisition-related non-neural variability in resting state fMRI

Impact factor: 5.0

HUMAN BRAIN MAPPING

UALCAN: An update to the integrated cancer data analysis platform

Impact factor: 6.3

NEOPLASIA

Workshop Papers

Accelerating Datalog applications with cuDF

Acceptance rate: 42%

IA^3 2022

Scalable, interactive and hierarchical visualization of virus taxonomic data

VAHC 2023

Submission

Topology assisted clustering of temporal fMRI brain networks with use-case in mitigating non-neural multi-site variability

IEEE Access

Roadmap

1. Introduction and motivation
2. **GPU based Datalog engine development**
3. Multi-node multi-GPU Datalog engine development
4. GPU portability of Datalog engines
5. Power analysis of Datalog engines
6. Exploration of high-dimensional scientific analytics
7. Future work
8. Conclusion

GPU based Datalog engine development

Off-the-shelf

Started with CPU and GPU based Python libraries

Compare GPU capabilities for iterative join operations

IA^3 2022

GPUJoin

Relational Algebra (RA) operations using CUDA

High performance GPU hashtable for iterative RA

GPULog

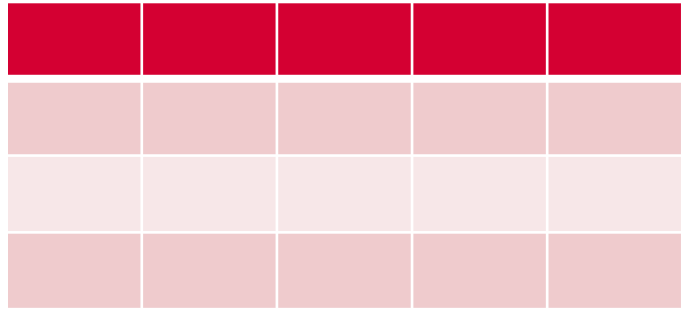
CUDA based SIMD API for deductive analytics

Novel Hash-Indexed Sorted Array data structure

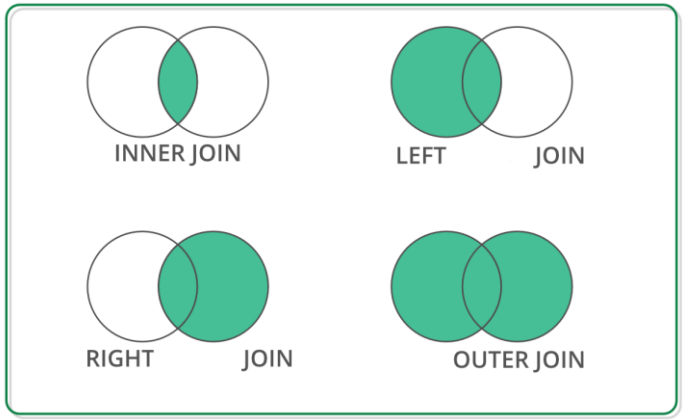
• Shovon, A. R., Dyken, L. R., Green, O., Gilray, T., & Kumar, S. (2022, November). Accelerating datalog applications with cudf. In 2022 IEEE/ACM Workshop on Irregular Applications: Architectures and Algorithms (IA3) (pp. 41-45). IEEE.

Off-the-shelf Data Structure for Join Operation

DataFrame: tabular data structure



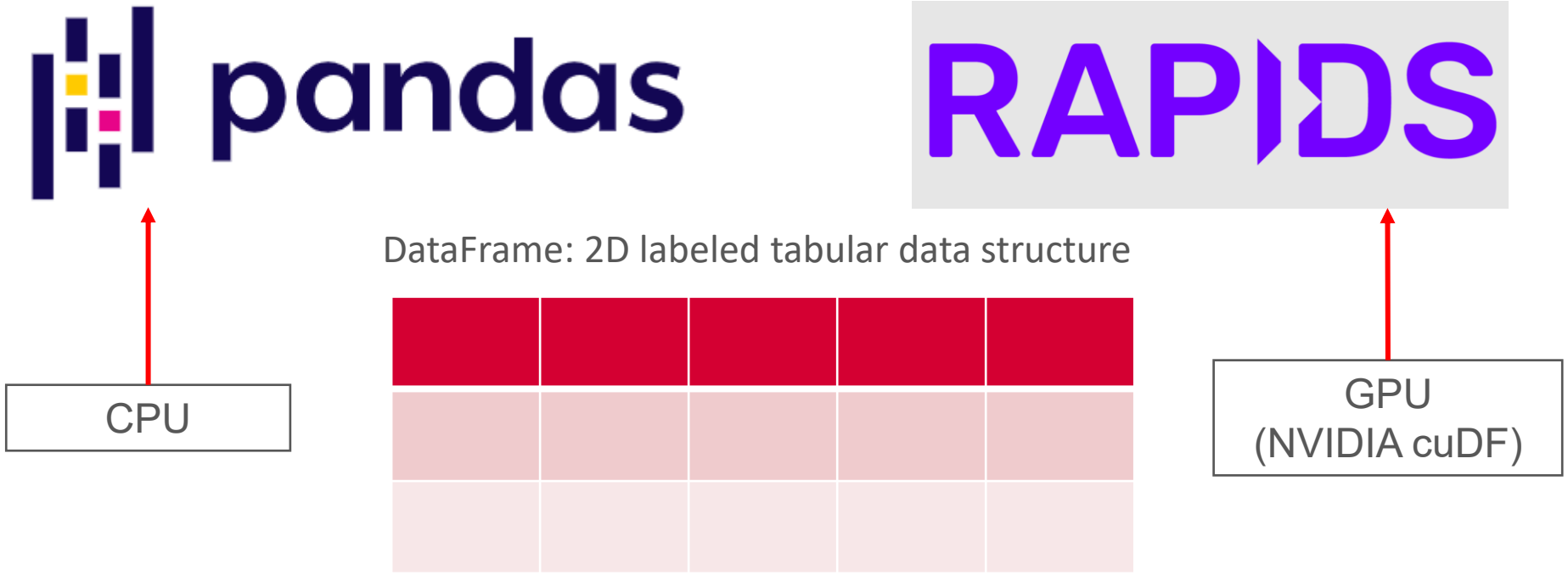
DataFrame has RA primitives APIs



- Reback, J., McKinney, W., Van Den Bossche, J., Augspurger, T., Cloud, P., Klein, A., ... & Seabold, S. (2020). pandas-dev/pandas: Pandas 1.0. 5. Zenodo.
- Chen, D. Y. (2017). Pandas for everyone: Python data analysis. Addison-Wesley Professional.
- Singh, R. (2020, July 1). Merging DataFrames with Pandas: Pd.merge(). Medium. Retrieved April 8, 2023, from https://medium.com/swlh/merging-dataframes-with-pandas-pd-merge-7764c7e2d46d

UIC

Off-the-shelf Python Libraries



Both supports join operation with similar APIs

- Reback, J., McKinney, W., Van Den Bossche, J., Augspurger, T., Cloud, P., Klein, A., ... & Seabold, S. (2020). pandas-dev/pandas: Pandas 1.0. 5. Zenodo.
- Chen, D. Y. (2017). Pandas for everyone: Python data analysis. Addison-Wesley Professional.
- Green, O., Du, Z., Patel, S., Xie, Z., Liu, H., & Bader, D. A. (2021, December). Anti-Section Transitive Closure. In 2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC) (pp. 192-201). IEEE.
- Fender, A., Rees, B., & Eaton, J. RAPIDS cuGraph. In Massive Graph Analytics (pp. 483-493). Chapman and Hall/CRC.

CPU (Pandas) and GPU (cuDF)

```
import pandas as pd
import cudf

def get_read_csv(filename, method='cudf', n):
    column_names = ['column 1', 'column 2']
    if method == 'df':
        return pd.read_csv(filename, sep='\t', header=None,
                           names=column_names, nrows=n)
    return cudf.read_csv(filename, sep='\t', header=None,
                         names=column_names, nrows=n)

def get_join(relation_1, relation_2):
    column_names = ['column 1', 'column 2']
    return relation_1.merge(relation_2, on=column_names[0],
                            how="inner",
                            suffixes=('_relation_1', '_relation_2'))
```

CPU Environment

GPU Environment

- https://github.com/harp-lab/GPUJoin/edit/main/IA3_2022/basic_operations.py

Evaluation environment, applications, datasets

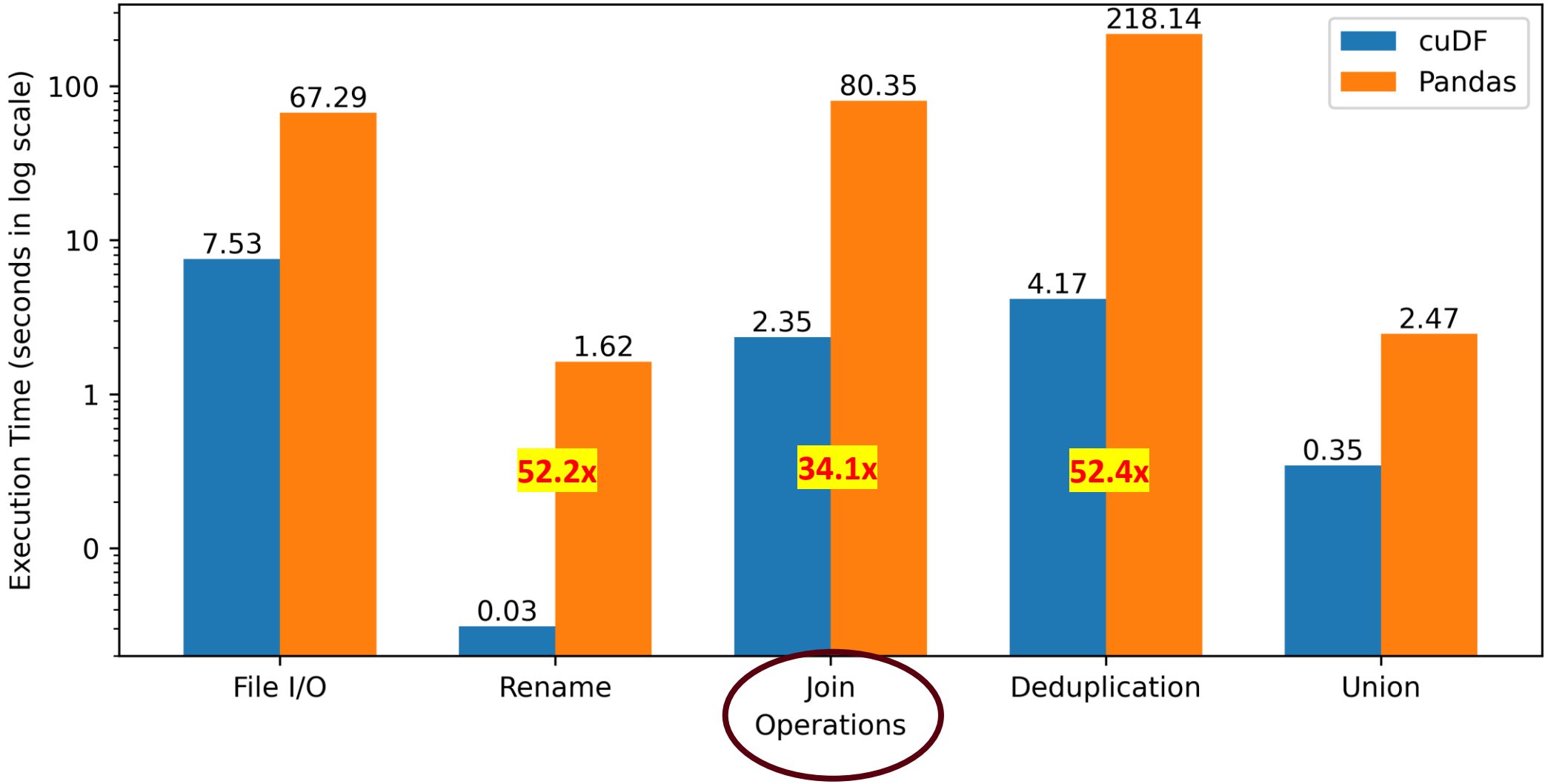
Apps: Transitive Closure, Triangle Counting

Datasets: Stanford large network, SuiteSparse, Road network

- Argonne Leadership Computing Facility. 2022. Theta. https://www.alcf.anl.gov/alcf-resources/theta
- Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data
- Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. ACM Trans. Math. Softw. 38, 1, Article 1 (dec 2011), 25 pages. doi:10.1145/2049662.2049663

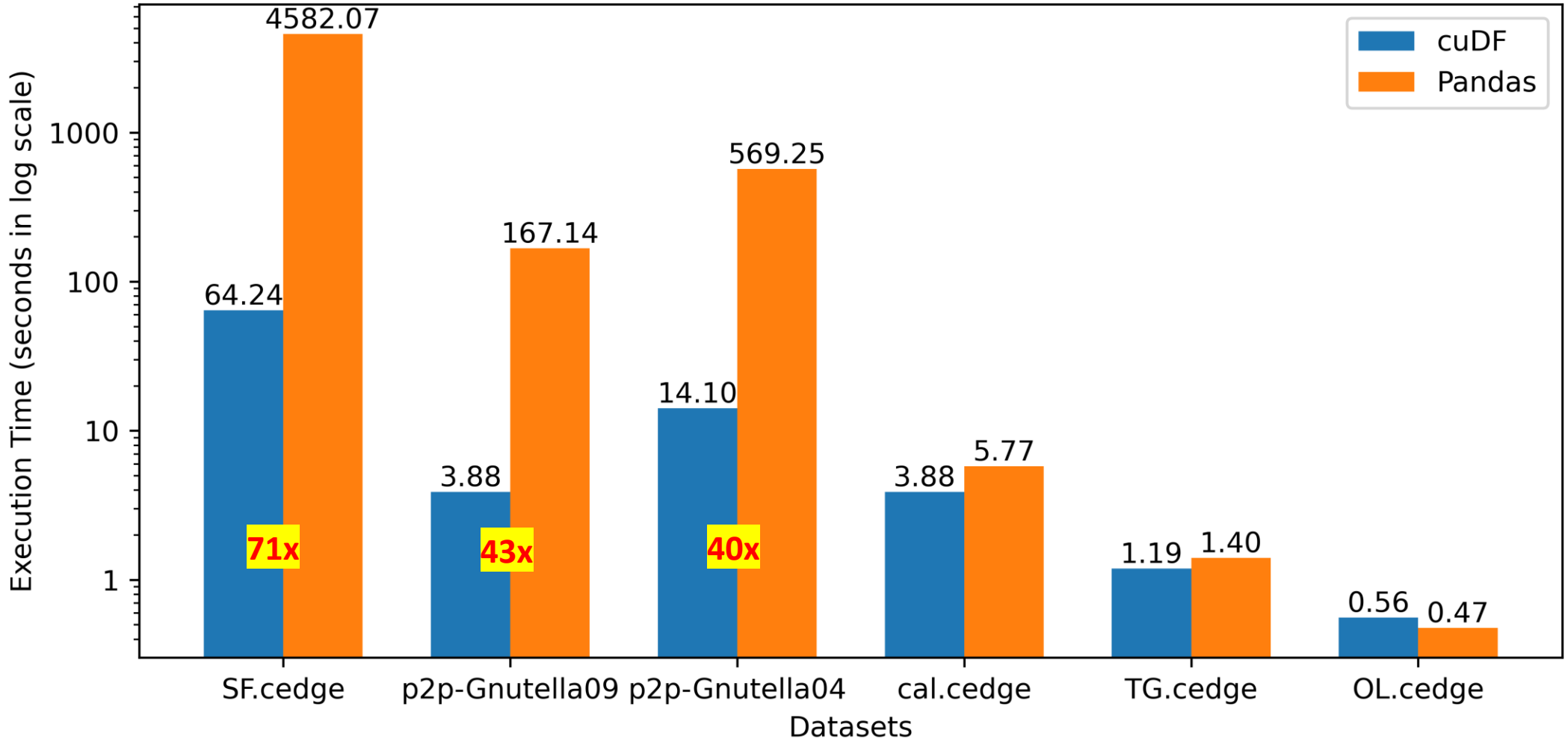
UIC

Performance Improvement for cuDF over Pandas (Standalone Operations)



• Shovon, A. R., Dyken, L. R., Green, O., Gilray, T., & Kumar, S. (2022, November). Accelerating datalog applications with cudf. In 2022 IEEE/ACM Workshop on Irregular Applications: Architectures and Algorithms (IA3) (pp. 41-45). IEEE.

Performance Improvement for cuDF over Pandas (Transitive Closure)



• Shovon, A. R., Dyken, L. R., Green, O., Gilray, T., & Kumar, S. (2022, November). Accelerating datalog applications with cudf. In 2022 IEEE/ACM Workshop on Irregular Applications: Architectures and Algorithms (IA3) (pp. 41-45). IEEE.

Improvement Opportunity

- Open-addressing based hashtable
- Fuse join and projection
- Sorted results for deduplication
- Pinned memory scheme
- Intermediate memory clearance

- A. R. Shovon, L. R. Dyken, O. Green, T. Gilray and S. Kumar, "Accelerating Datalog applications with cuDF," 2022 IEEE/ACM Workshop on Irregular Applications: Architectures and Algorithms (IA3), Dallas, TX, USA, 2022, pp. 41-45
- Green, O., Du, Z., Patel, S., Xie, Z., Liu, H., & Bader, D. A. (2021, December). Anti-Section Transitive Closure. In 2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC) (pp. 192-201). IEEE.

UIC

GPU based Datalog engine development

Off-the-shelf

Started with CPU and GPU based Python libraries

Compare GPU capabilities for iterative join operations

GPUJoin

Relational Algebra (RA) operations using CUDA

High performance GPU hashtable for iterative RA

USENIX ATC '23

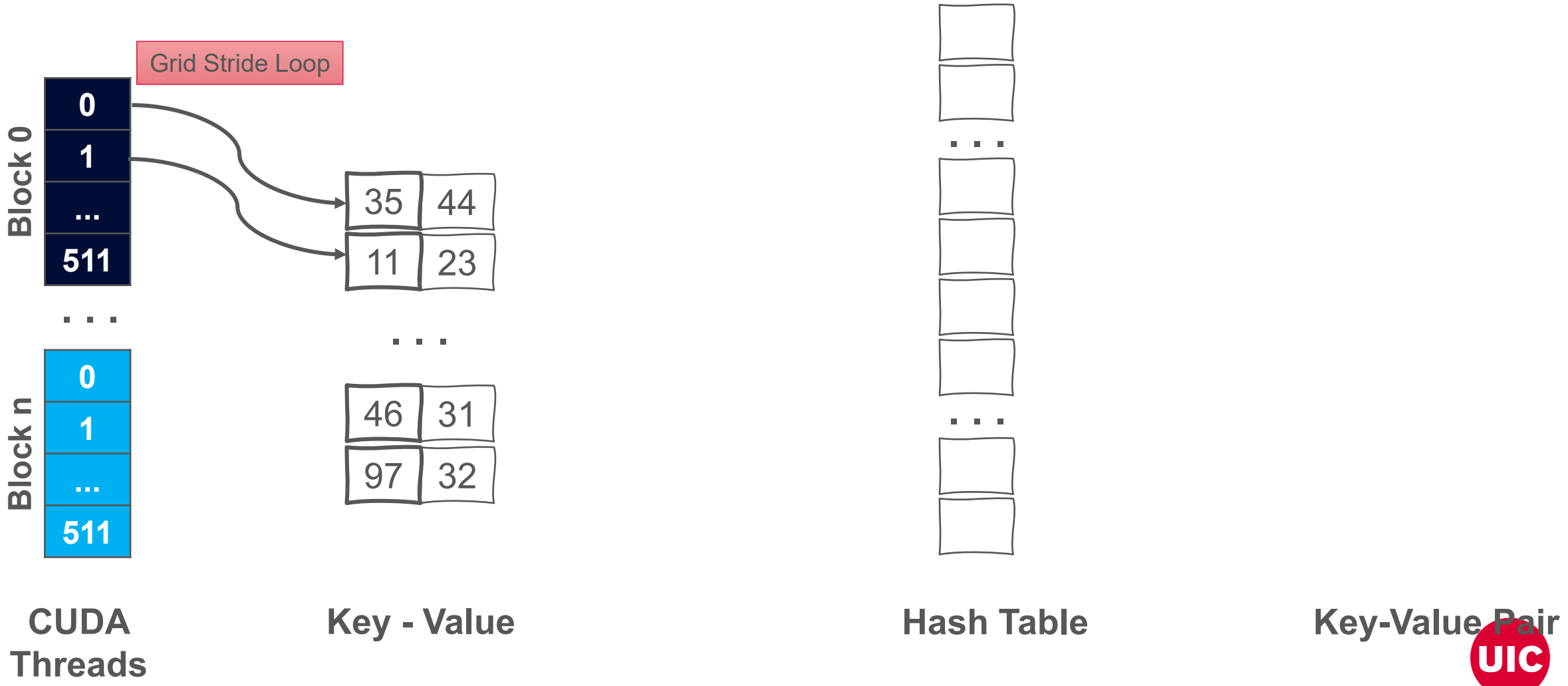
GPULog

CUDA based SIMD API for deductive analytics

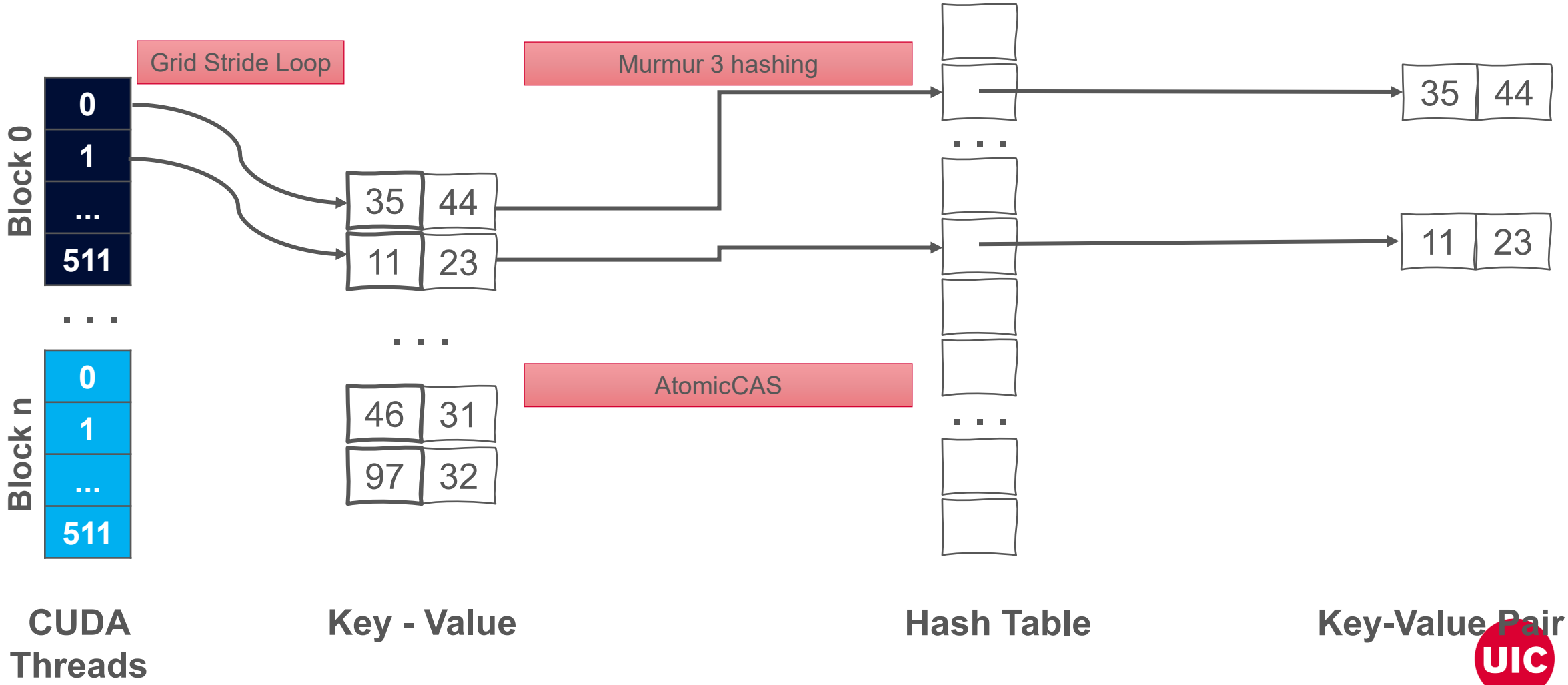
Novel Hash-Indexed Sorted Array data structure

• Shovon, A. R., Gilray, T., Micinski, K., & Kumar, S. (2023). Towards iterative relational algebra on the {GPU}. In 2023 USENIX Annual Technical Conference (USENIX ATC 23) (pp. 1009-1016).

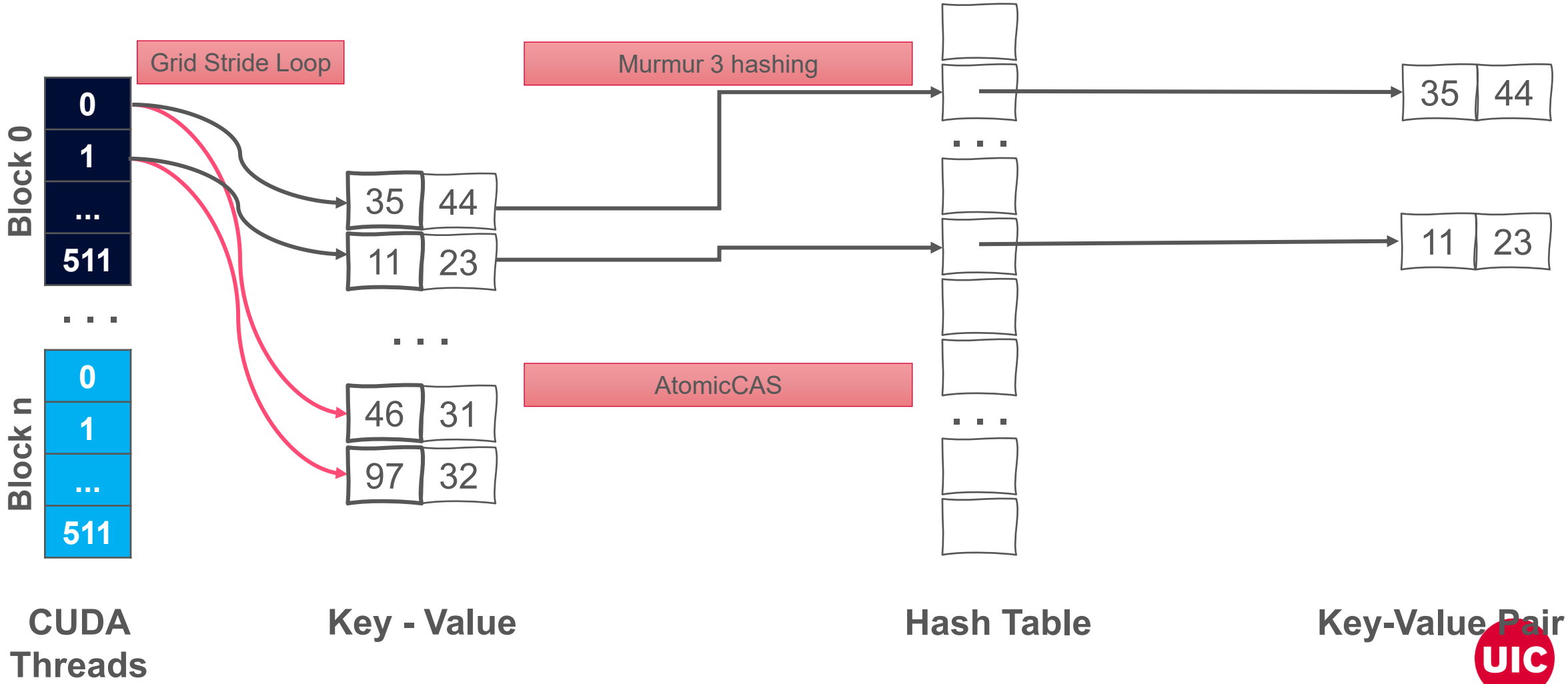
Hash Table (Open Addressing, Linear Probing)



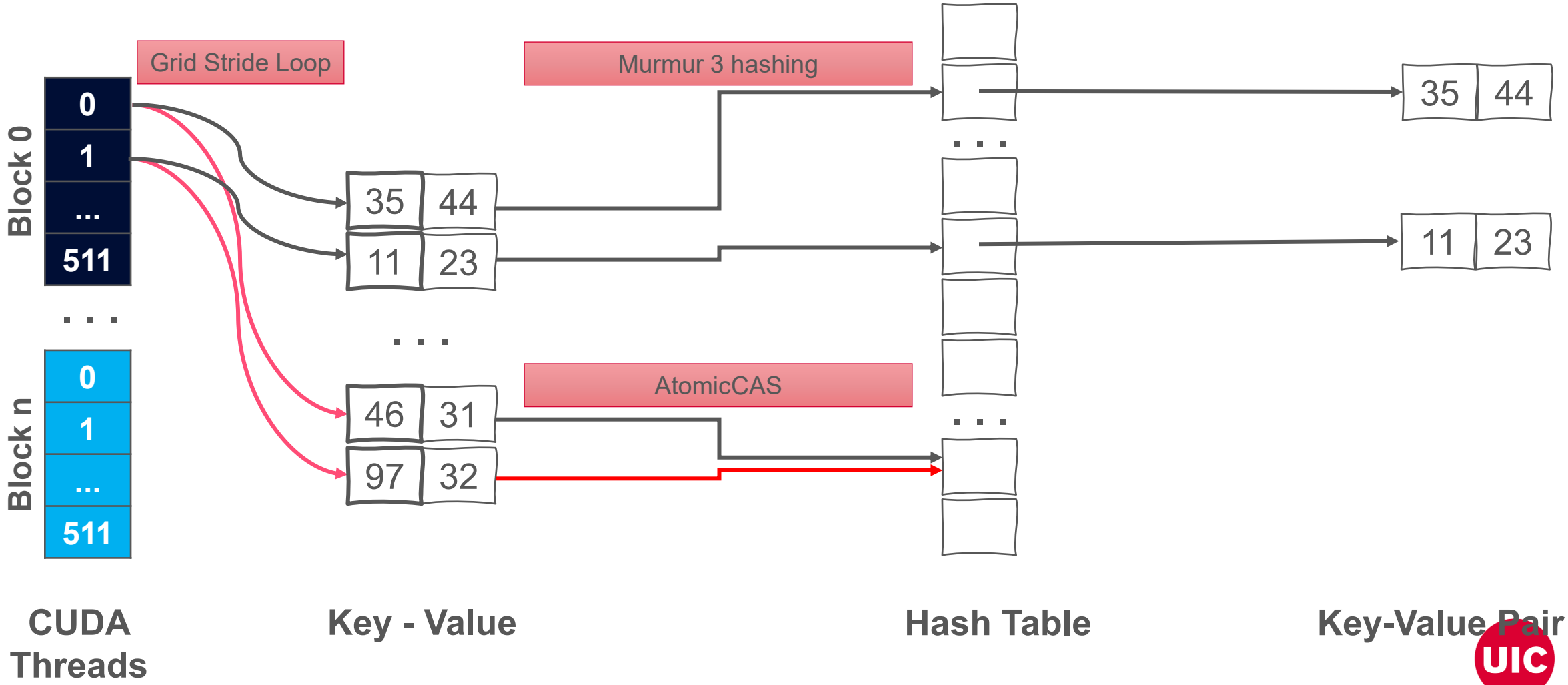
Hash Table (Open Addressing, Linear Probing)



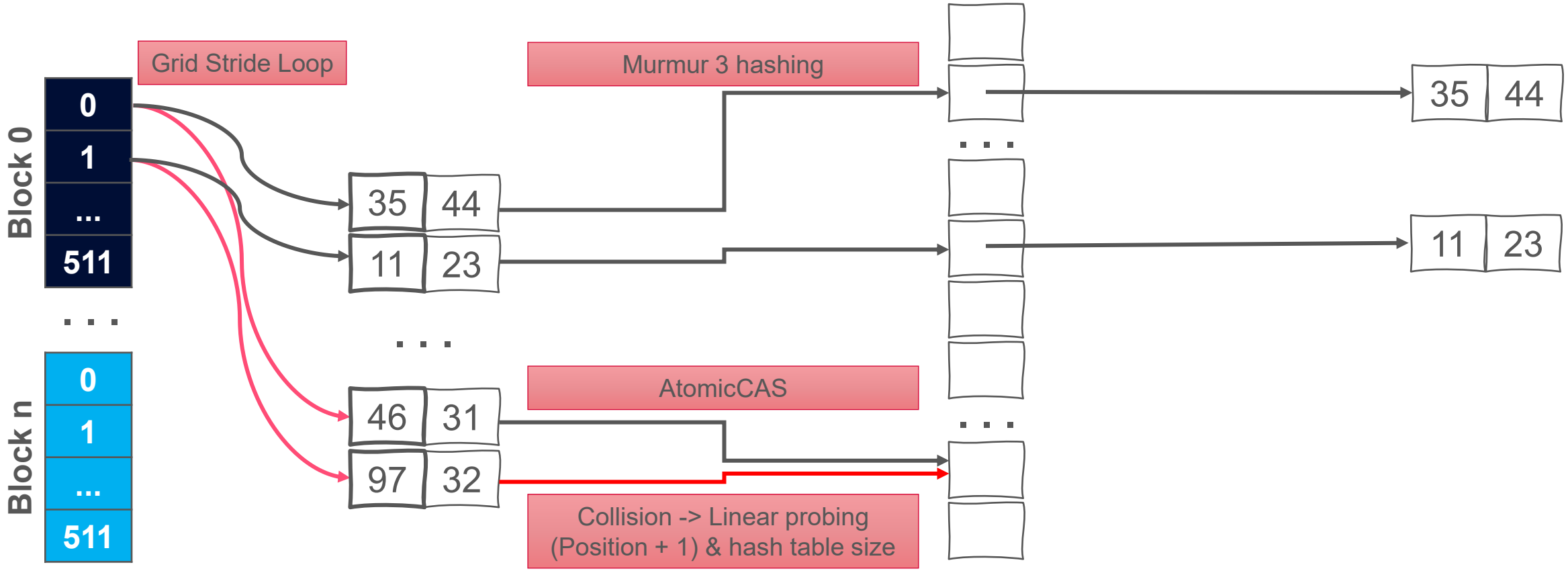
Hash Table (Open Addressing, Linear Probing)



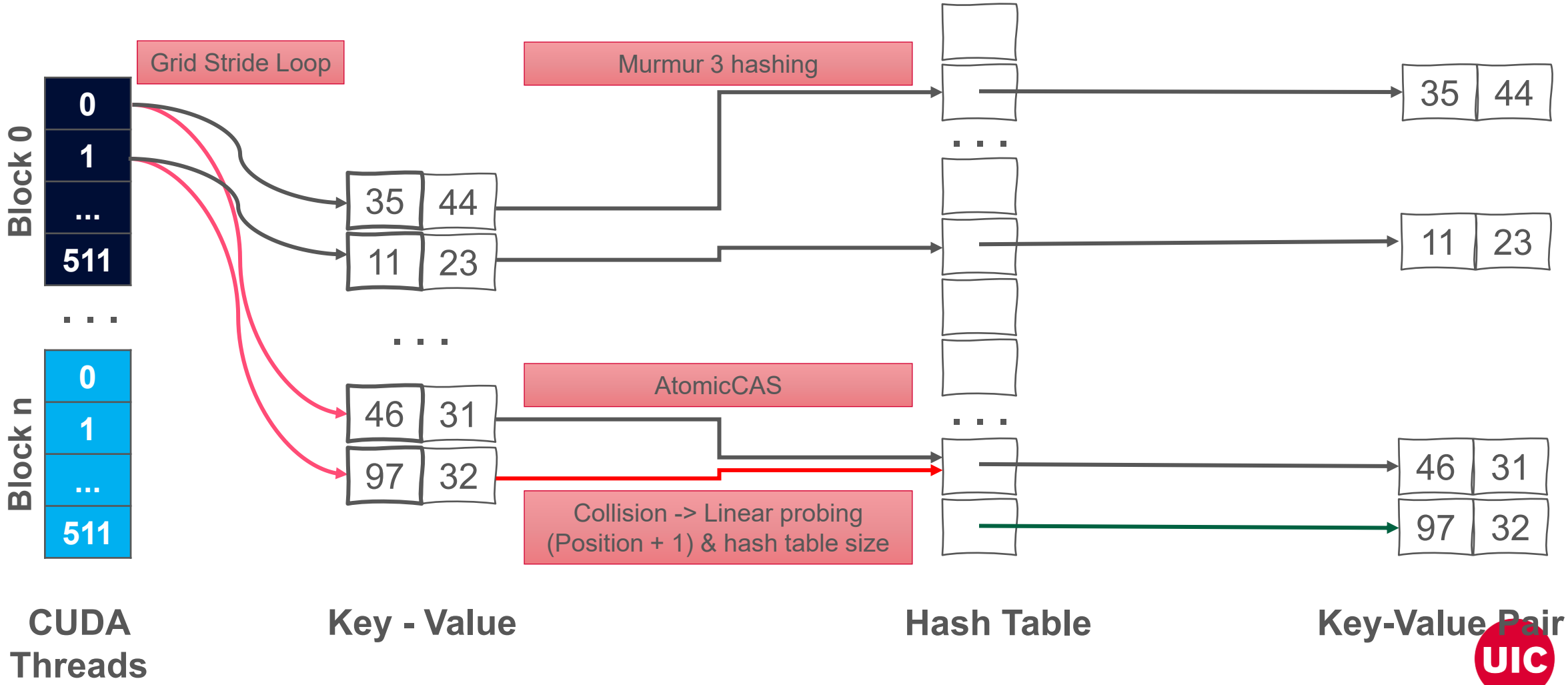
Hash Table (Open Addressing, Linear Probing)



Hash Table (Open Addressing, Linear Probing)

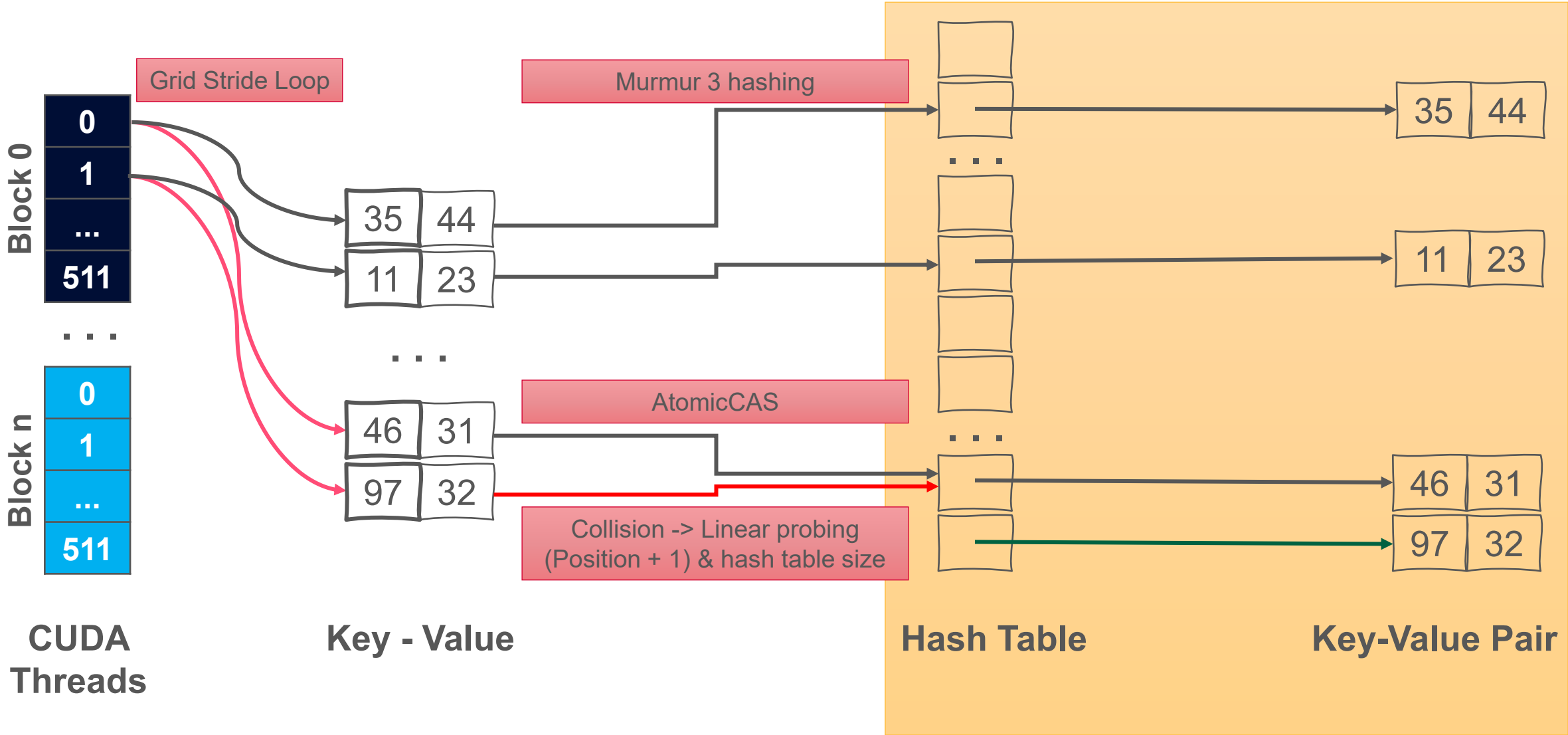


Hash Table (Open Addressing, Linear Probing)



Hash Table (Open Addressing, Linear Probing)

Stores key-value pairs in sparse space
Uses extra memory



Hash Table Performance

Build rate:

- Random synthetic graph: 400 million keys/second
- String graph: 4 billion keys/second

String graph

Performing Hash Join on GPU

$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$

Static Hash Table

Key	Value

⋈

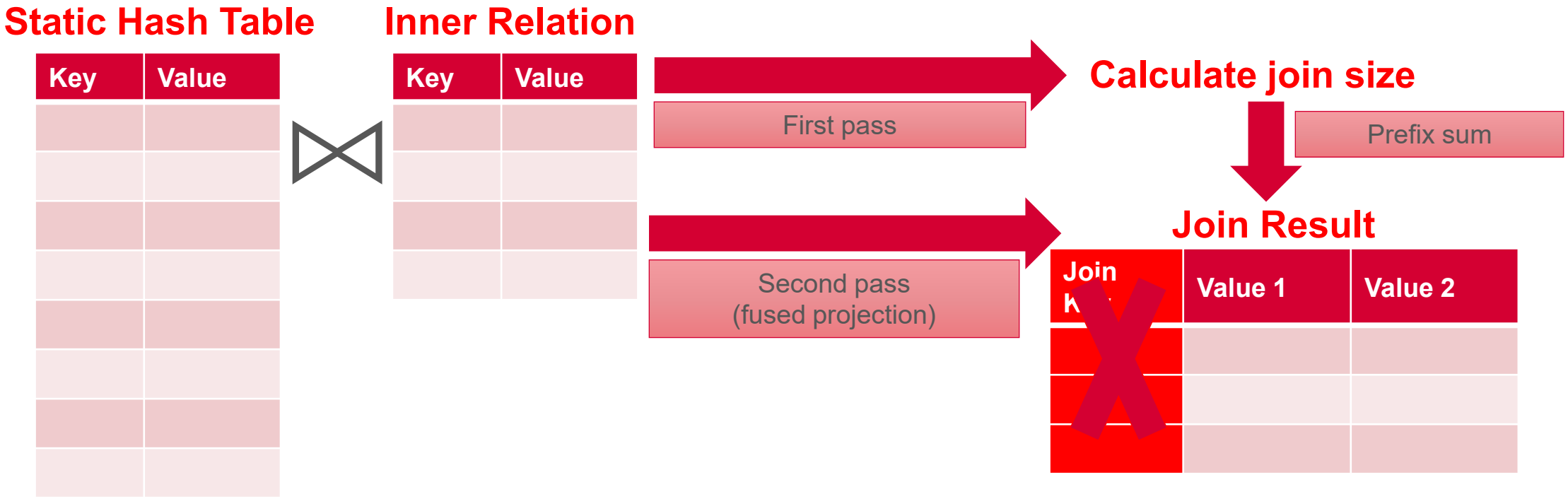
Inner Relation

Key	Value

First pass → **Calculate join size**

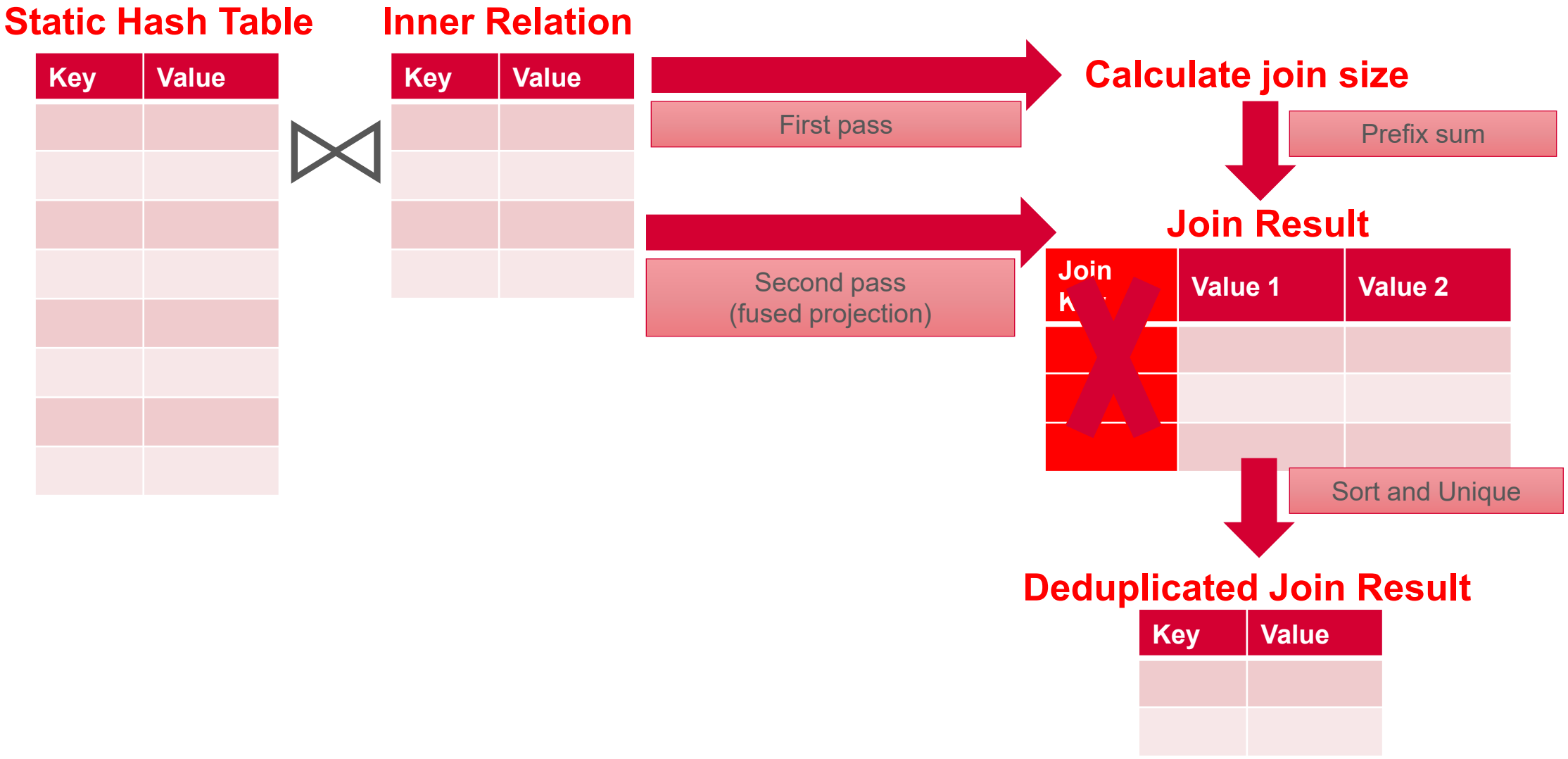
Performing Hash Join on GPU

$$F_G(T) \triangleq G \cup \boxed{\Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)}$$



Performing Hash Join on GPU

$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$



Evaluation environment, applications, datasets

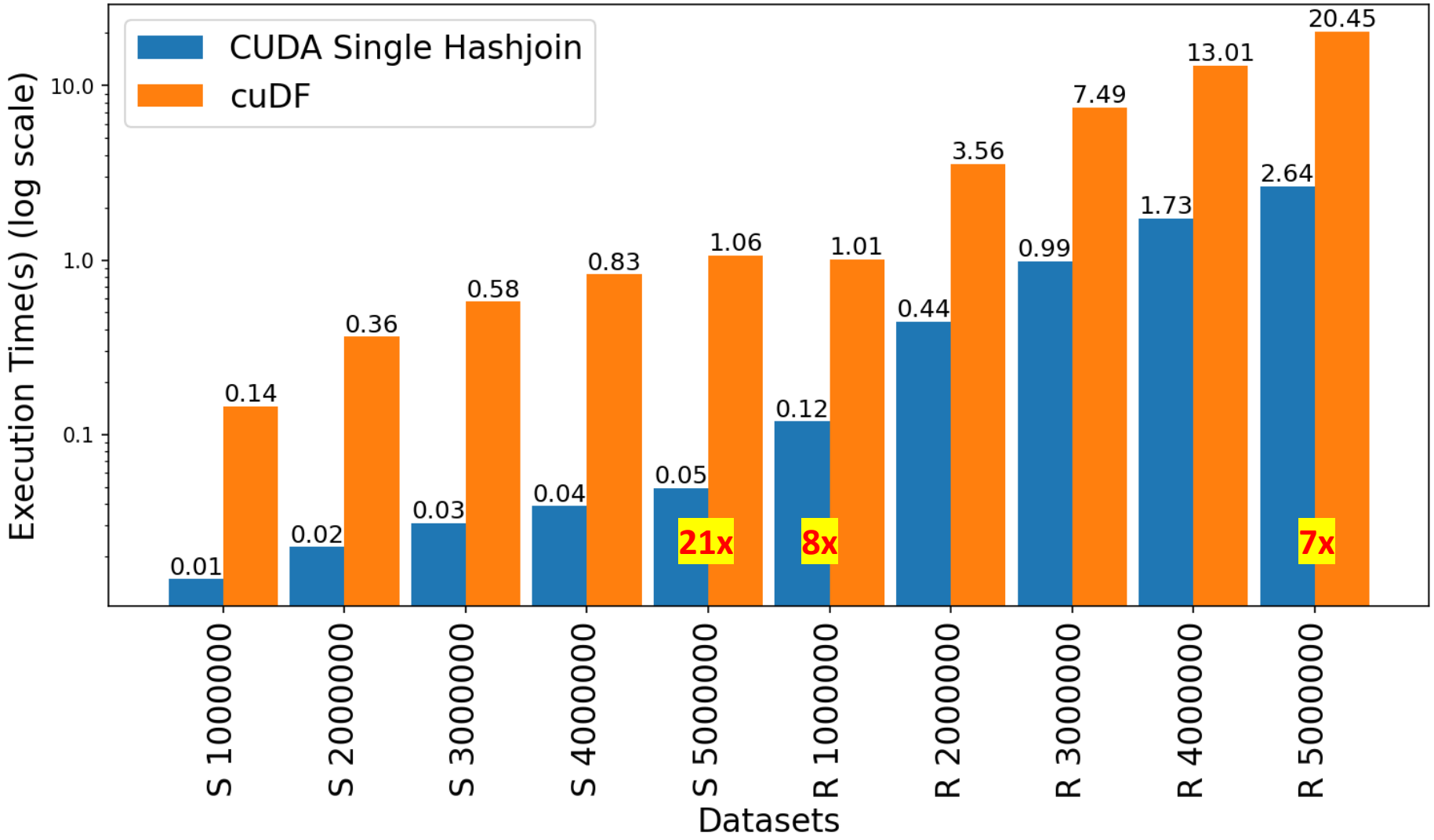
Apps: Transitive Closure, Iterated Hash Join

Datasets: Stanford large network, SuiteSparse, Road network

- Argonne Leadership Computing Facility. 2022. Theta. https://www.alcf.anl.gov/alcf-resources/theta
- Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data
- Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. ACM Trans. Math. Softw. 38, 1, Article 1 (dec 2011), 25 pages. doi:10.1145/2049662.2049663

UIC

Join Performance Comparison: GPUJoin vs cuDF



- Leadership Computing Facility, A. (2022). Argonne Leadership Computing Facility. Theta GPU Nodes. URL: https://www.alcf.anl.gov/support-center/theta-gpu-nodes

Transitive closure performance evaluation

Dataset	Type	Rows	TC size	Iterations	GPUJoin(s)	Soufflé(s)	cuDF(s)
fe_ocean	U	409,593	1,669,750,513	247	138.237	536.233	OOM
p2p-Gnutella31	D	147,892	884,179,859	31	OOM	128.917	OOM
usroads	U	165,435	871,365,688	606	364.554	222.761	OOM
fe_body	U	163,734	156,120,489	188	47.758	29.070	OOM
loc-Brightkite	U	214,078	138,269,412	24	15.880	29.184	OOM
SF.cedge	U	223,001	80,498,014	287	11.274	17.073	64.417
fe_sphere	U	49,152	78,557,912	188	13.159	20.008	80.077
CA-HepTh	D	51,971	74,619,885	18	4.318	15.206	26.115
p2p-Gnutella04	D	39,994	47,059,527	26	2.092	7.537	14.005
p2p-Gnutella09	D	26,013	21,402,960	20	0.720	3.094	3.906
wiki-Vote	D	103,689	11,947,132	10	1.137	3.172	6.841
cti	U	48,232	6,859,653	53	0.295	1.496	3.181
delaunay_n16	U	196,575	6,137,959	101	1.137	1.612	5.596
luxembourg_osm	U	119,666	5,022,084	426	1.322	2.548	8.194
ego-Facebook	U	88,234	2,508,102	17	0.544	0.606	3.719
cal.cedge	U	21,693	501,755	195	0.489	0.455	2.756
TG.cedge	U	23,874	481,121	58	0.198	0.219	0.857
wing	U	121,544	329,438	11	0.085	0.193	0.905
OL.cedge	U	7,035	146,120	64	0.148	0.181	0.523

3.9x

10x

- Leadership Computing Facility, A. (2022). Argonne Leadership Computing Facility. Theta GPU Nodes. URL: https://www.alcf.anl.gov/support-center/theta-gpu-nodes

UIC

Fallback: Memory usage

GPU based Datalog engine development

Off-the-shelf

Started with CPU and GPU based Python libraries

Compare GPU capabilities for iterative join operations

GPUJoin

Relational Algebra (RA) operations using CUDA

High performance GPU hashtable for iterative RA

GPULog

CUDA based SIMD API for deductive analytics

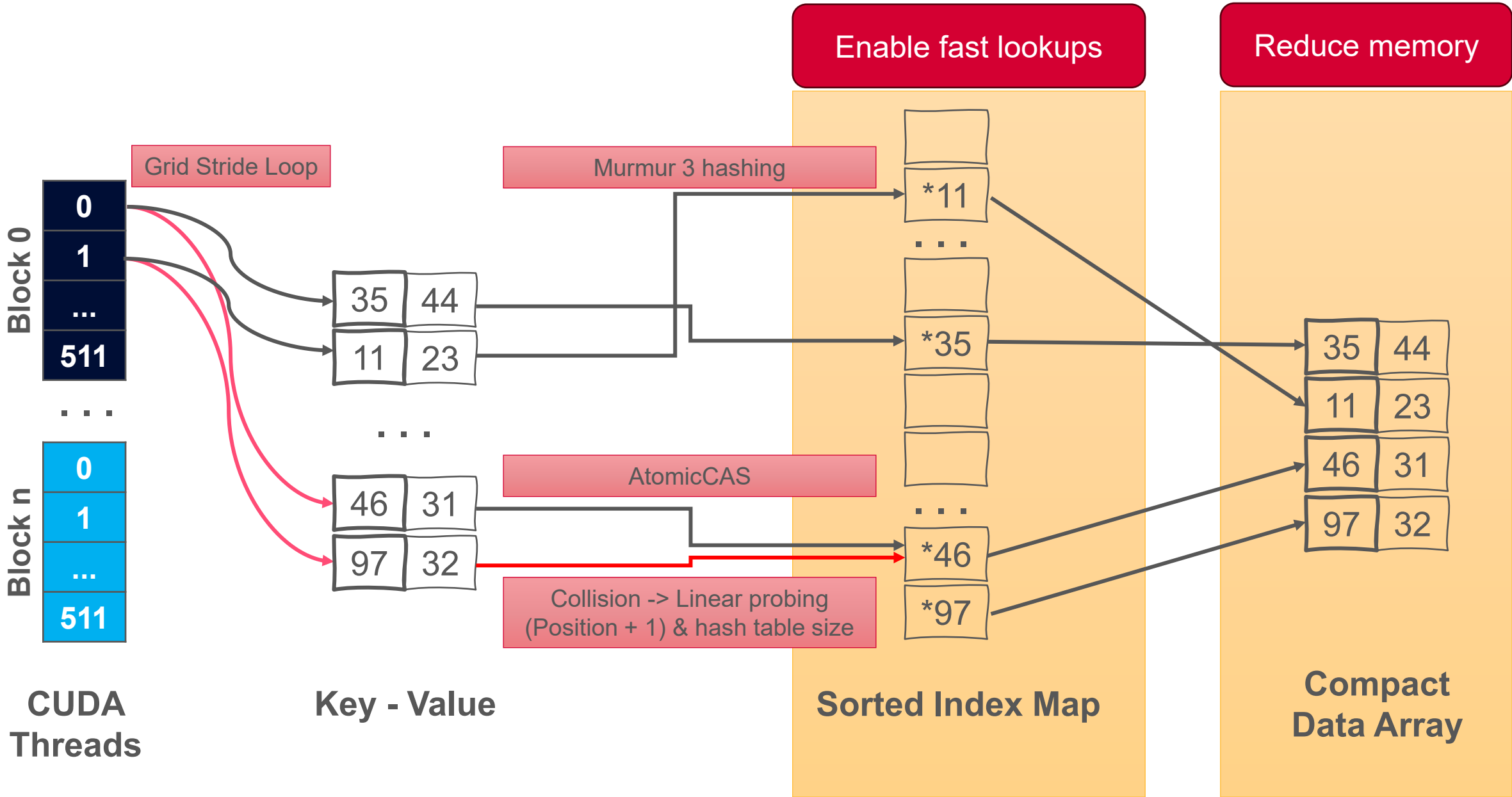
Novel Hash-Indexed Sorted Array data structure

ASPLOS 2025

• Sun, Y., Shovon, A. R., Gilray, T., Kumar, S., & Micinski, K. (2025, March). Optimizing Datalog for the GPU. In Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1 (pp. 762-776).

UIC

Hash Indexed Sorted Array (HISA)

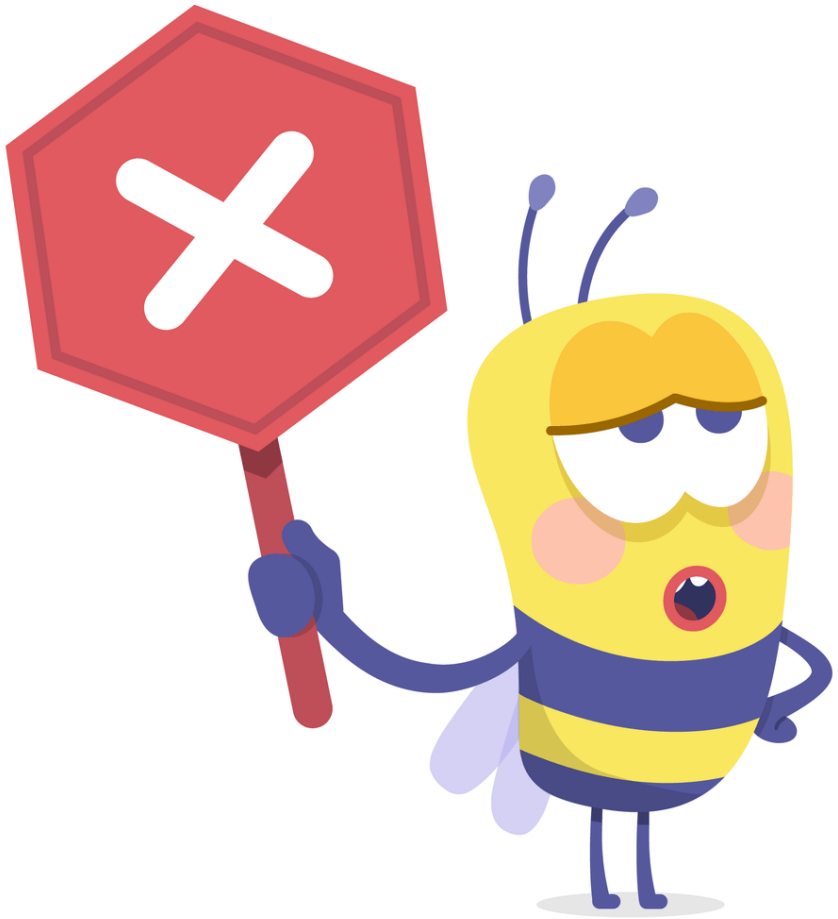


Performance Enhancement (Transitive closure)

			12x to 20x	3x to 6x	39x
Dataset name	*Reach edges*	GPUlog	Soufflé	GPUJoin	cuDF
com-dblp	1.91B	**14.30**	232.99	OOM	OOM
fe_ocean	1.67B	**23.36**	292.15	100.30	OOM
vsp_finan	910M	**21.91**	239.33	125.94	OOM
Gnutella31	884M	**5.58**	96.82	OOM	OOM
fe_body	156M	**3.76**	23.40	22.35	OOM
SF.cedge	80M	**1.63**	33.27	3.76	64.29

- Sun, Y., Shovon, A. R., Gilray, T., Micinski, K., & Kumar, S. Optimizing Datalog for the GPU. ASPLOS 2025 (Accepted).

Limitations



Limited to a single GPU

Memory overflow error for larger graphs

Roadmap

1. Introduction and motivation
2. GPU based Datalog engine development
3. **Multi-node multi-GPU Datalog engine development**
4. GPU portability of Datalog engines
5. Power analysis of Datalog engines
6. Exploration of high-dimensional scientific analytics
7. Future work
8. Conclusion

Multi-node multi-GPU Datalog engine development

MNMGDatalog

Multi-node multi-GPU Datalog engine

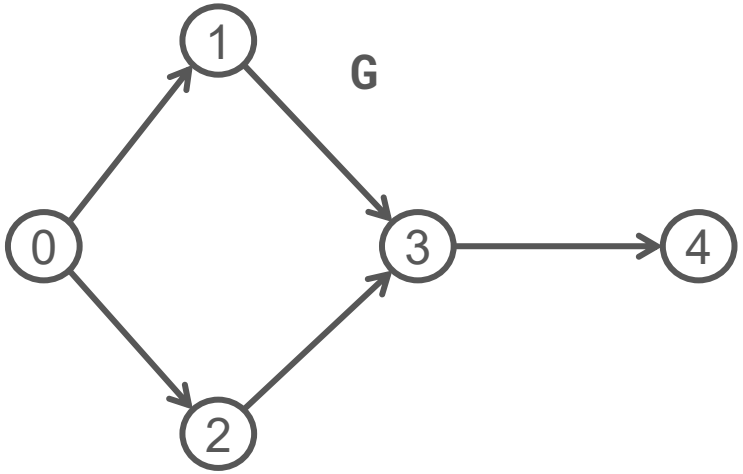
MPI + X (Cuda) design

3rd Best Poster Award at Computational Research Symposium 2025

ACM ICS '25

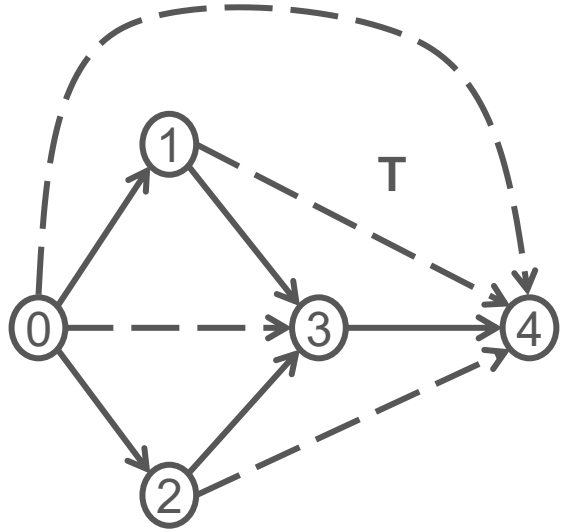
- Shovon, A. R., Sun, Y., Gilray, T., Micinski, K., & Kumar, S. (2025). Multi-Node Multi-GPU Datalog, ICS 2025.

Requirements for Multi-node Multi-GPU Datalog




```
tc(X, Y) :- edges(X, Y).
tc(X, Z) :- tc(X, Y), edges(Y, Z).
```

2 Data representation



0	1
0	2
1	3
2	3
3	4

1 Workload partitioning

3 Communication

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

MNMGDatalog Implementation

Hash-based data distribution

GPU-optimized data representation

GPU-Aware all-to-all communication

1 Workload partitioning

2 Data representation

3 Communication

MNMGDatalog: Radix-hash-based partitioning

Hash-based data distribution

1. Workload partitioning

GPU-optimized data representation

2. Data representation

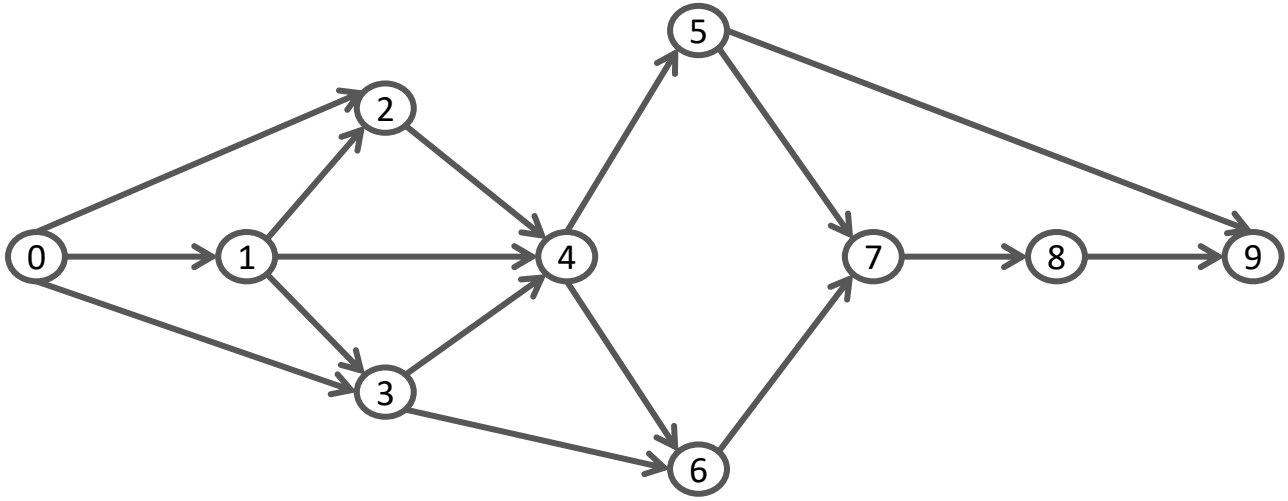
GPU-Aware all-to-all communication

3. Communication

The join column decides the GPU (bucket-id)

(k)

0	1
0	2
0	3
1	2
1	3
1	4
2	4
3	4
3	6
4	5
4	6
5	7
5	9
7	6
7	8
8	9



Hash-partitioning based on the Join Column

Kumar, S., & Gilray, T. (2020). Load-balancing parallel relational algebra. In *High Performance Computing: 35th International Conference, ISC High Performance 2020, Frankfurt/Main, Germany, June 22–25, 2020, Proceedings 35* (pp. 288-308). Springer International Publishing.

The join column decides the GPU (bucket-id)

(k)	
0	1
0	2
0	3
1	2
1	3
1	4
2	4
3	4
3	6
4	5
4	6
5	7
5	9
7	6
7	8
8	9

Hash(k) % 3

0	1
0	2
0	3
1	2
1	3
1	4
2	4
3	4
3	6
4	5
4	6
5	7
5	9
7	6
7	8
8	9

0

0	1
0	2
0	3
3	4
3	6

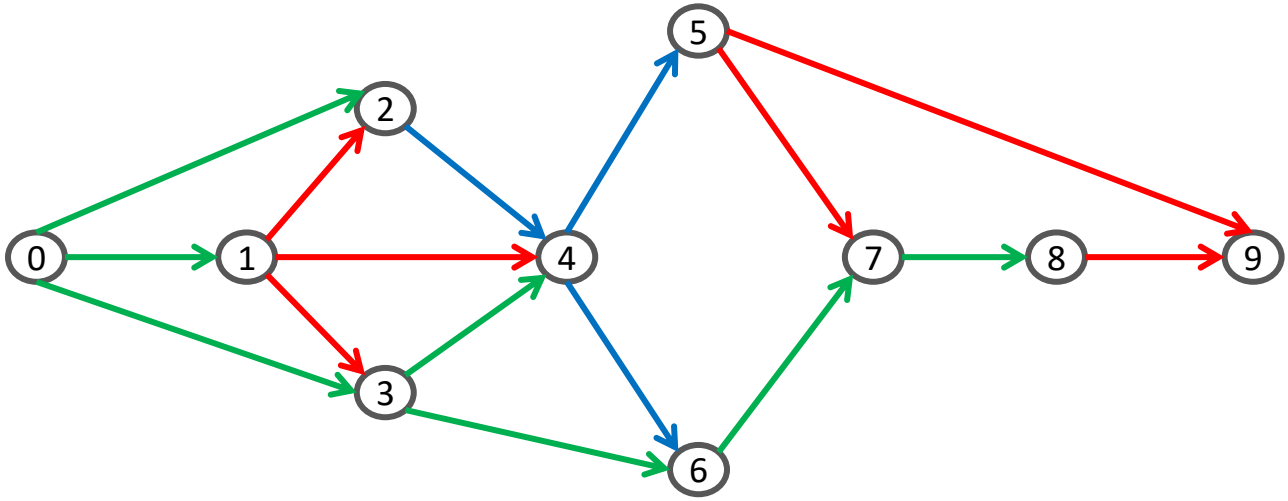
1

7	6
7	8
5	7
5	9
8	9

2

1	2
1	3
1	4
2	4
4	5
4	6

GPU (ID)



Kumar, S., & Gilray, T. (2020). Load-balancing parallel relational algebra. In *High Performance Computing: 35th International Conference, ISC High Performance 2020, Frankfurt/Main, Germany, June 22–25, 2020, Proceedings 35* (pp. 288-308). Springer International Publishing.

MNMGDatalog: Data representation

Hash-based data distribution

GPU-optimized data representation

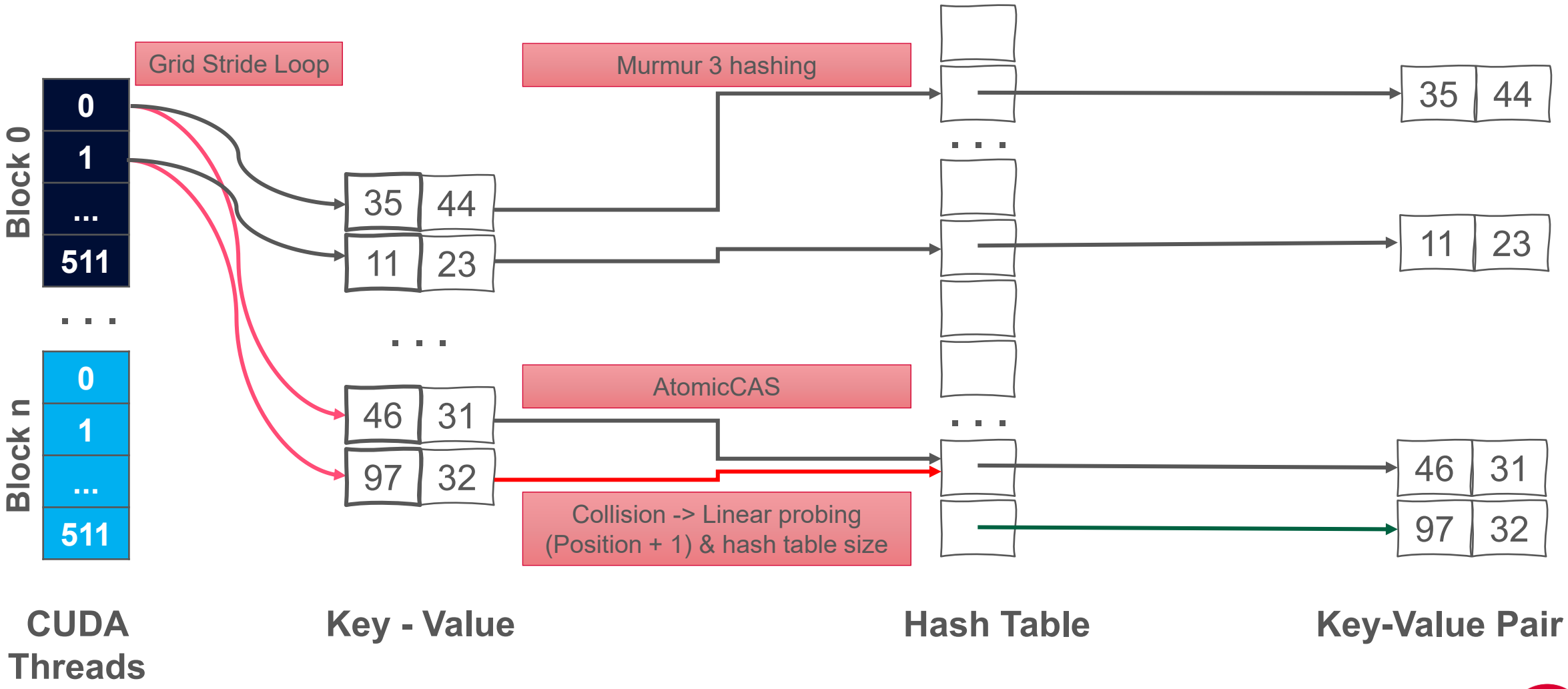
GPU-Aware all-to-all communication

1 Workload partitioning

2 Data representation

3 Communication

GPU Hash Table (Open Addressing, Linear Probing)



Shovon, A. R., Gilray, T., Micinski, K., & Kumar, S. (2023). Towards iterative relational algebra on the {GPU}. In 2023 USENIX Annual Technical Conference (USENIX ATC 23) (pp. 1009-1016).

MNMGDatalog: Communication for Iterative RA

Hash-based data distribution

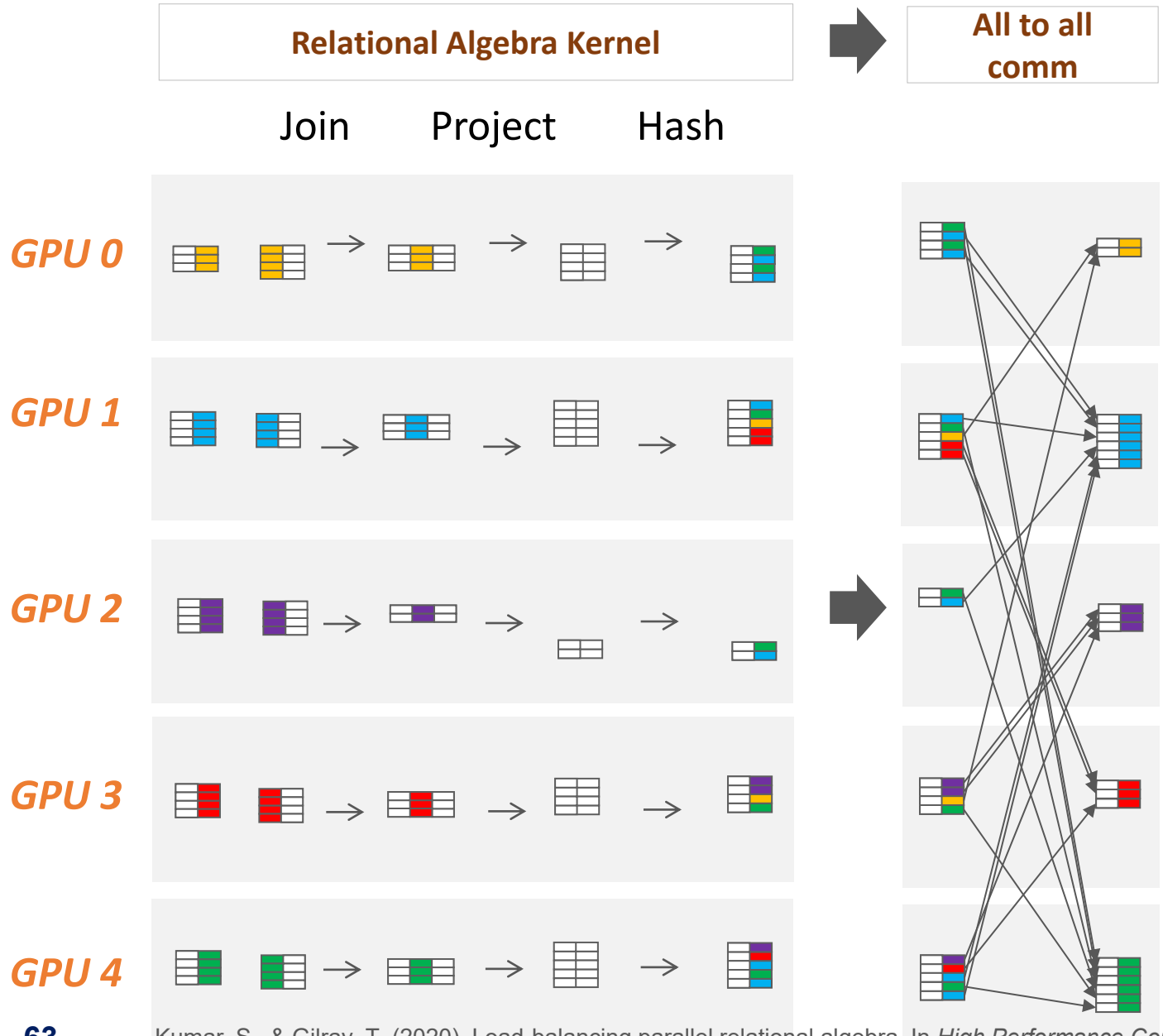
GPU-optimized data representation

GPU-Aware all-to-all communication

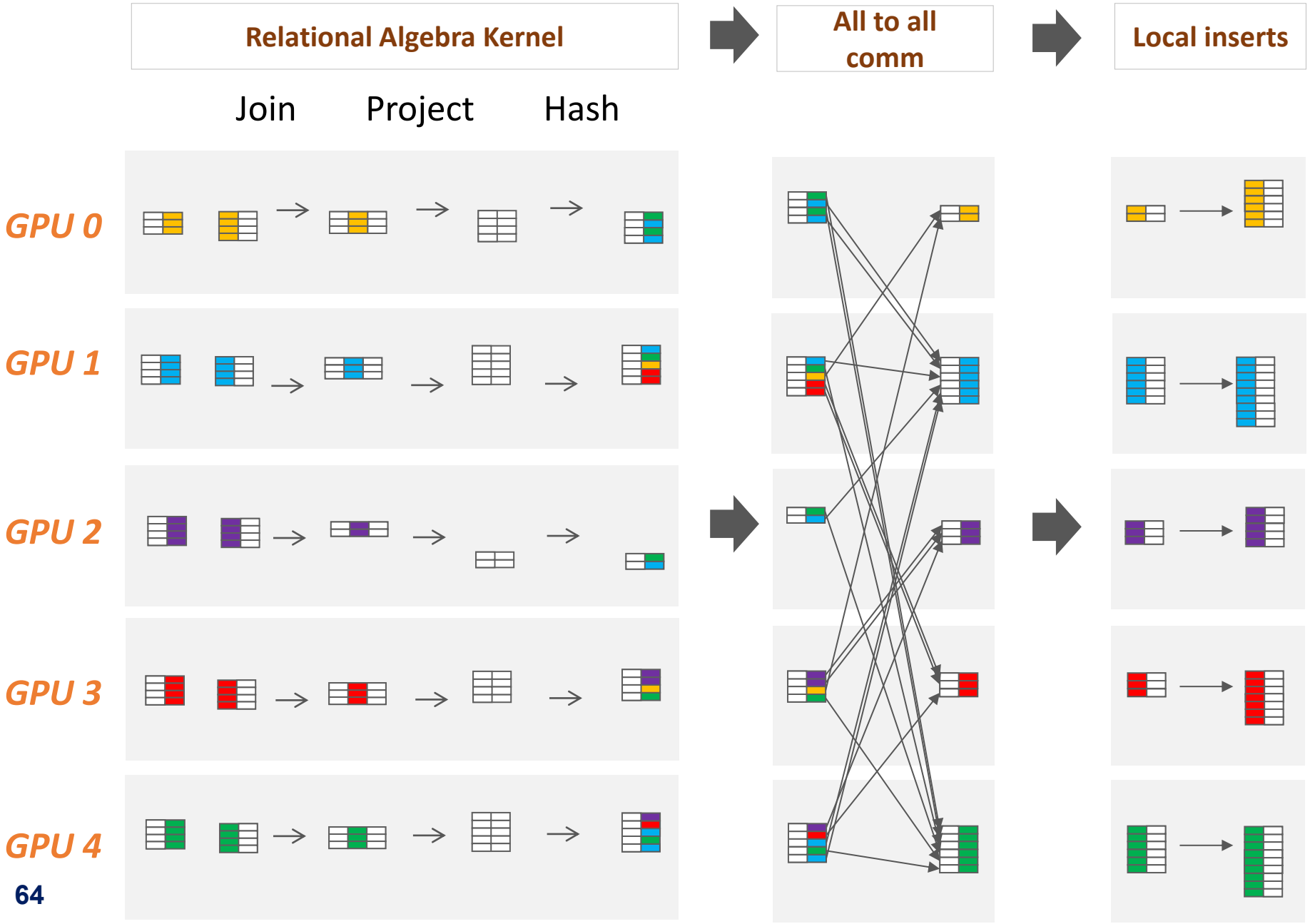
1 Workload partitioning

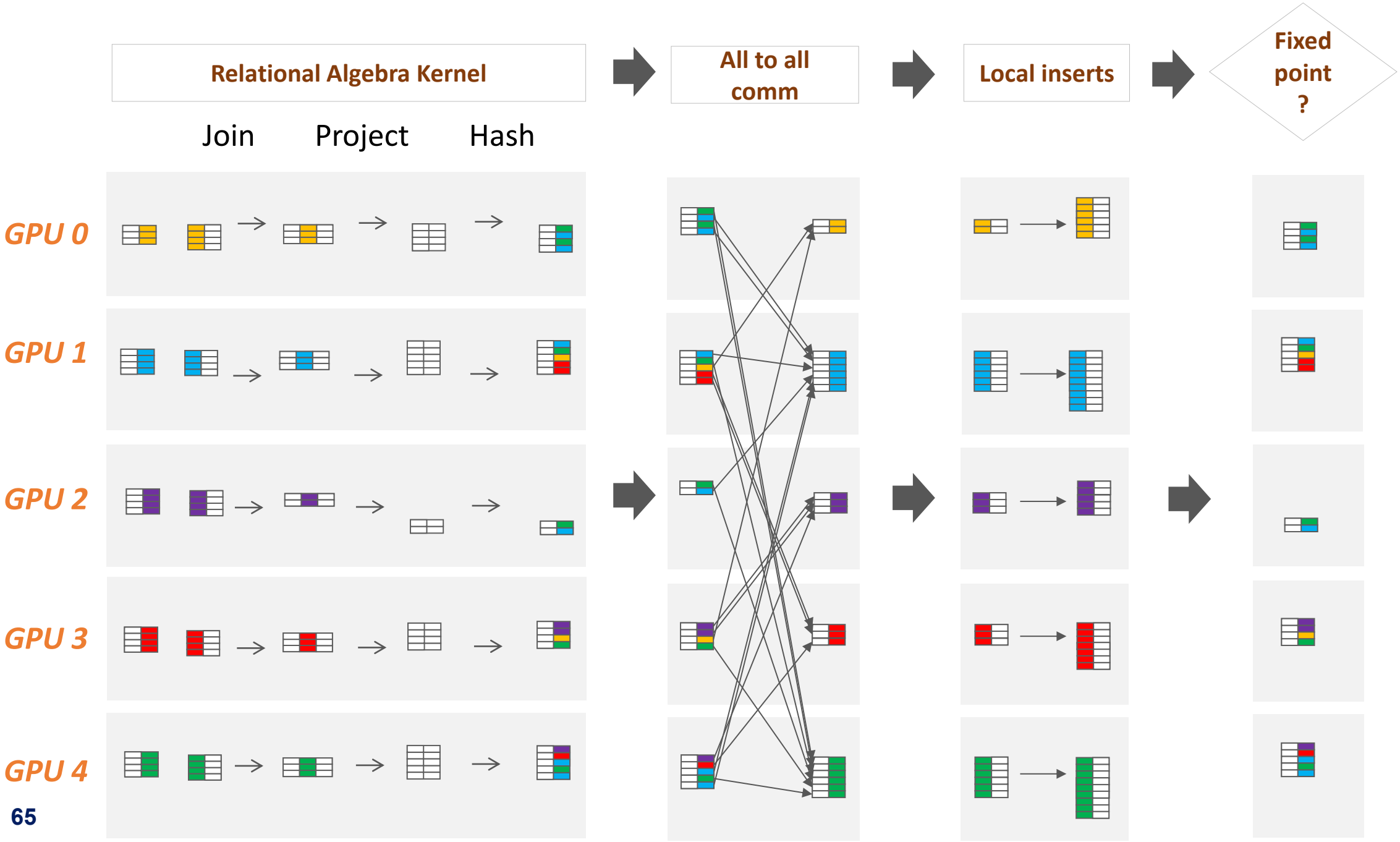
2 Data representation

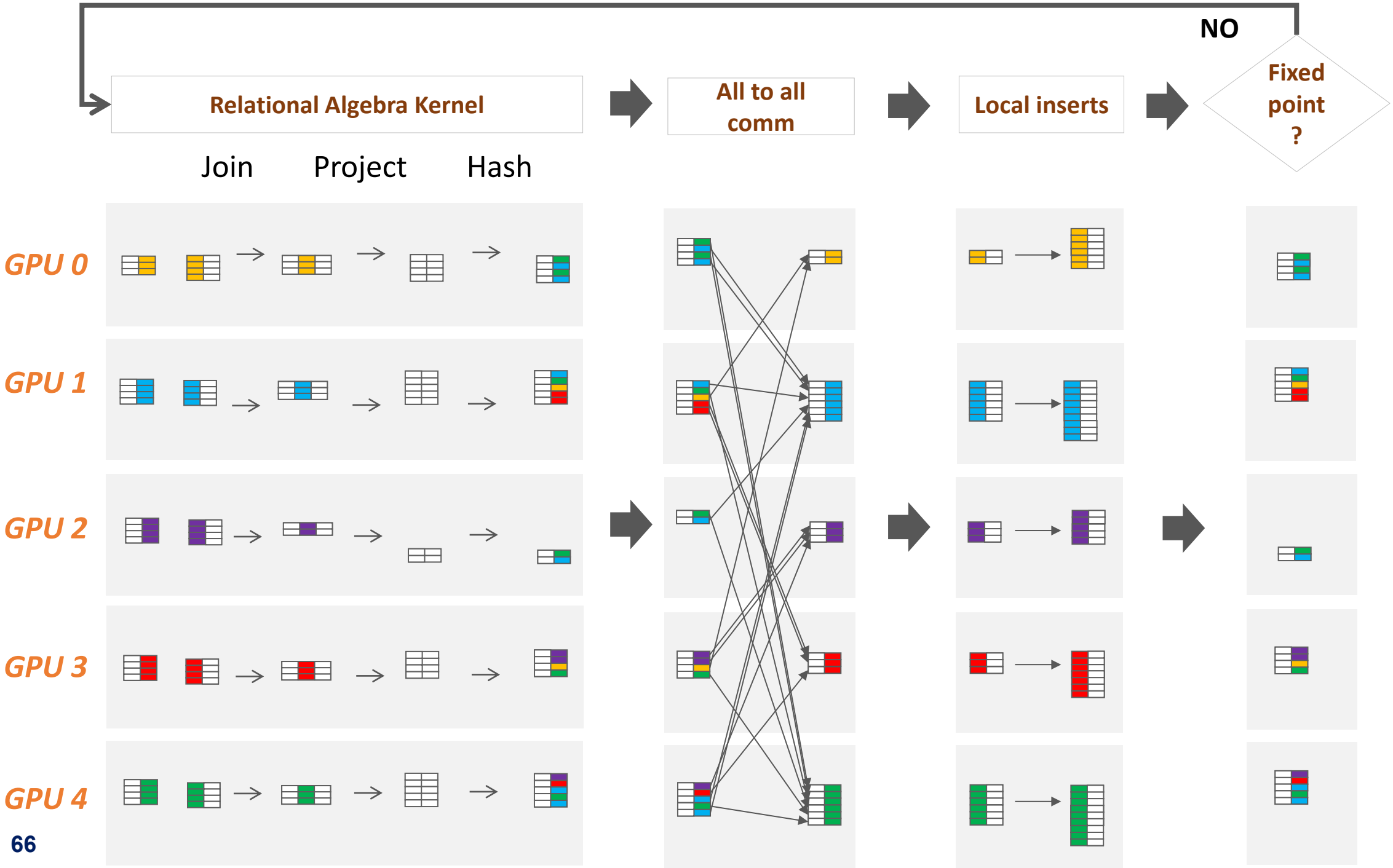
3 Communication

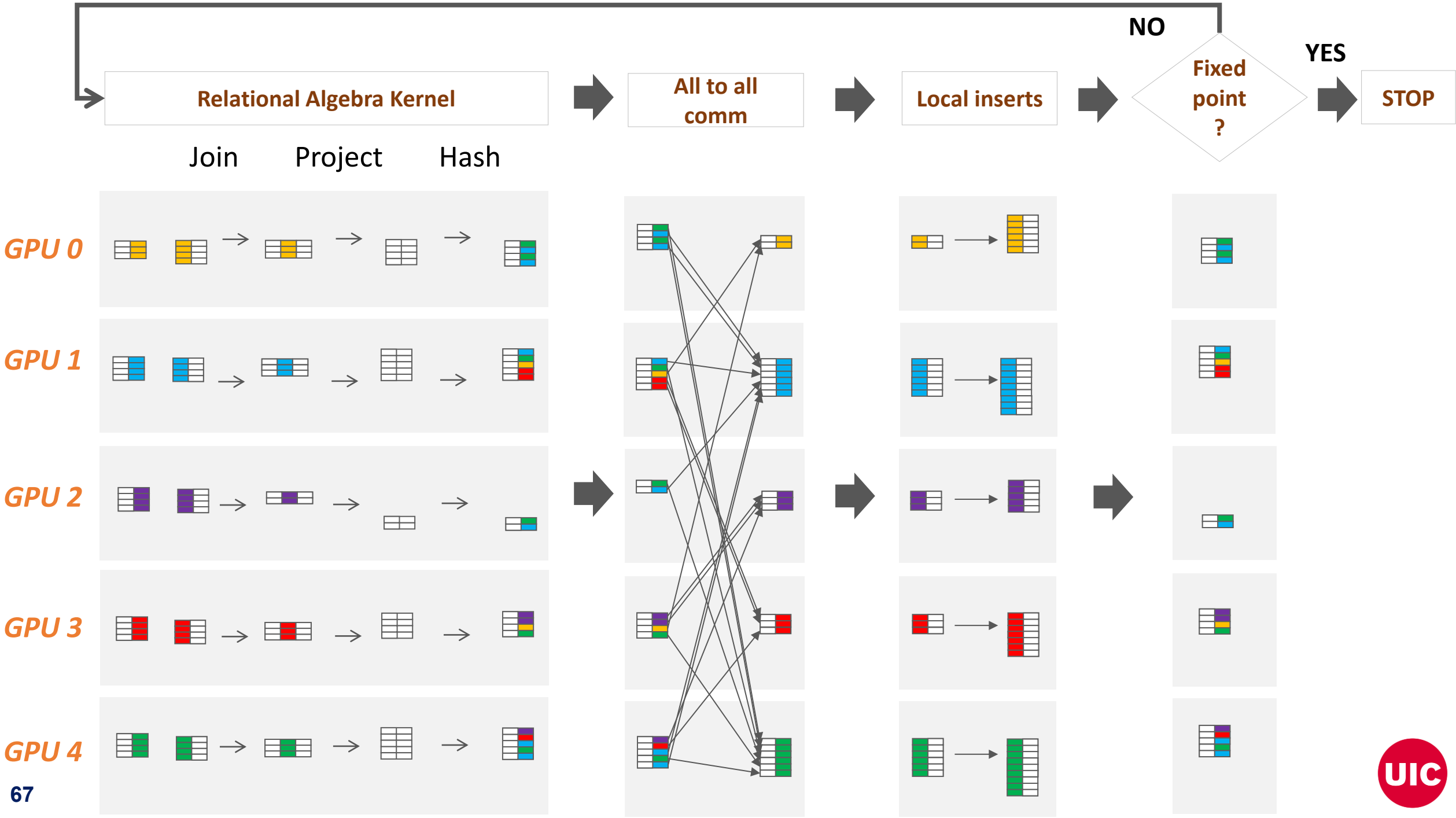


Kumar, S., & Gilray, T. (2020). Load-balancing parallel relational algebra. In *High Performance Computing: 35th International Conference, ISC High Performance 2020, Frankfurt/Main, Germany, June 22–25, 2020, Proceedings 35* (pp. 288-308). Springer International Publishing.


Relational Algebra Kernel
All to all comm
Local inserts
Join
Project
Hash
GPU 0
GPU 1
GPU 2
GPU 3
GPU 4


Relational Algebra Kernel
All to all comm
Local inserts
Fixed point ?
Join
Project
Hash
GPU 0
GPU 1
GPU 2
GPU 3
GPU 4


NO
Relational Algebra Kernel
All to all comm
Local inserts
Fixed point ?
Join
Project
Hash
GPU 0
GPU 1
GPU 2
GPU 3
GPU 4


NO
YES
Relational Algebra Kernel
All to all comm
Local inserts
Fixed point ?
STOP
Join
Project
Hash
GPU 0
GPU 1
GPU 2
GPU 3
GPU 4

MNMGDatalog Evaluation

Polaris (Argonne National Lab)

Multi-threaded	Multi-node Multi-threaded	Single-GPU
Soufflé	SLOG	GPUJoin
		GPULog
		cuDF

- Argonne Leadership Computing Facility. 2022. Polaris. https://www.alcf.anl.gov/polaris
- Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data
- Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. ACM Trans. Math. Softw. 38, 1, Article 1 (dec 2011), 25 pages. doi:10.1145/2049662.2049663

Single GPU benchmark

Dataset name	*TC* edges	Time (s) MNMGDATALOG	GPULOG	Soufflé	GPUJoin
com-dblp	1.91B	**13.58**	26.95	232.99	OOM
fe_ocean	1.67B	**66.34**	72.74	292.15	100.30
usroads	871M	**75.07**	78.08	222.76	364.55
vsp_finan	910M	**81.14**	82.75	239.33	125.94

Transitive closure benchmark on single-GPU

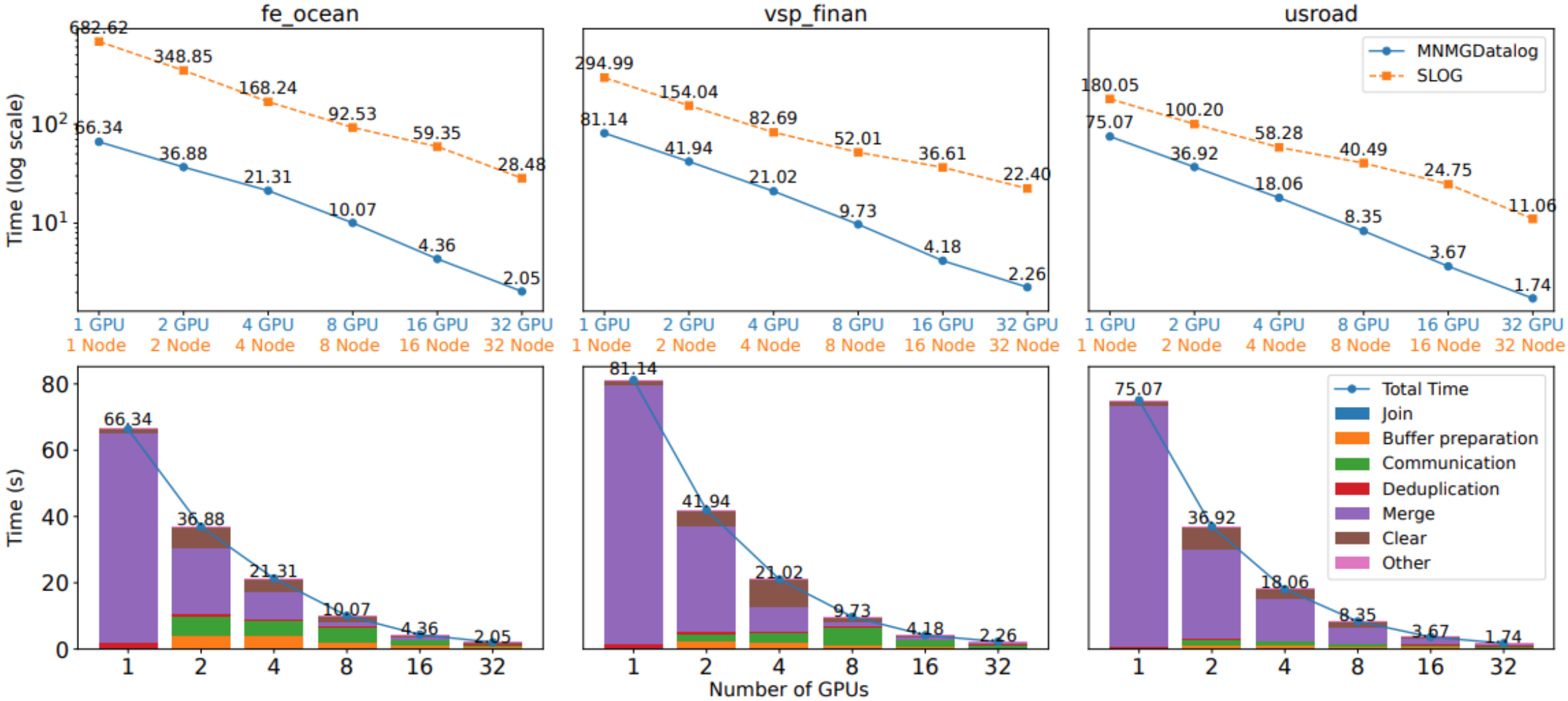
Speedup up to 1.9x over GPULog, 20x over Souffle, 4.8x over GPUJoin

Dataset name	*SG* size	Time (s) MNMGDATALOG	GPULOG	Soufflé	cuDF
fe_body	408M	**9.08**	18.41	74.26	OOM
loc-Brightkite	92.3M	**1.66**	11.67	48.18	OOM
fe_sphere	205M	**3.55**	7.88	48.12	OOM
CA-HepTH	74M	**0.60**	4.79	20.12	21.24

Same generation benchmark on single-GPU

Speedup up to 7x over GPULog, 33.5x over Souffle, 35.4x over cuDF

Multi-node multi-GPU benchmark



Transitive closure benchmark and breakdown up to 32 GPUs

Speedup up to 32x

Roadmap

1. Introduction and motivation
2. GPU based Datalog engine development
3. Multi-node multi-GPU Datalog engine development
4. **GPU portability of Datalog engines**
5. Power analysis of Datalog engines
6. Exploration of high-dimensional scientific analytics
7. Future work
8. Conclusion

Parallel Programming Models

Argonne NATIONAL LABORATORY

W.J. Cody Associates

Rosetta parallel programming models benchmarking framework

- Sequential
- OpenMP
- OpenMP-target
- CUDA
- HIP
- SYCL

Rosetta Benchmark | Code

Sections

- All Results (Table)
- Walltime Plot
- Speedup Plot: openmp-parallel
- Speedup Plot: openmp-task
- Speedup Plot: serial

Benchmark Report

All Results (Table)

Benchmark	PPM	# Samples	Wall	User	Kernel	ompwtime	Max RSS	Peak Allocation
idioms.assign	serial	54	1.57µs	1.63µs	None	N/A	N/A	78.12KiB
idioms.assign	openmp-parallel	37925	50.58µs	2.14ms	45.35µs	50.43µs	22.17MiB	78.12KiB
idioms.assign	openmp-task	11644	757.55µs	42.62ms	192.99µs	757.34µs	17.44MiB	78.12KiB
idioms.pointwise	serial	103	1.59µs	1.57µs	None	N/A	N/A	78.12KiB
idioms.pointwise	openmp-parallel	38113	56.17µs	2.48ms	55.12µs	56.35µs	22.23MiB	78.12KiB
idioms.pointwise	openmp-task	10837	786.66µs	45.08ms	176.15µs	786.75µs	14.20MiB	78.12KiB
idioms.reduction	serial	9	8.29µs	8.22µs	None	N/A	N/A	8.00B
idioms.reduction	openmp-parallel	32145	68.30µs	2.93ms	53.20µs	68.10µs	17.71MiB	8.00B
idioms.reduction	openmp-task	11618	760.66µs	43.64ms	172.65µs	760.60µs	17.00MiB	8.00B
suites.polybench.2mm	serial	4	7.26ms	7.25ms	7.00µs	N/A	N/A	1.19MiB
suites.polybench.2mm	openmp-parallel	13124	638.60µs	37.28ms	187.96µs	638.60µs	13.33MiB	1.19MiB
suites.polybench.3mm	serial	4	11.72ms	11.72ms	None	N/A	N/A	1.72MiB
suites.polybench.3mm	openmp-parallel	15288	503.06µs	27.71ms	265.82µs	503.43µs	15.01MiB	1.72MiB
suites.polybench.adi	serial	4	71.42ms	71.42ms	None	N/A	N/A	1.22MiB
suites.polybench.adi	openmp-parallel	566	17.36ms	1.05	5.26ms	17.36ms	6.20MiB	1.22MiB
suites.polybench.atax	serial	4	148.06µs	142.25µs	5.75µs	N/A	N/A	1.16MiB
suites.polybench.atax	openmp-parallel	69372	56.16µs	2.92ms	40.66µs	56.45µs	36.61MiB	1.16MiB

- Rosetta-bench Github repository. https://github.com/Meinersbur/rosetta

GPULog-HIP

- Ported GPULog to GPULog-HIP
- AMD MI250 (dual chiplet) ≈ Nvidia A100
- GPULog-HIP uses only one MI250 chiplet
- GPULog benefits from Nvidia RMM library

		GPUlog (NVIDIA)		GPUlog (AMD)	
Query	Dataset	H100	A100	MI250	MI50
SG	fe_body	5.05	8.61	19.57	41.99
	BrightKite	3.42	6.79	14.00	30.05
	fe_sphere	2.36	4.64	8.48	19.426
CSPA	httpd	1.33	2.73	6.75	15.27
	linux	0.39	0.77	1.39	3.32
	postgres	1.27	2.68	6.79	14.55

GPULog and GPULog-HIP comparison on Nvidia and AMD GPUs

- GPULog-HIP Github repository. https://github.com/harp-lab/gdlog-hip

GPUJoin-SYCL

- Ported GPUJoin to SYCL using SYCLomatic
- Required bug fixing of the generated code
- Scratch implementation of TC using SYCL
- Won Intel® Arc A770 Limited Edition GPU

Clean	Clean CUDA Project
Install	Install SYCLomatic
Convert	Convert CUDA code to SYCL
Check	Check the SYCL code and modify if necessary
Execute	Run in Intel Dev Cloud

CUDA-based GPUJoin to SYCL using SYCLomatic

- oneAPI Hackathon: CUDA to SYCL Migration https://oneapi-sycl-hackathon.devpost.com/
- Github repository: https://github.com/arsho/tc

Roadmap

1. Introduction and motivation
2. GPU based Datalog engine development
3. Multi-node multi-GPU Datalog engine development
4. GPU portability of Datalog engines
5. **Power analysis of Datalog engines**
6. Exploration of high-dimensional scientific analytics
7. Future work
8. Conclusion

Compare energy profiles across Datalog engines

Engine	Join strategy	Data structure
MNMGDatalog	Radix hash join	GPU Hash table
INLJoin	Index nested loop join	GPU struct array
GPULog	Sorted hash join	Hash-Indexed Sorted Array
BJoin	Batch join with memory pooling	Hash-Indexed Sorted Array
cuDF	Dataframe's merge	Python's Dataframe

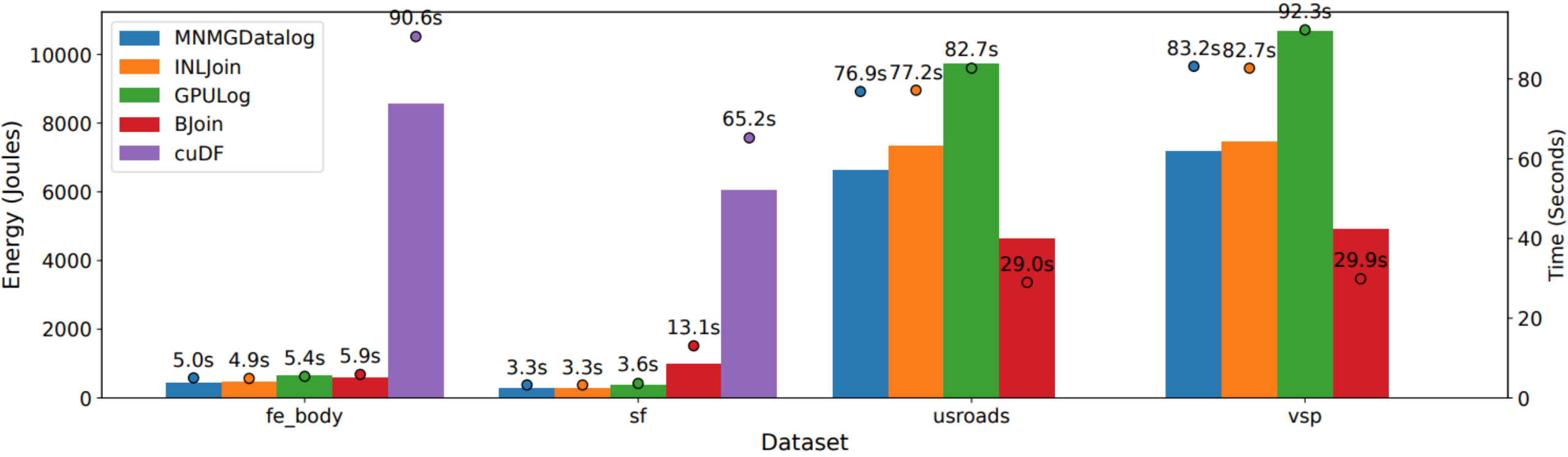
Polaris (Argonne National Lab)

Tools: nvidia-smi, nvml, NVIDIA Nsight systems, NVIDIA Nsight compute

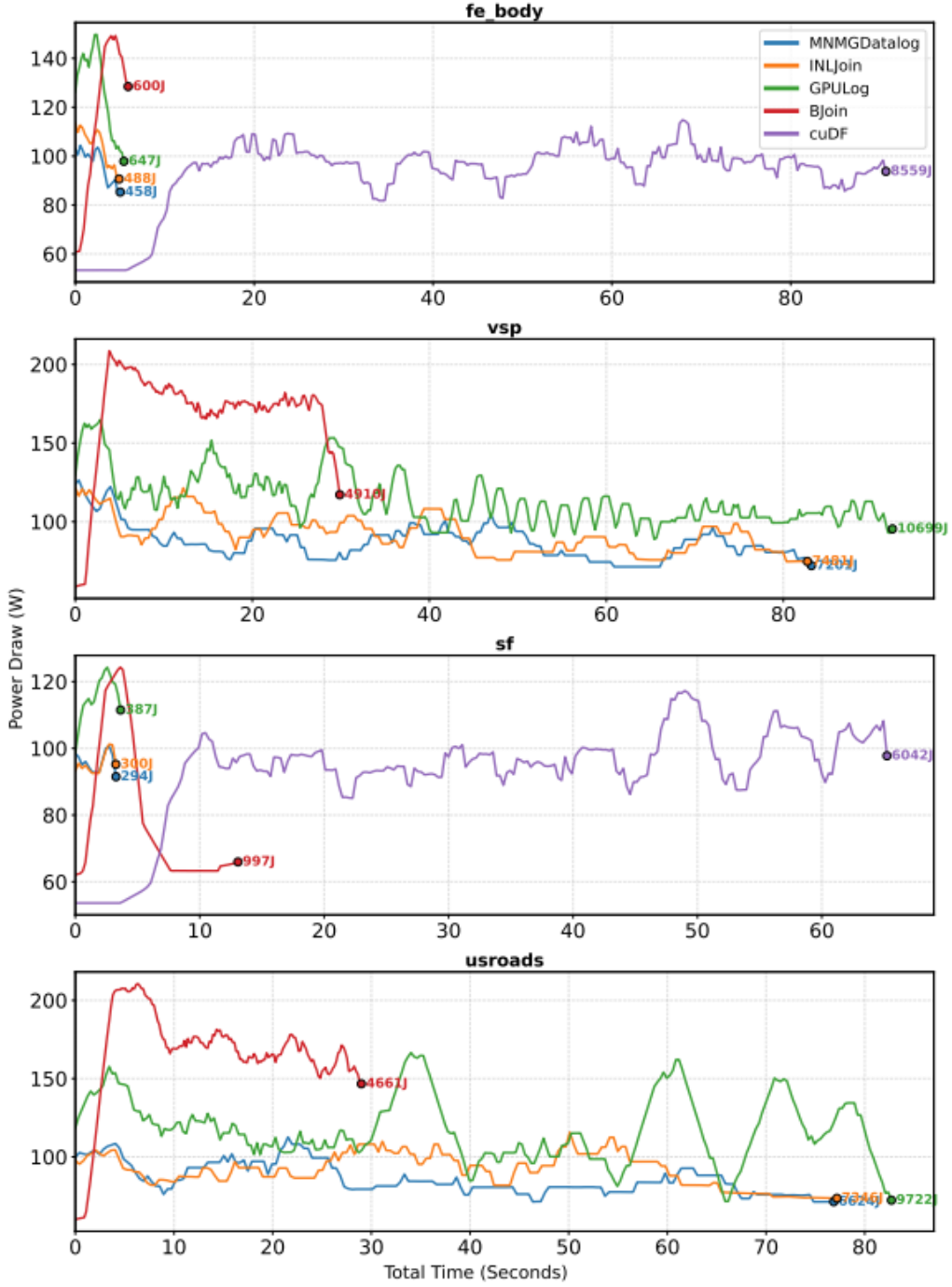
Energy profile: TC energy consumption (single-GPU)

$$E = \sum_{i=1}^{N} P_i \cdot \Delta t_i$$

- N is the total number of sampling intervals
- P_i is the power draw (in watts) measured at interval i
- Δt_i is the time (in seconds) between samples i and i - 1

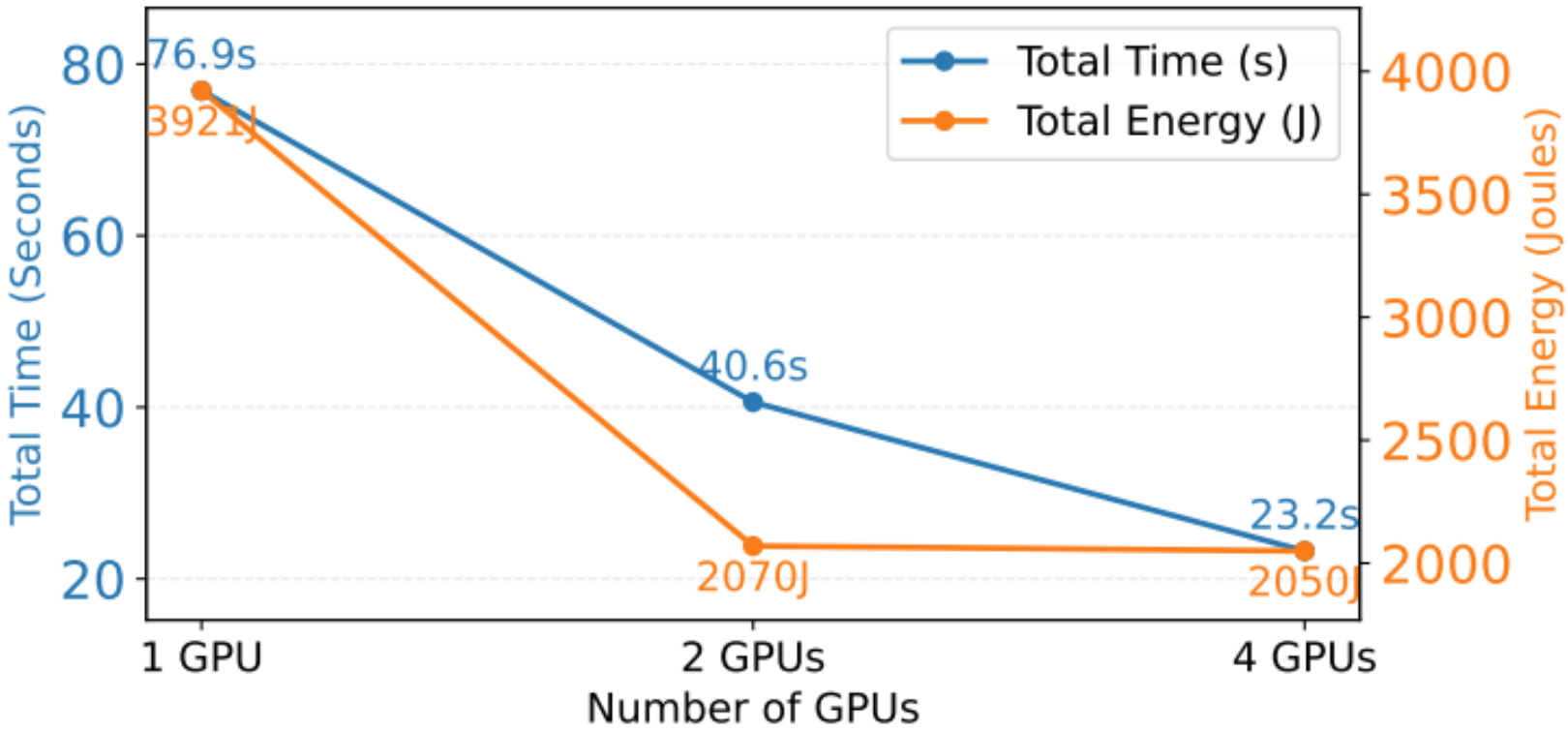


- BJoin's memory pooling draws more power, runs faster for larger iterations
- HISA-based joins (BJoin, GPULog) use more energy
- Simpler GPU friendly data structure (MNMGDatalog and INLJoin) draws less power but could run longer



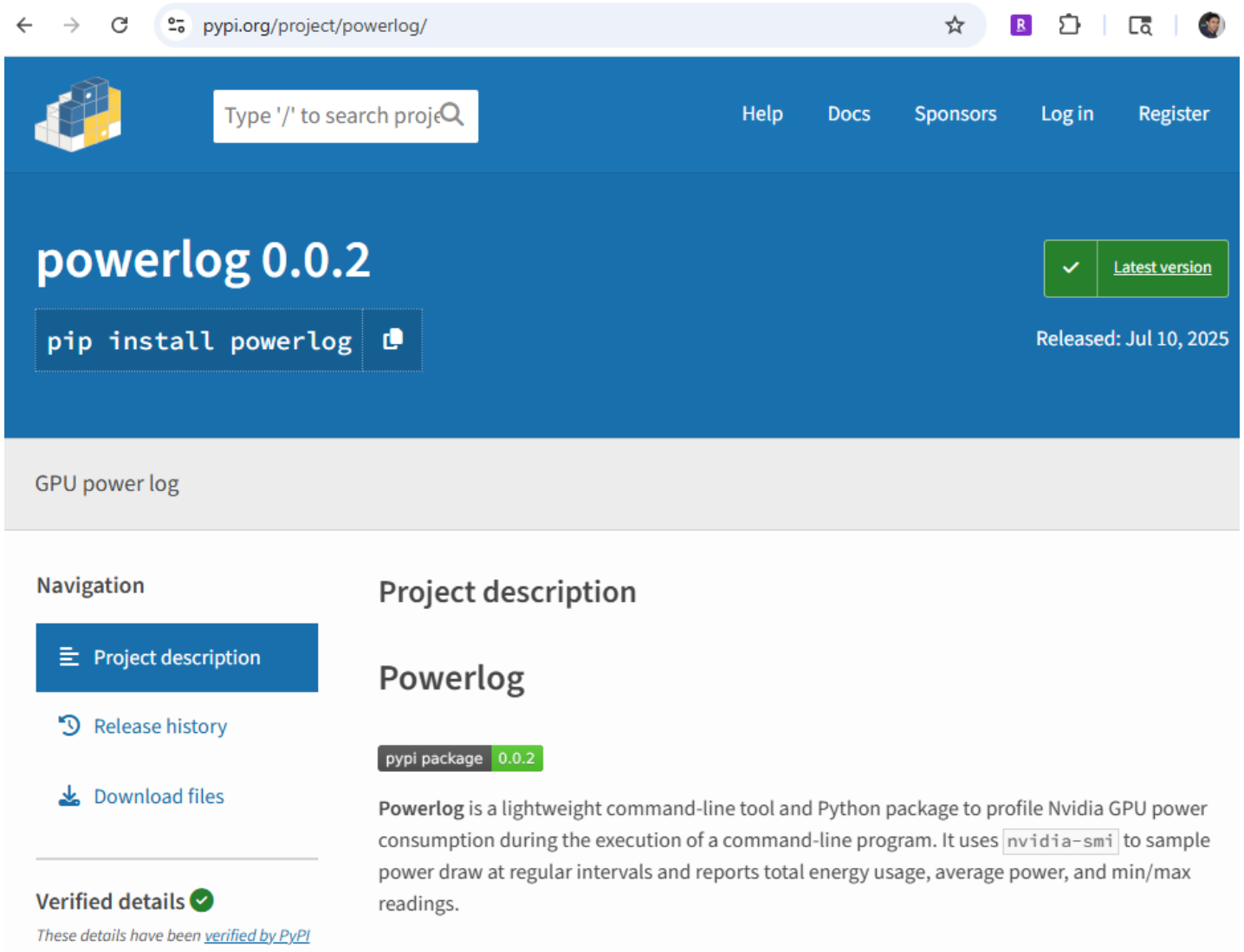
Power draw (TC)

Energy profile: TC energy consumption (multi-GPUs)



TC energy consumption (1 - 4 GPUs) for usroads dataset using MNMGDatalog

Powerlog CLI Tool for Energy Profiling GPU Applications



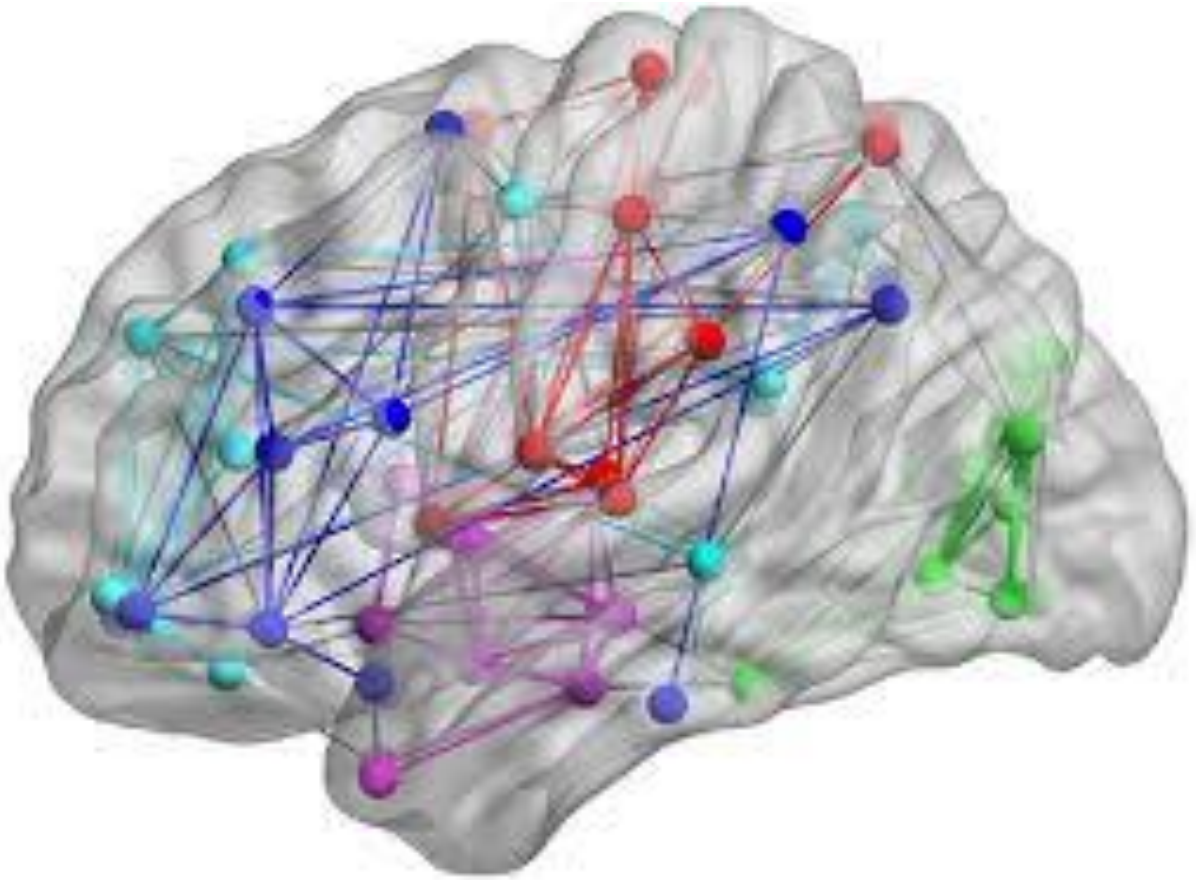
Powerlog Python package (https://pypi.org/project/powerlog/)

Roadmap

1. Introduction and motivation
2. GPU based Datalog engine development
3. Multi-node multi-GPU Datalog engine development
4. GPU portability of Datalog engines
5. Power analysis of Datalog engines
6. **Exploration of high-dimensional scientific analytics**
7. Future work
8. Conclusion

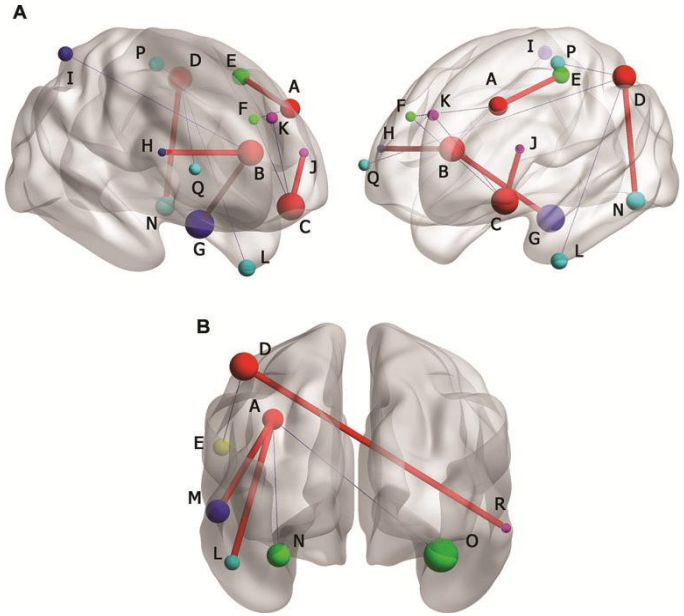
UIC

What is a brain network?

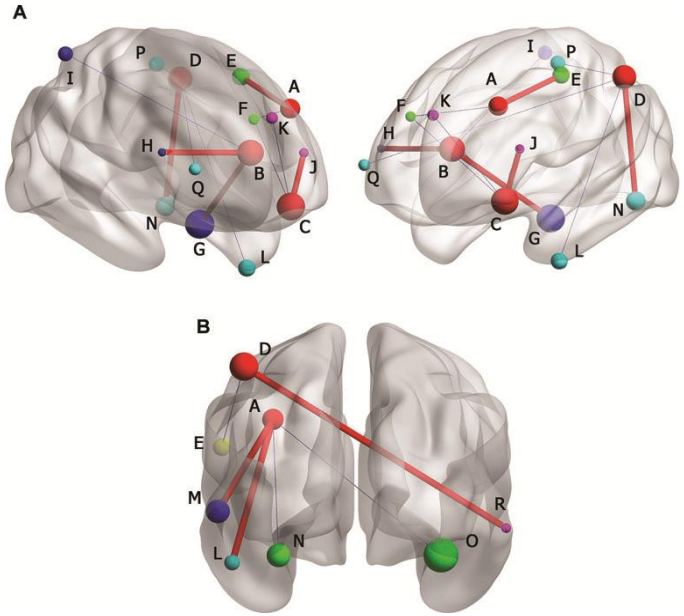


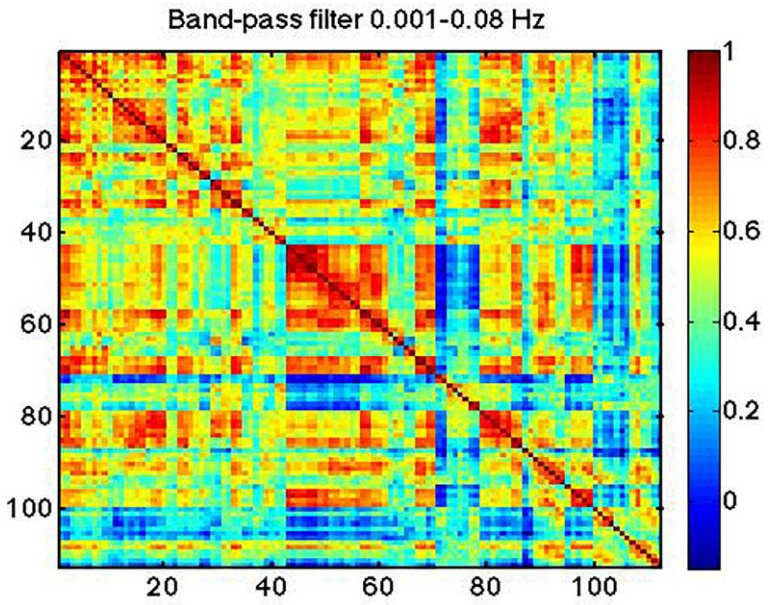
How to construct brain network?

How to construct brain network?



How to construct brain network?





Multi-site brain image analysis

UAB THE UNIVERSITY OF ALABAMA AT BIRMINGHAM

THE UNIVERSITY OF ALABAMA

AUBURN UNIVERSITY

USA UNIVERSITY OF SOUTH ALABAMA

Multi-site rs-fMRI studies allow gathering larger sample sizes

- B. B. Biswal, M. Mennes, X.-N. Zuo, S. Gohel, C. Kelly, S. M. Smith, C. F. Beckmann, J. S. Adelstein, R. L. Buckner, S. Colcombe et al., "Toward discovery science of human brain function," Proceedings of the national academy of sciences, vol. 107, no. 10, pp. 4734–4739, 2010.

UIC

Multi-site brain image analysis (continue)

UAB THE UNIVERSITY OF ALABAMA AT BIRMINGHAM

THE UNIVERSITY OF ALABAMA

AUBURN UNIVERSITY

USA UNIVERSITY OF SOUTH ALABAMA

But different **scanners** and **sampling periods** negates the advantages of larger sample sizes

• C. Dansereau, Y. Benhajali, C. Risterucci, E. M. Pich, P. Orban, D. Arnold, and P. Bellec, "Statistical power and prediction accuracy in multisite resting-state fmri connectivity," Neuroimage, vol. 149, pp. 220–232, 2017.

UIC

Convert fMRI image to FCN

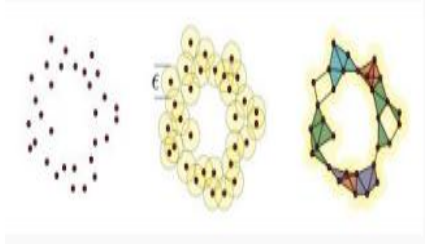
Apply Topological Data Analysis on FCN

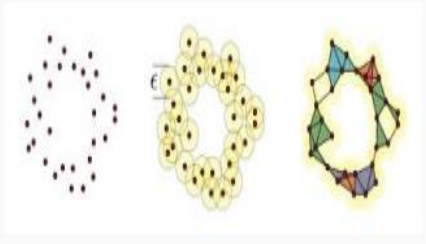
UAB
THE UNIVERSITY OF ALABAMA AT BIRMINGHAM

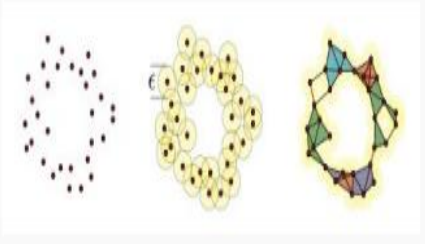
THE UNIVERSITY OF ALABAMA

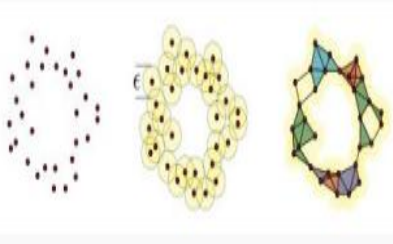
AUBURN UNIVERSITY

USA UNIVERSITY OF SOUTH ALABAMA









Topological Data Analysis (TDA)

Studies shape of data like connected components, loops, voids

Main tool: persistent homology, adapts homology to point cloud

Analyzes social networks, brain connectivity, internet

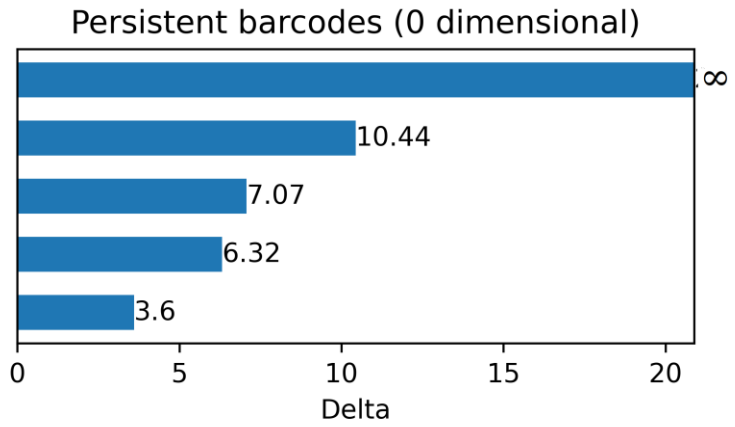
- M. Rubinov and O. Sporns, “Complex network measures of brain connectivity: uses and interpretations,” Neuroimage, vol. 52, no. 3, pp. 1059–1069, 2010.
- Aktas, M.E., Akbas, E. & Fatmaoui, A.E. Persistence homology of networks: methods and applications. Appl Netw Sci 4, 61 (2019). https://doi.org/10.1007/s41109-019-0179-3

UIC

Persistent Barcodes using Connected Component

Persistent Homology tracks birth and death of topological features in the graph

0-dim barcodes: each bar shows when each features (connected component) born and die

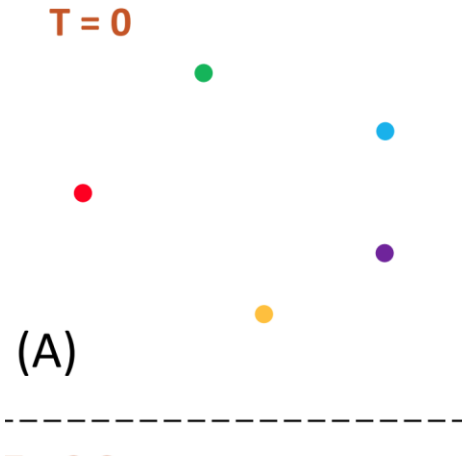


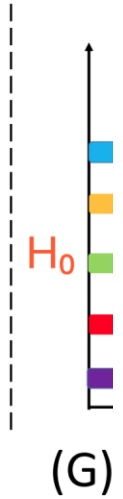
Adjacency matrix

0.00	8.54	12.36	7.07	10.44
8.54	0.00	20.88	3.60	18.00
12.36	20.88	0.00	18.78	6.32
7.07	3.60	18.78	0.00	15.13
10.44	18.00	6.32	15.13	0.00

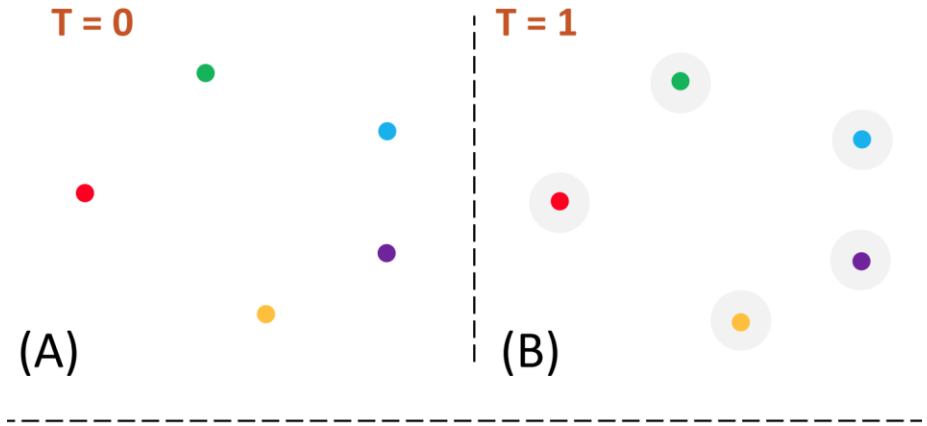
• Kumar, S., Shovon, A. R., & Deshpande, G. (2023). The robustness of persistent homology of brain networks to data acquisition-related non-neural variability in resting state fMRI. Human Brain Mapping, 44(13), 4637-4651.

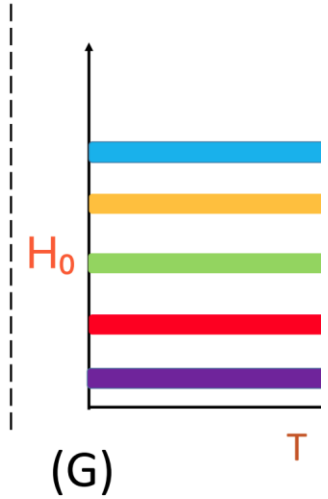
Computing Persistent Homology (Point Cloud)



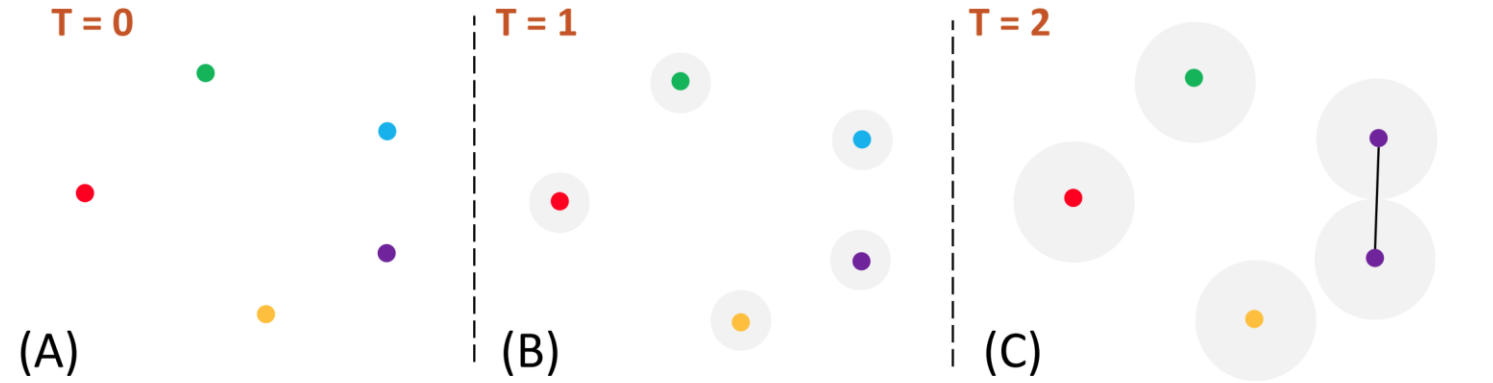


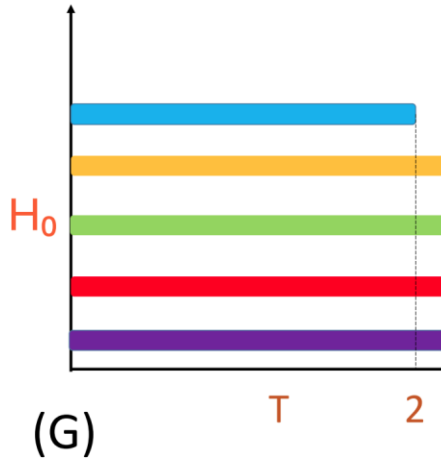
Computing Persistent Homology (Point Cloud)



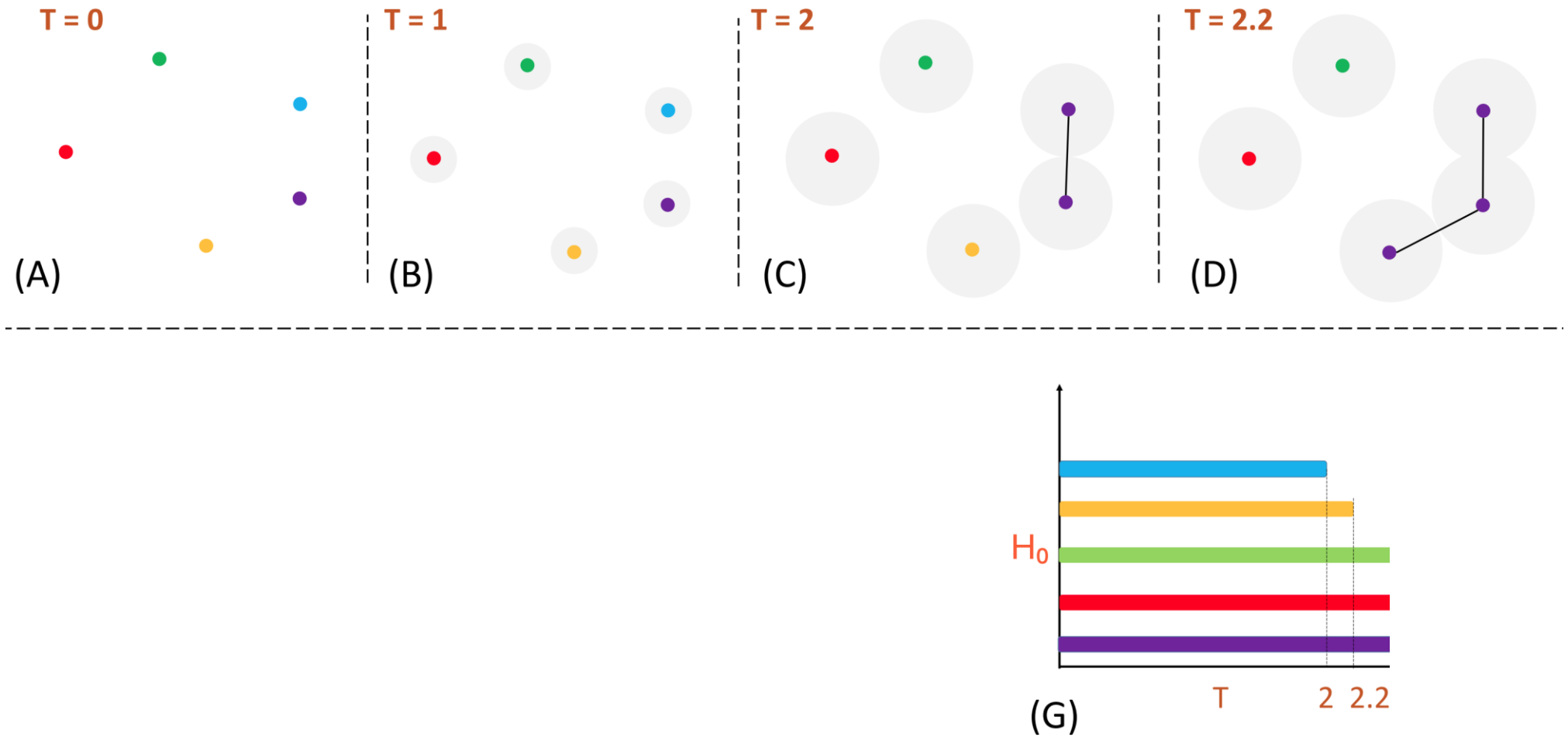


Computing Persistent Homology (Point Cloud)

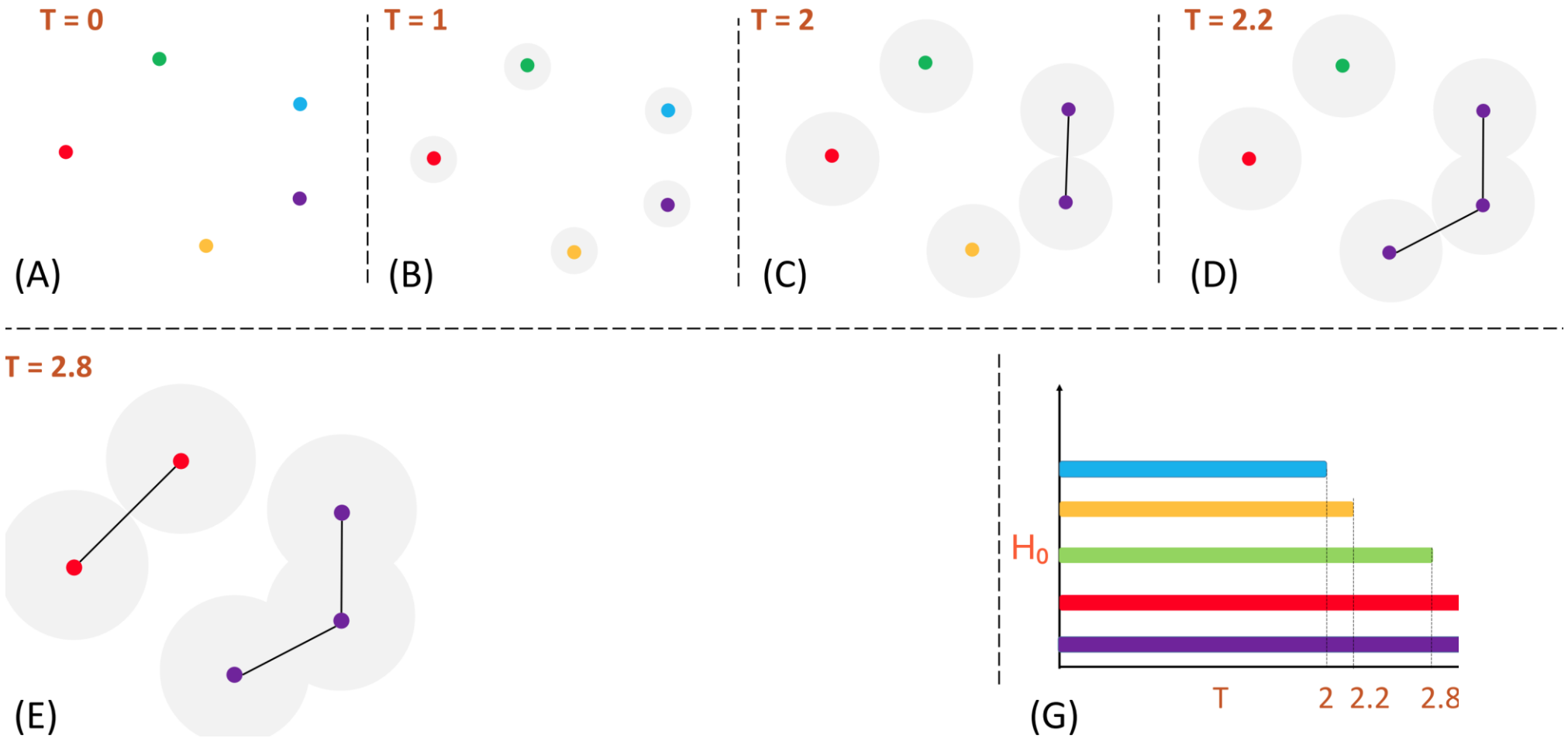




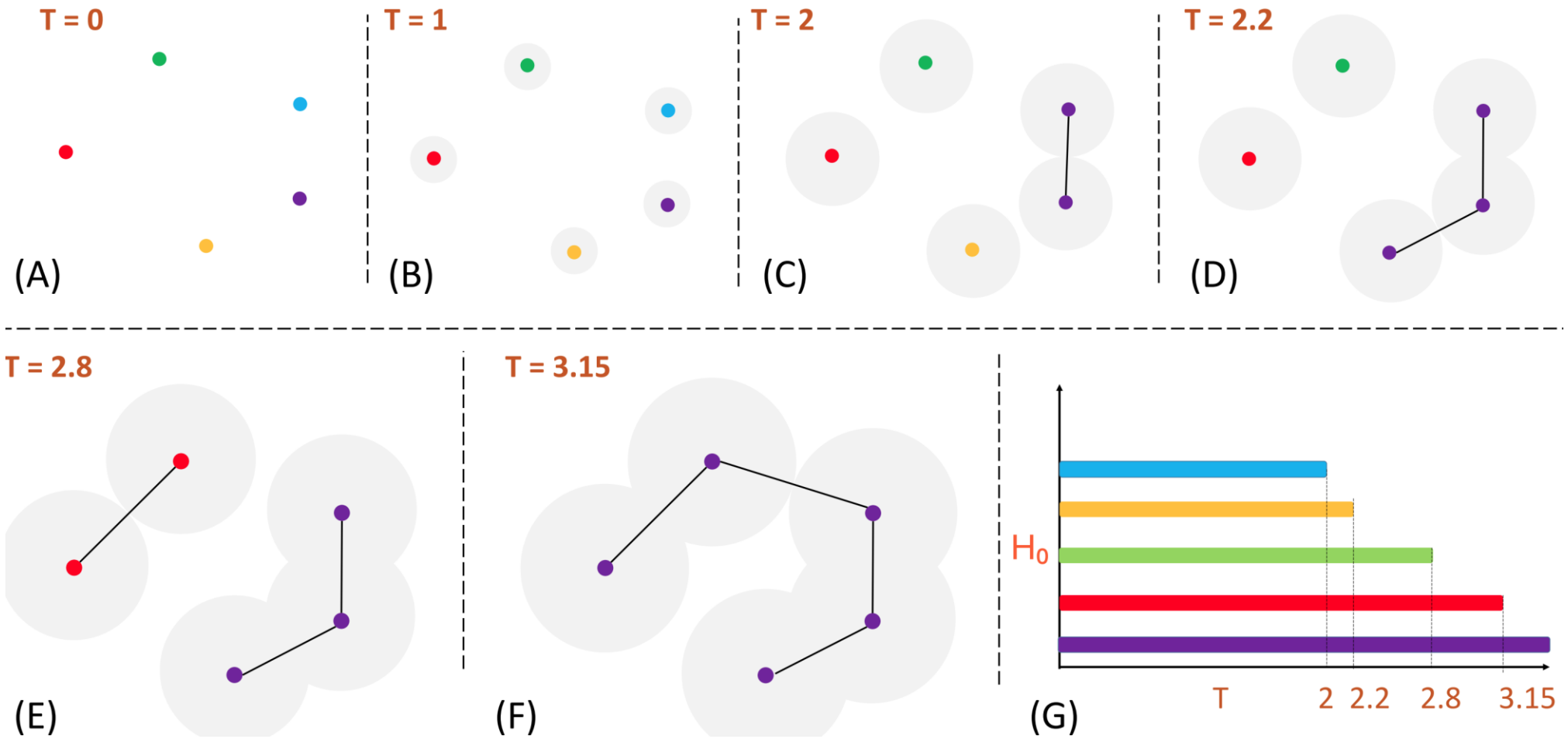
Computing Persistent Homology (Point Cloud)



Computing Persistent Homology (Point Cloud)



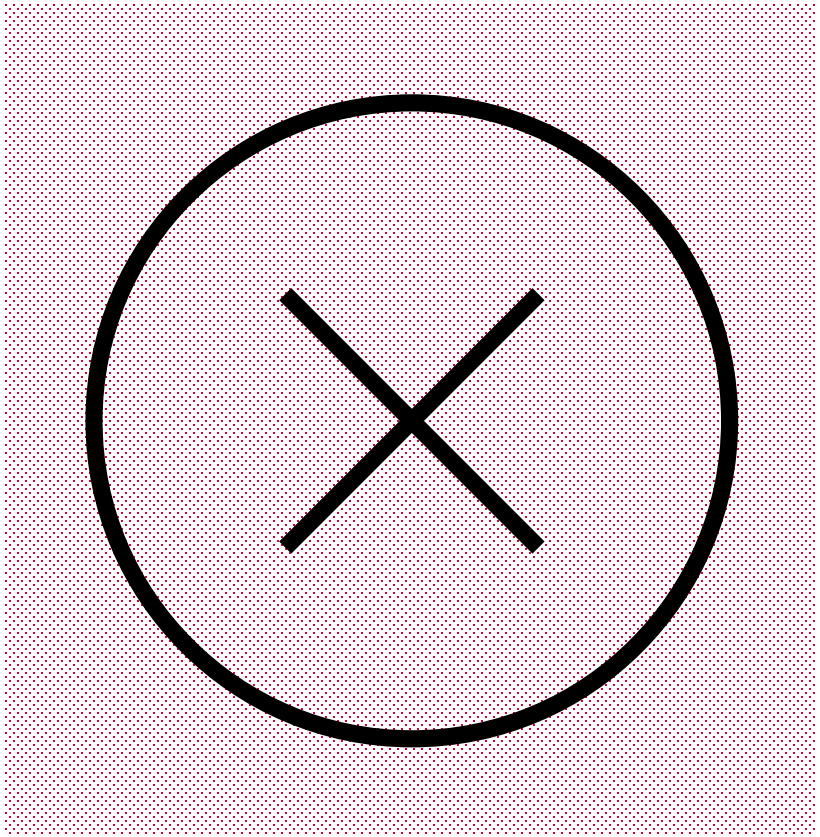
Computing Persistent Homology (Point Cloud)



PH for High-Dimensional Data Analysis

Verify	PH of rs-fMRI can represent topological features of FCNs
Demonstrate	FCN networks are statistically similar despite variations in TRs
Remove	Non-neural variability in multi-site brain networks

- Ashley Suh, Mustafa Hajij, Bei Wang, Carlos Scheidegger, and Paul Rosen. Persistent homologyguided force-directed graph layouts. IEEE Transactions on Visualization and Computer Graphics,26(1):697–707, January 2020
- Tananun Songdechakraiwut, Li Shen, and Moo Chung. Topological learning and its application tomultimodal brain network integration. In Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021,Proceedings, Part II 24, pages 166–176. Springer, 2021.



Limitations

Imperative and complex high-dimensional data analysis pipelines

Declarative Topological Data Analysis

1

Use Datalog to simplify persistent barcodes computations

2

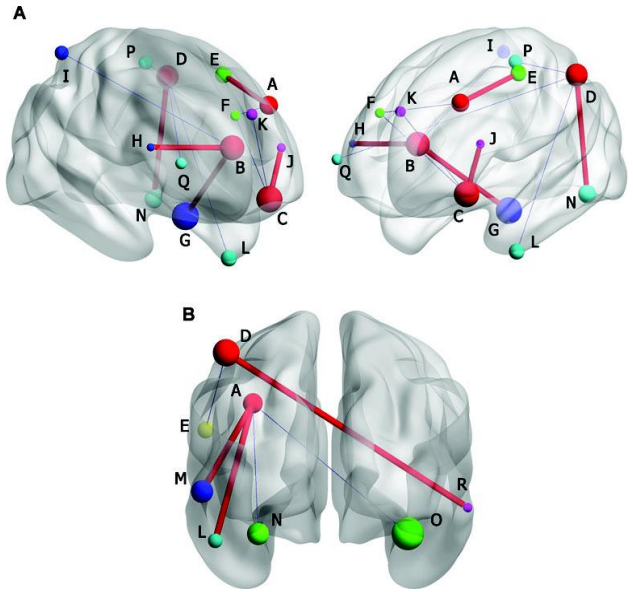
Explore potential applications of using declarative TDA

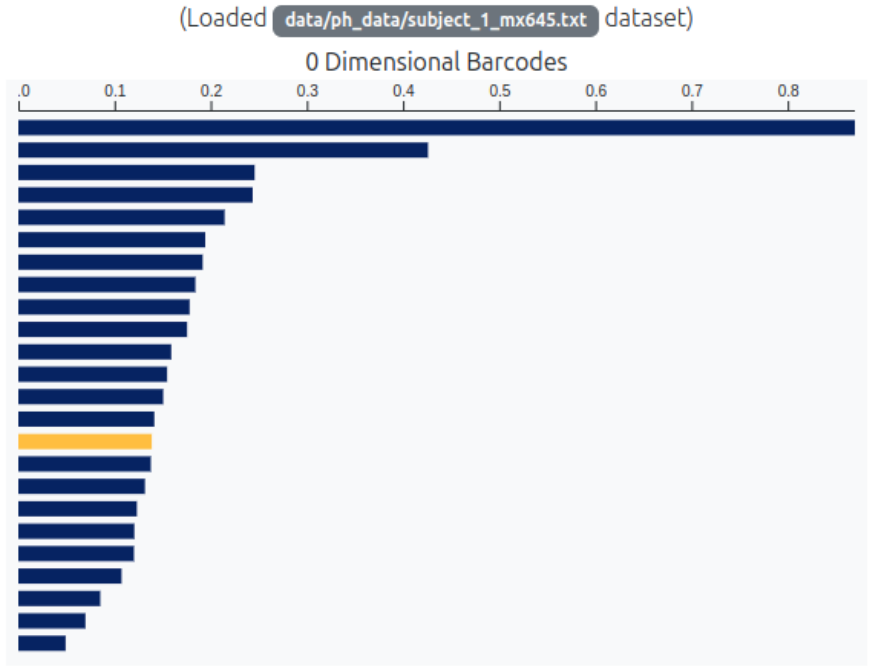
3

Scale TDA methods to large datasets in HPC environment

UIC

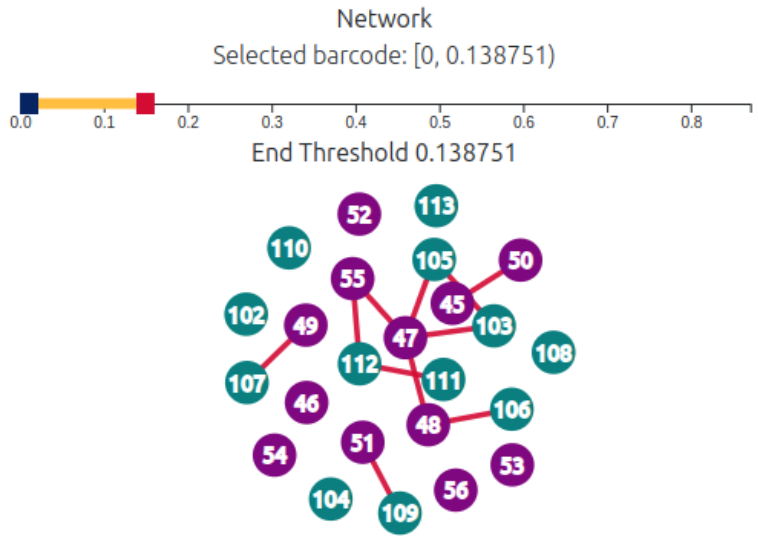
Persistent Barcodes using Declarative Analytics





Construct a filtration

Compute persistence of connected components at each threshold

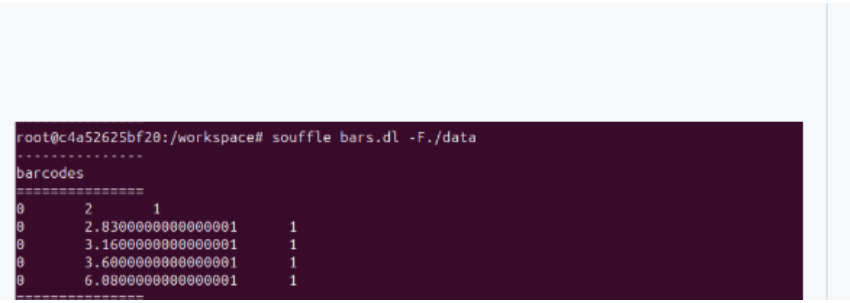


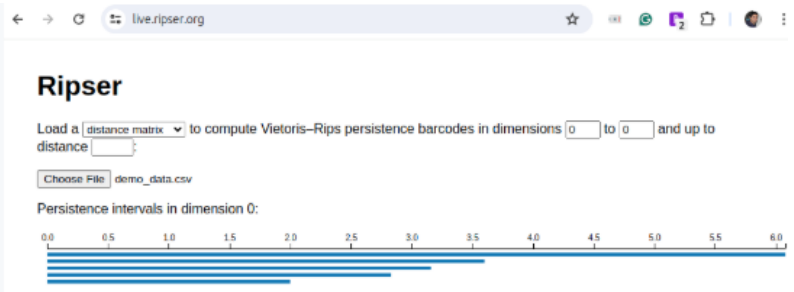
Build graph based on the selected threshold

Web URL: https://harp-lab.github.io/TDA/

Persistent Barcodes using Declarative Analytics

- Use Datalog for efficient connected components calculation
- Successfully implemented 0-dimensional barcodes using Soufflé
- Extend barcodes computation to the **SLOG** engine for scalable multi-node HPC environments





• Thomas Gilray, Arash Sahebolamri, Yihao Sun, Sowmith Kunapaneni, Sidharth Kumar, and Kristopher Micinski. 2024. Datalog with First-Class Facts. Proc. VLDB Endow. 18, 3 (November 2024), 651–665.

Roadmap

1. Introduction and motivation
2. GPU based Datalog engine development
3. Multi-node multi-GPU Datalog engine development
4. GPU portability of Datalog engines
5. Power analysis of Datalog engines
6. Exploration of high-dimensional scientific analytics
7. **Future work**
8. Conclusion

Future/ongoing work

Enhance MNMGDatalog implementation

Extend declarative applications

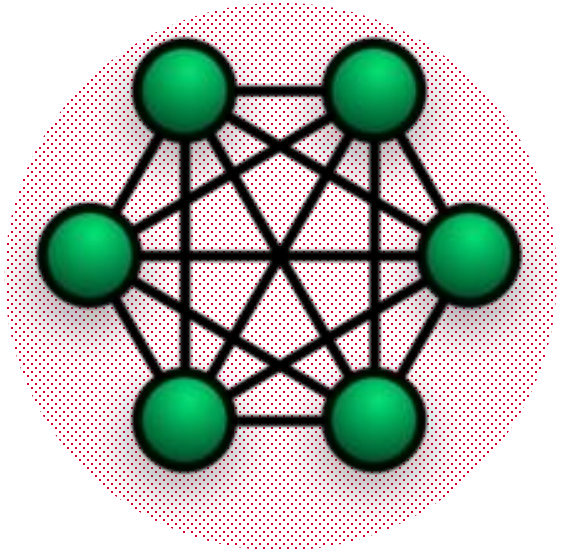
Balance energy and performance

Enhance MNMGDatalog Implementation

Add per-iteration checkpoint/restart capability

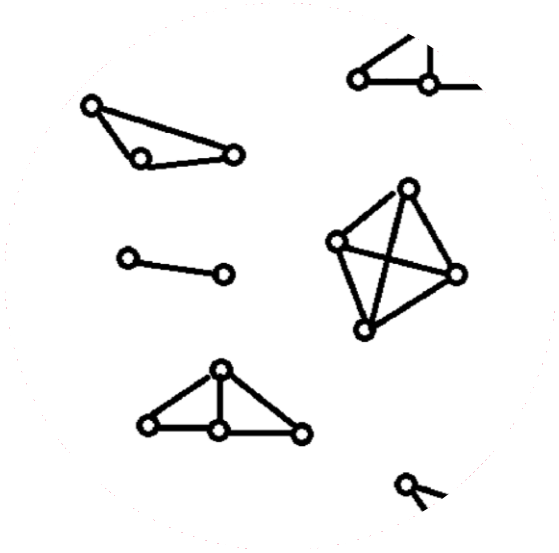
Develop HIP and SYCL version for MNMGDatalog

MNMGDatalog for High-dimensional Brain Imaging



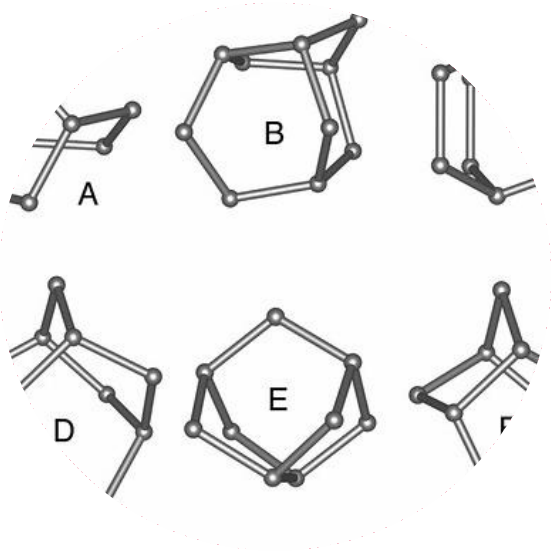
MNMGDatalog for fully connected network

Use high-level expressive query to handle dense connectivity graphs from brain imaging data



Declarative modeling of topological structure

Encode high-dimensional relationships using recursive rules over pairwise similarities

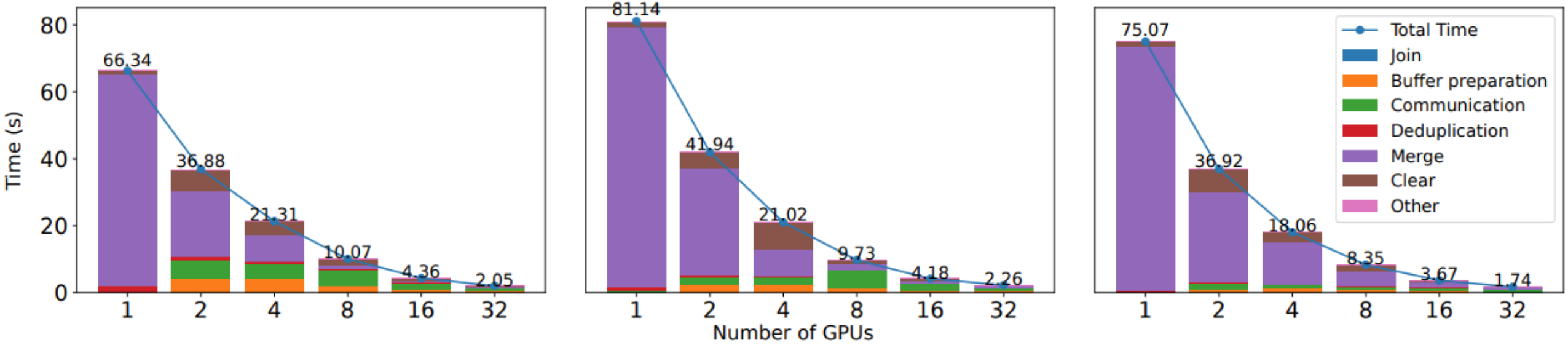


Topological pattern matching

Discover persistent structural motifs across subjects using distributed recursive evaluation

Power-aware Declarative Analytics

Compare energy profiles across Datalog engines



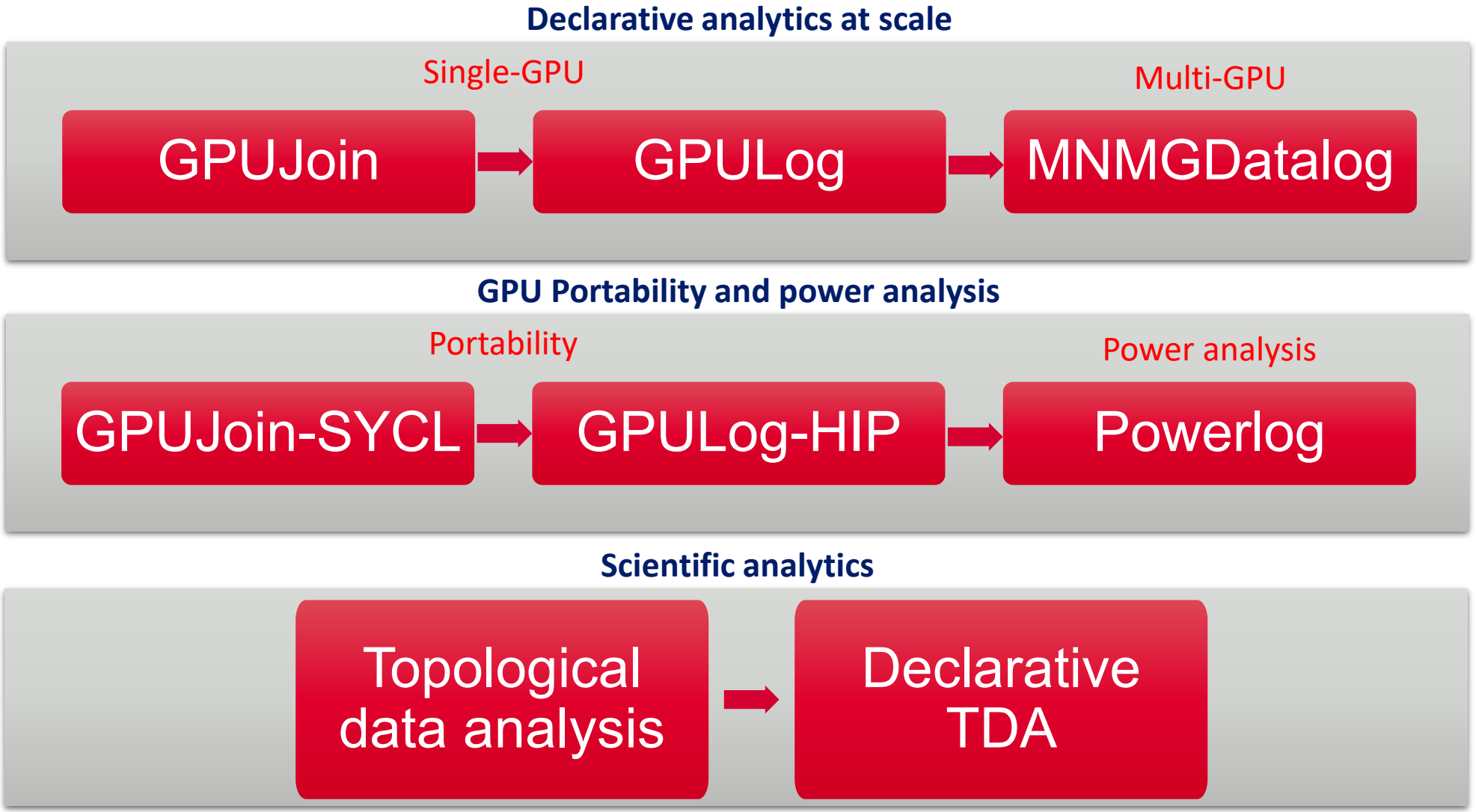
Design runtime tools to correlate iterative operations with power draw

Develop AI-driven models to optimize query plans for energy efficiency

Roadmap

1. Introduction and motivation
2. GPU based Datalog engine development
3. Multi-node multi-GPU Datalog engine development
4. GPU portability of Datalog engines
5. Power analysis of Datalog engines
6. Exploration of high-dimensional scientific analytics
7. Future work
8. **Conclusion**

Conclusion


Declarative analytics at scale
Single-GPU
Multi-GPU
GPUJoin
GPULog
MNMGDatalog
GPU Portability and power analysis
Portability
Power analysis
GPUJoin-SYCL
GPULog-HIP
Powerlog
Scientific analytics
Topological data analysis
Declarative TDA

Acknowledgements

The HARP Lab

evl electronic visualization laboratory

Argonne NATIONAL LABORATORY

UIC UNIVERSITY OF ILLINOIS CHICAGO

AUBURN UNIVERSITY

WASHINGTON STATE UNIVERSITY

UAB The University of Alabama at Birmingham

Syracuse University

NSF

Thank You

ashov@uic.edu
https://arshovon.com/

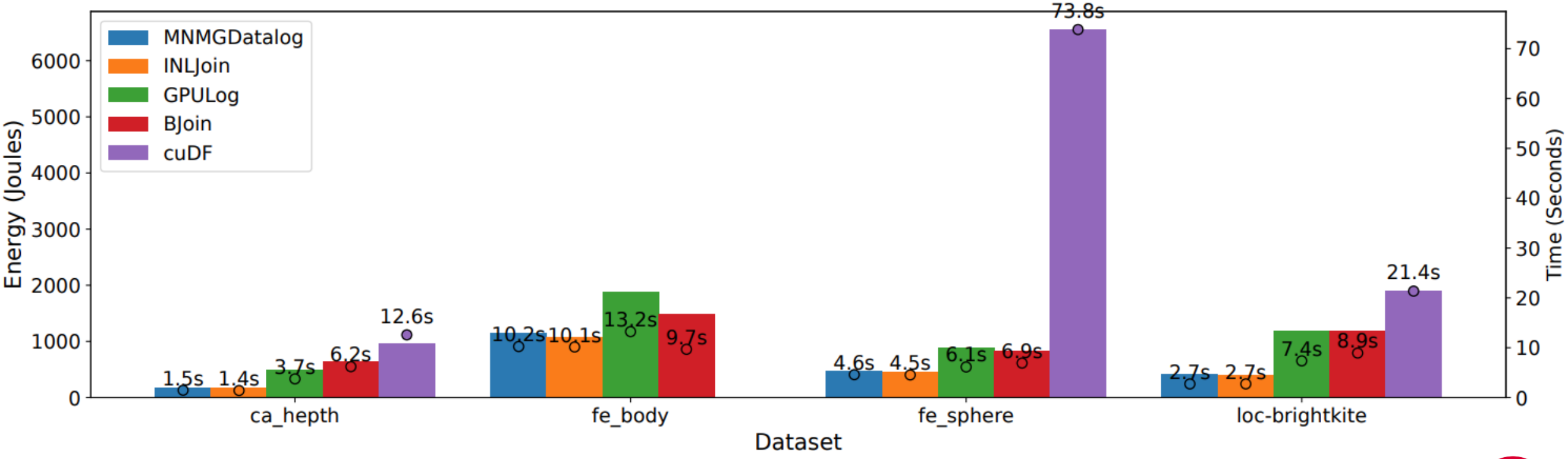
UIC UNIVERSITY OF ILLINOIS CHICAGO

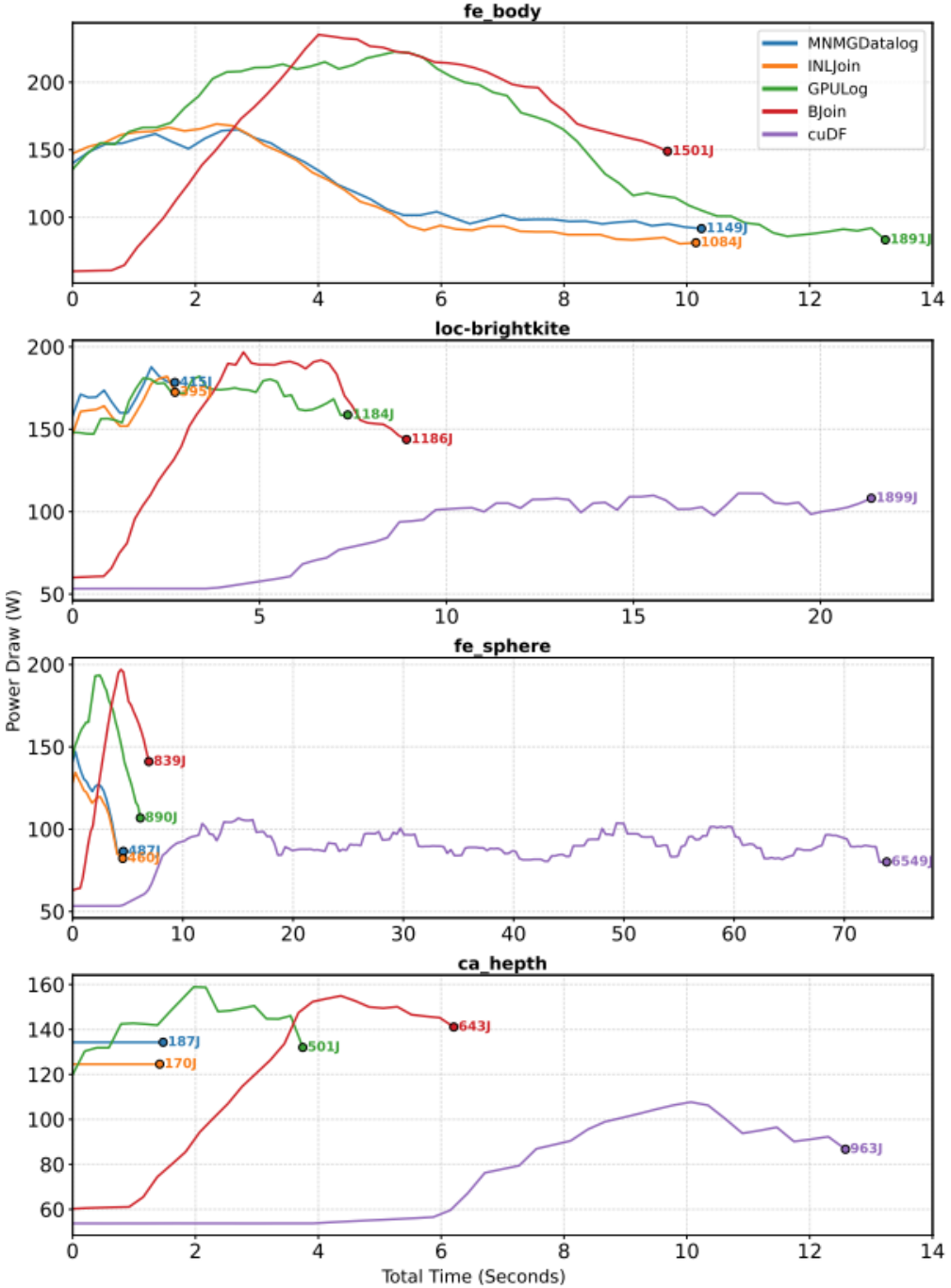
Appendix

Energy profile: SG energy consumption (single-GPU)

$$E = \sum_{i=1}^{N} P_i \cdot \Delta t_i$$

- N is the total number of sampling intervals
- P_i is the power draw (in watts) measured at interval i
- Δt_i is the time (in seconds) between samples i and i - 1





Power draw (SG)