USENIX ATC 2023

Towards Iterative Relational Algebra on the GPU

Authors:
Ahmedur Rahman Shovon, Thomas Gilray, Kristopher Micinski, Sidharth Kumar

THE UNIVERSITY OF ALABAMA AT BIRMINGHAM.

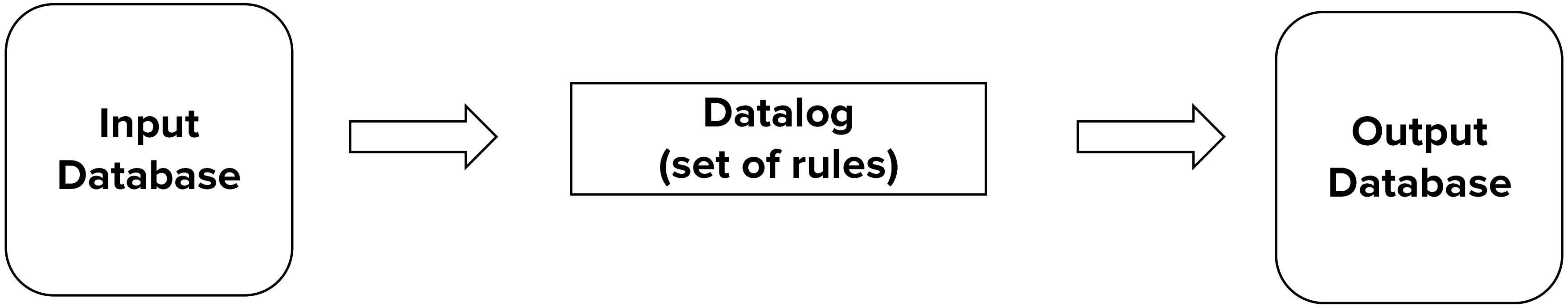
Syracuse University

Table of Contents

Table of Contents

Datalog: Bottom-Up Logic Programming Language

A lightweight logic-programming language for deductive-database systems



Running the Datalog program extends data from input database creating the output database with all data transitively derivable via the program rules

- Ceri, S., Gottlob, G., & Tanca, L. (1989). What you always wanted to know about Datalog(and never dared to ask). IEEE transactions on knowledge and data engineering, 1(1), 146-166.
- Gilray, T., Kumar, S., & Micinski, K. (2021, March). Compiling data-parallel datalog. In Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction (pp. 23-35).

Classic Problems for Datalog

- Transitive closure
- Triangle counting
- Finding maximal cliques
- Finding frequent itemsets
- Data mining

- Oege De Moor, Georg Gottlob, Tim Furche, and Andrew Sellers. Datalog Reloaded: First InternationalWorkshop, Datalog 2010, Oxford, UK, March 16-19, 2010. Revised Selected Papers, volume 6702.Springer, 2012.
- Jiwon Seo, Stephen Guo, and Monica S Lam. Socialite: Datalog extensions for efficient social networkanalysis. In 2013 IEEE 29th International Conference on Data Engineering (ICDE), pages 278–289.IEEE, 2013.

Bottom-Up Logic Programming with Datalog

Datalog → Iterative Relational Algebra

Datalog rule for computing Transitive Closure (TC)

```
T(x,y) <- G(x,y).
T(x,z) <- T(x,y), G(y,z).
```

Operationalized as a fixed-point iteration using F_G

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

Relational algebra: **Union** **Projection** **Join**

- 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.
- Kumar, S., & Gilray, T. (2020, June). Load-balancing parallel relational algebra. In International Conference on High Performance Computing (pp. 288-308). Springer, Cham.

Table of Contents

Relational Algebra Primitives

- Main relational algebra primitives of two flat relations **R** and **S** are:
  - Union: $R \cup S$
  - Intersection: $R \cap S$
  - Cartesian product: $R \times S$
  - Join: $R \bowtie S$
  - Rename: $\rho_{(i,j)}(R)$
  - Selection: $\sigma_i(R)$
  - Projection: $\Pi_{(i,j)}(R)$

Relation

UserID	UserName	UserEmail
101	Alice	alice@example.com
102	Bob	bob@example.com

Tuple (Row)

Attribute (Column)

- Differ from traditional set theory: **R** and **S** have a fixed arity

- Sidharth Kumar and Thomas Gilray. Distributed relational algebra at scale. In International Conference on High Performance Computing, Data, and Analytics (HiPC). IEEE, 2019.
- Sidharth Kumar and Thomas Gilray. Load-balancing parallel relational algebra. In International Conference on High Performance Computing, pages 288–308. Springer, 2020.

Example of Natural Join ⋈

User

UserID	UserName	UserEmail
101	Alice	alice@example.com
102	Bob	bob@example.com
103	Eve	eve@example.com

⋈

Order

UserID	OrderTotal	Items
101	25.69	2
102	145.66	3
103	12.11	1
103	44.00	2

Example of Natural Join ⋈

User

UserID	UserName	UserEmail
101	Alice	alice@example.com
102	Bob	bob@example.com
103	Eve	eve@example.com

⋈

Order

UserID	OrderTotal	Items
101	25.69	2
102	145.66	3
103	12.11	1
103	44.00	2

User ⋈ **Order**

UserID	UserName	UserEmail	OrderTotal	Items
101	Alice	alice@example.com	25.69	2

Example of Natural Join ⋈

User

UserID	UserName	UserEmail
101	Alice	alice@example.com
102	Bob	bob@example.com
103	Eve	eve@example.com

⋈

Order

UserID	OrderTotal	Items
101	25.69	2
102	145.66	3
103	12.11	1
103	44.00	2

User ⋈ **Order**

UserID	UserName	UserEmail	OrderTotal	Items
101	Alice	alice@example.com	25.69	2
102	Bob	bob@example.com	145.66	3

Example of Natural Join ⋈

User

UserID	UserName	UserEmail
101	Alice	alice@example.com
102	Bob	bob@example.com
103	Eve	eve@example.com

⋈

Order

UserID	OrderTotal	Items
101	25.69	2
102	145.66	3
103	12.11	1
103	44.00	2

User ⋈ **Order**

UserID	UserName	UserEmail	OrderTotal	Items
101	Alice	alice@example.com	25.69	2
102	Bob	bob@example.com	145.66	3
103	Eve	eve@example.com	12.11	1

Example of Natural Join ⋈

User

UserID	UserName	UserEmail
101	Alice	alice@example.com
102	Bob	bob@example.com
103	Eve	eve@example.com

⋈

Order

UserID	OrderTotal	Items
101	25.69	2
102	145.66	3
103	12.11	1
103	44.00	2

User ⋈ Order

UserID	UserName	UserEmail	OrderTotal	Items
101	Alice	alice@example.com	25.69	2
102	Bob	bob@example.com	145.66	3
103	Eve	eve@example.com	12.11	1
103	Eve	eve@example.com	44.00	2

Duplicates on Join Result

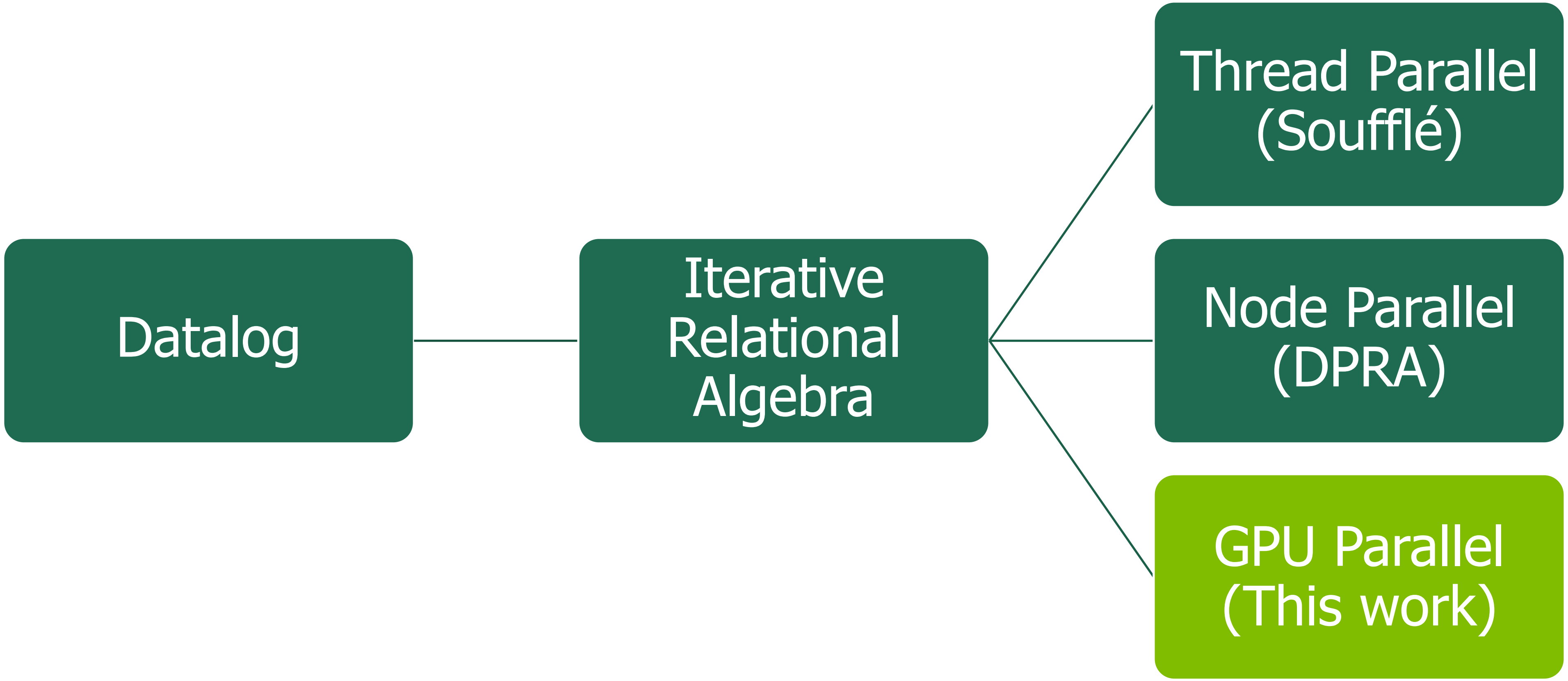
$$\text{User} \bowtie \text{Order}$$

UserID	UserName	UserEmail	OrderTotal	Items
101	Alice	alice@example.com	25.69	2
102	Bob	bob@example.com	145.66	3
103	Eve	eve@example.com	12.11	1
103	Eve	eve@example.com	44.00	2

$$\Pi_{(\text{UserName},\text{UserEmail})}(\text{User} \bowtie \text{Order})$$

UserName	UserEmail
Alice	alice@example.com
Bob	bob@example.com
Eve	eve@example.com
Eve	eve@example.com

Towards Parallel Relational Algebra



- Herbert Jordan, Bernhard Scholz, and Pavle Suboti´c. Souffl´e: On synthesis of program analyzers. InInternational Conference on Computer Aided Verification, pages 422–430. Springer, 2016.
- Kumar, S., & Gilray, T. (2019). Distributed relational algebra at scale. In International Conference on High Performance Computing, Data, and Analytics (HiPC). IEEE (Vol. 1).
- Thomas Gilray, Sidharth Kumar, and Kristopher Micinski. Compiling data-parallel datalog. In Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction, CC 2021,page 23–35, New York, NY, USA, 2021. Association for Computing Machinery.

Parallel Join: Algorithms

Popular algorithms

Sort-Merge Join (SMJ)	Hash Join (HJ)

SMJ is suitable for small to medium-sized tables, HJ is suitable for large tables

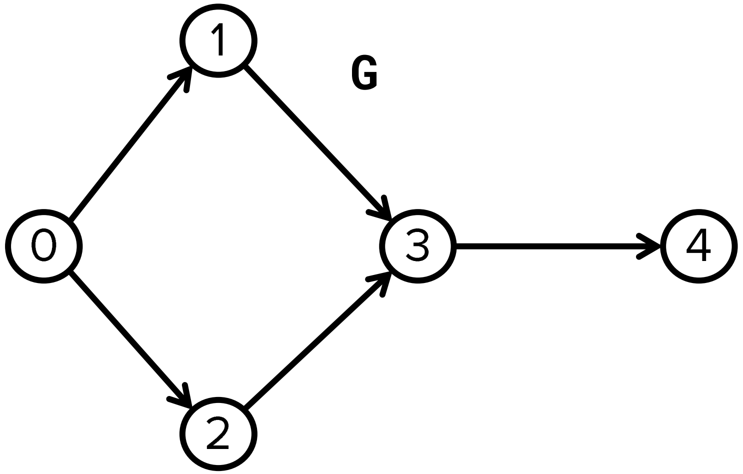
- Chengxin Guo, Hong Chen, Feng Zhang, and Cuiping Li. Parallel hybrid join algorithm on gpu. 2019IEEE 21st International Conference on High Performance Computing and Communications; IEEE17th International Conference on Smart City; IEEE 5th International Conference on Data Science andSystems (HPCC/SmartCity/DSS), pages 1572–1579, 2019.
- Hongzhi Wang, Ning Li, Zheng ke Wang, and Jianing Li. Gpu-based efficient join algorithms on hadoop.The Journal of Supercomputing, 77:292 – 321, 2020.

Research Gaps in Parallel Join Implementations

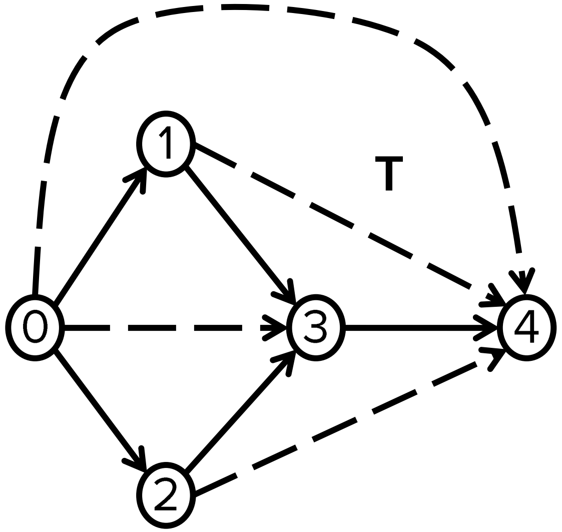
- GPU-based join implementations does not sort result (by default)
- Challenge for iterated relational algebra algorithms
- Negative impact on algorithm performance
- Memory overhead in Python libraries

Table of Contents

Transitive Closure: Logical Inference for Graphs




```
T(x,y) <- G(x,y)
T(x,z) <- T(x,y),G(y,z)
```

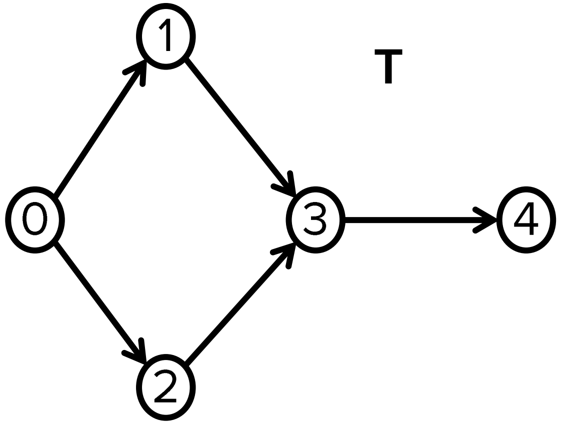


0	1
0	2
1	3
2	3
3	4

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

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

Transitive Closure: Iterations

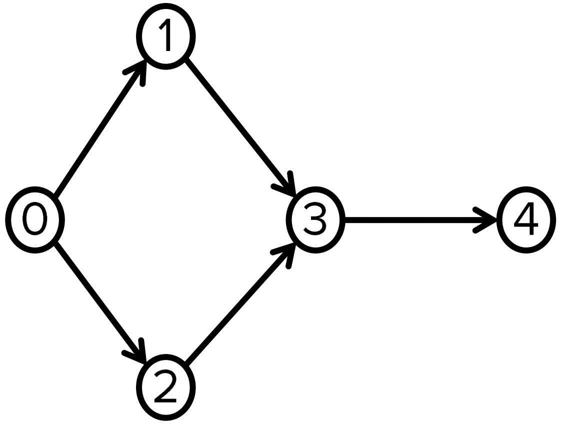


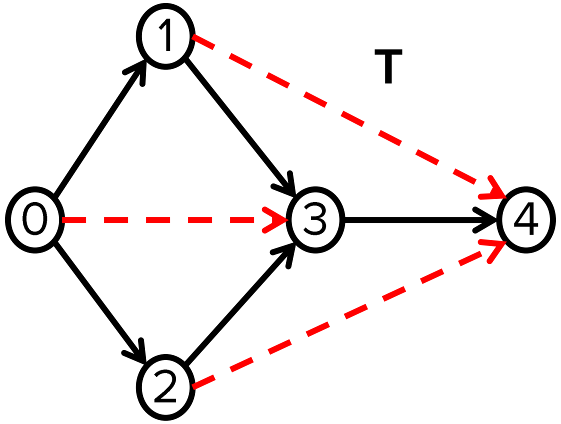
T

0	1
0	2
1	3
2	3
3	4

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

Transitive Closure: Iterations 1



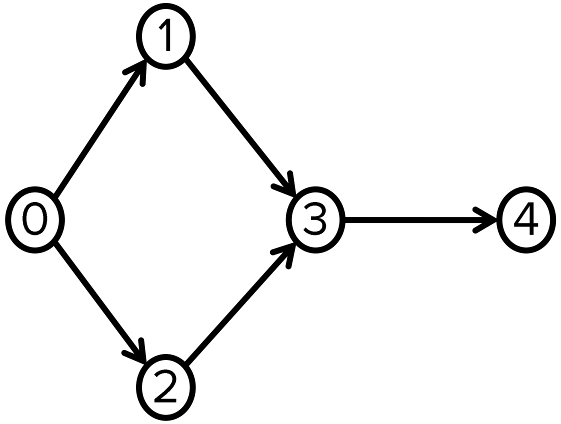


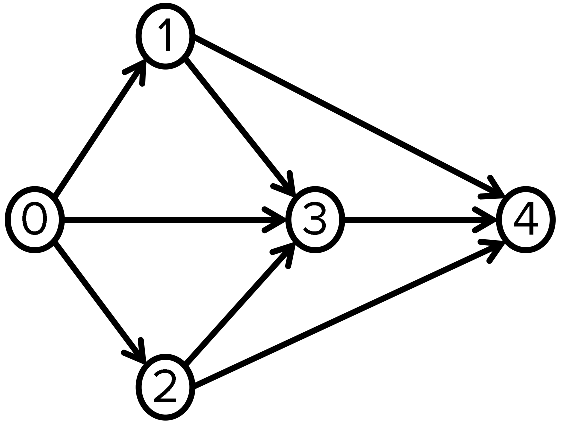
0	1
0	2
1	3
2	3
3	4

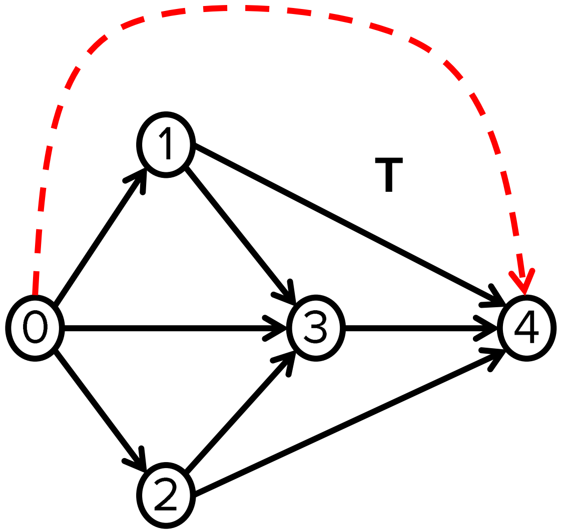
0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4

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

Transitive Closure: Iterations 2







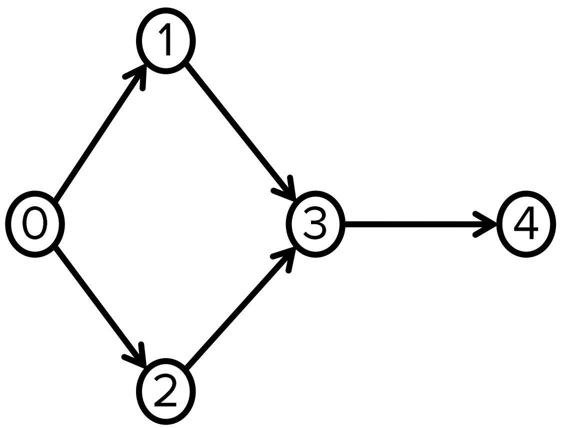
0	1
0	2
1	3
2	3
3	4

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

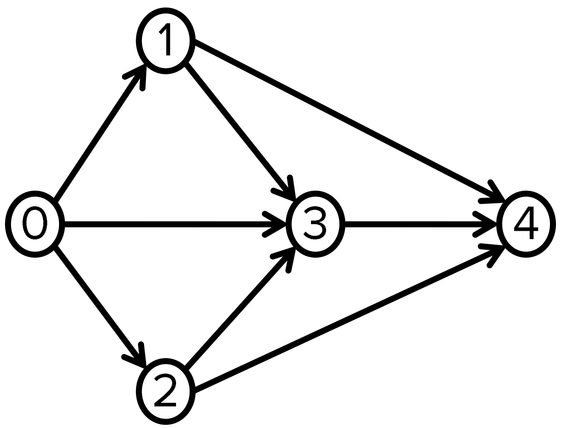
0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4
0	4

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

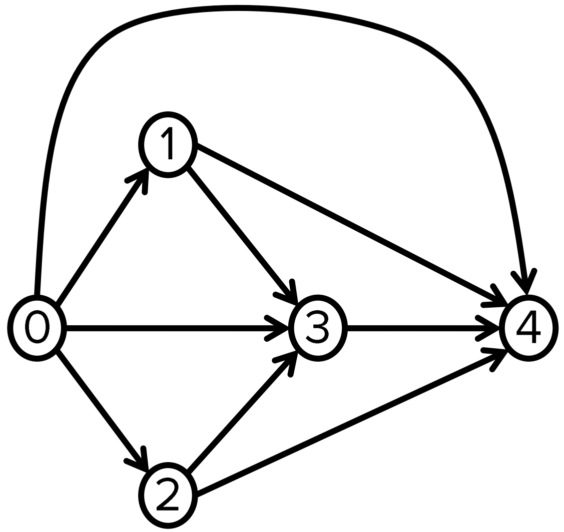
Transitive Closure: Iterations 3



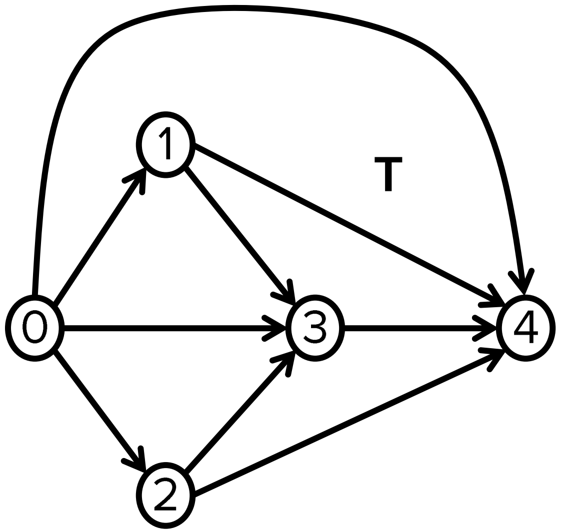
0	1
0	2
1	3
2	3
3	4



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



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

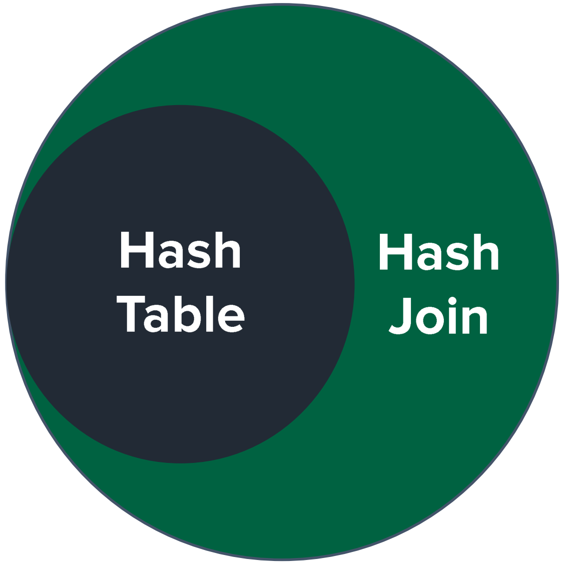


0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4
0	4

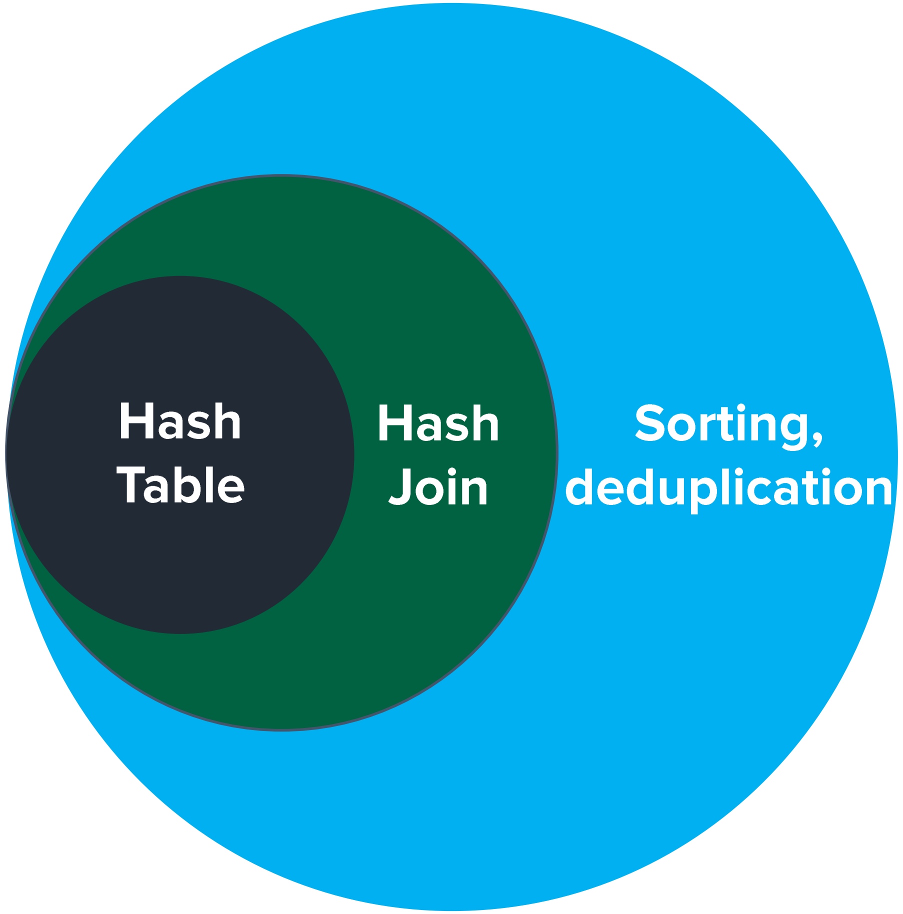
TC Computation in Iterated Relational Algebra

Hash
Table

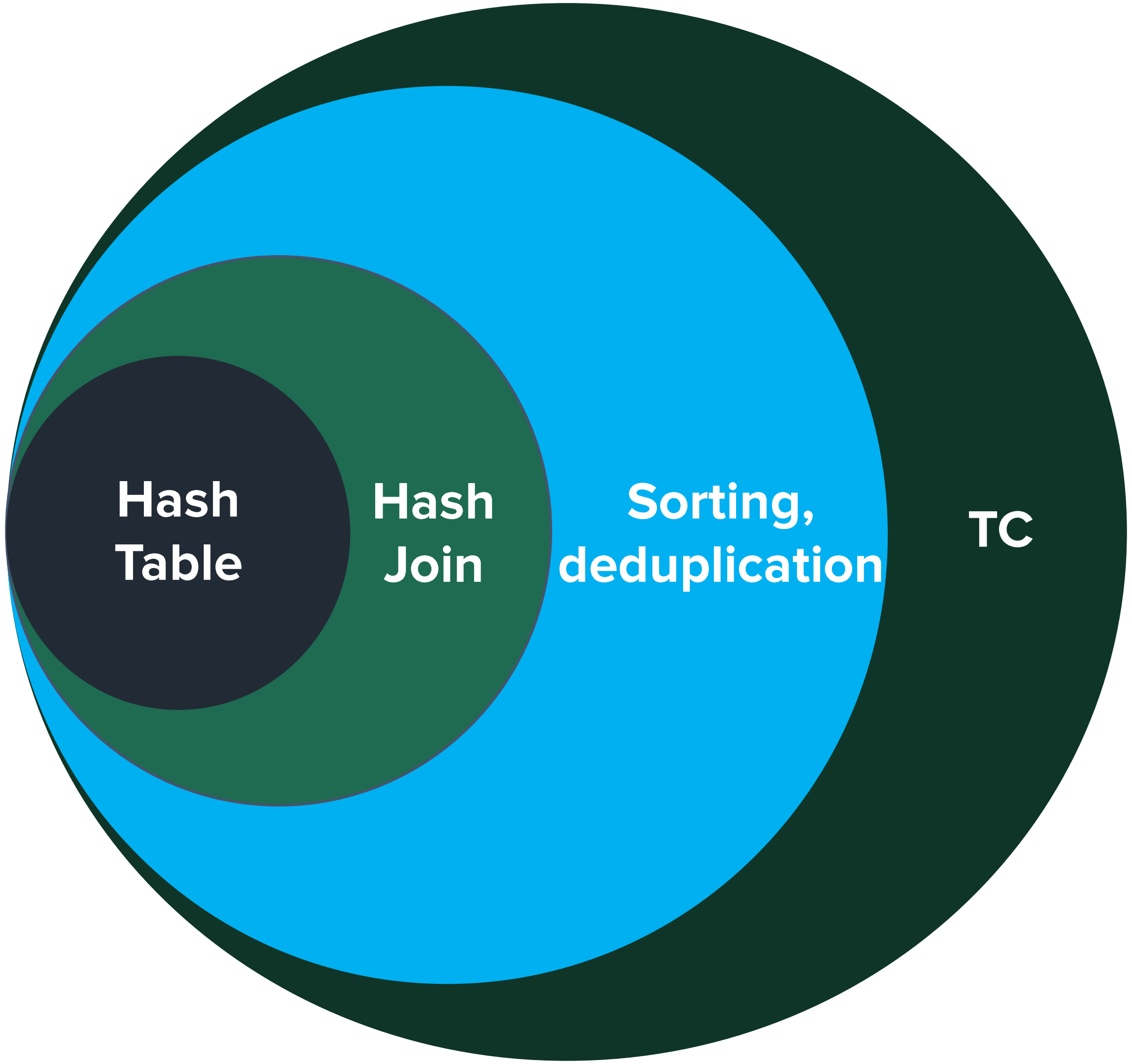
TC Computation in Iterated Relational Algebra



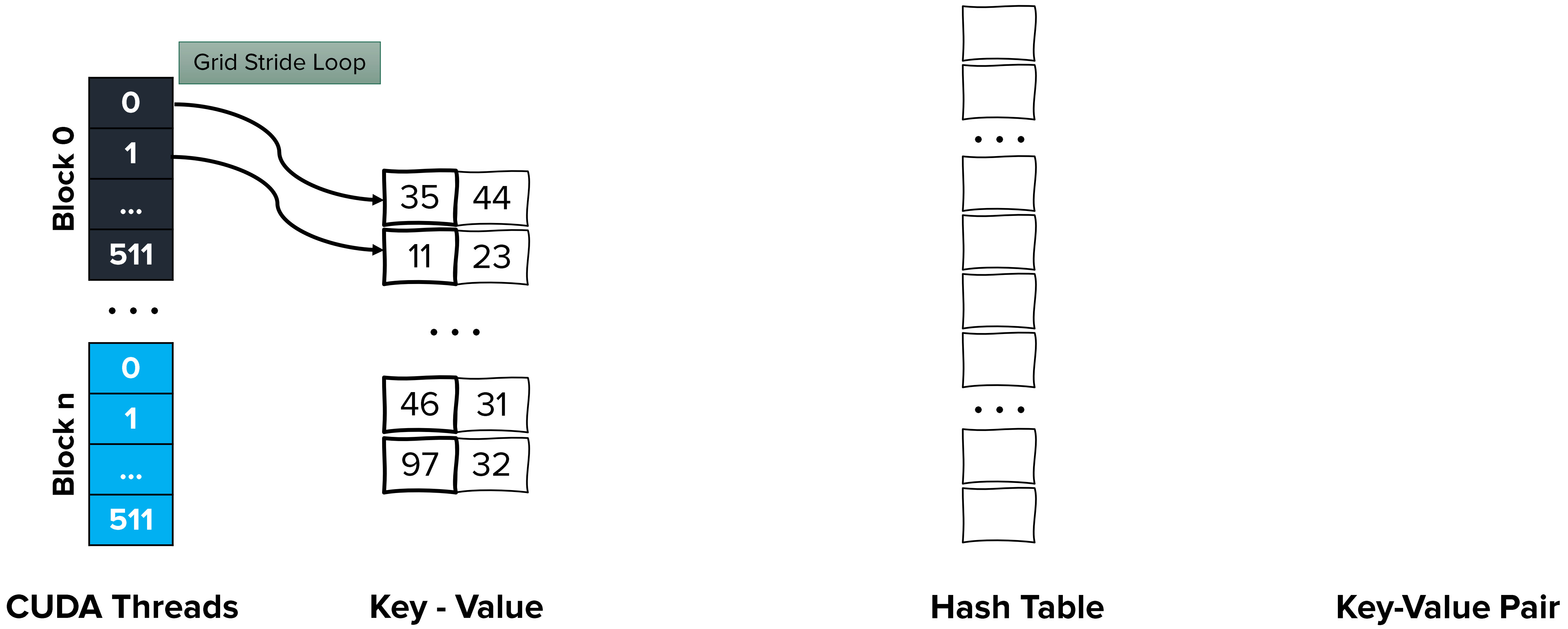
TC Computation in Iterated Relational Algebra



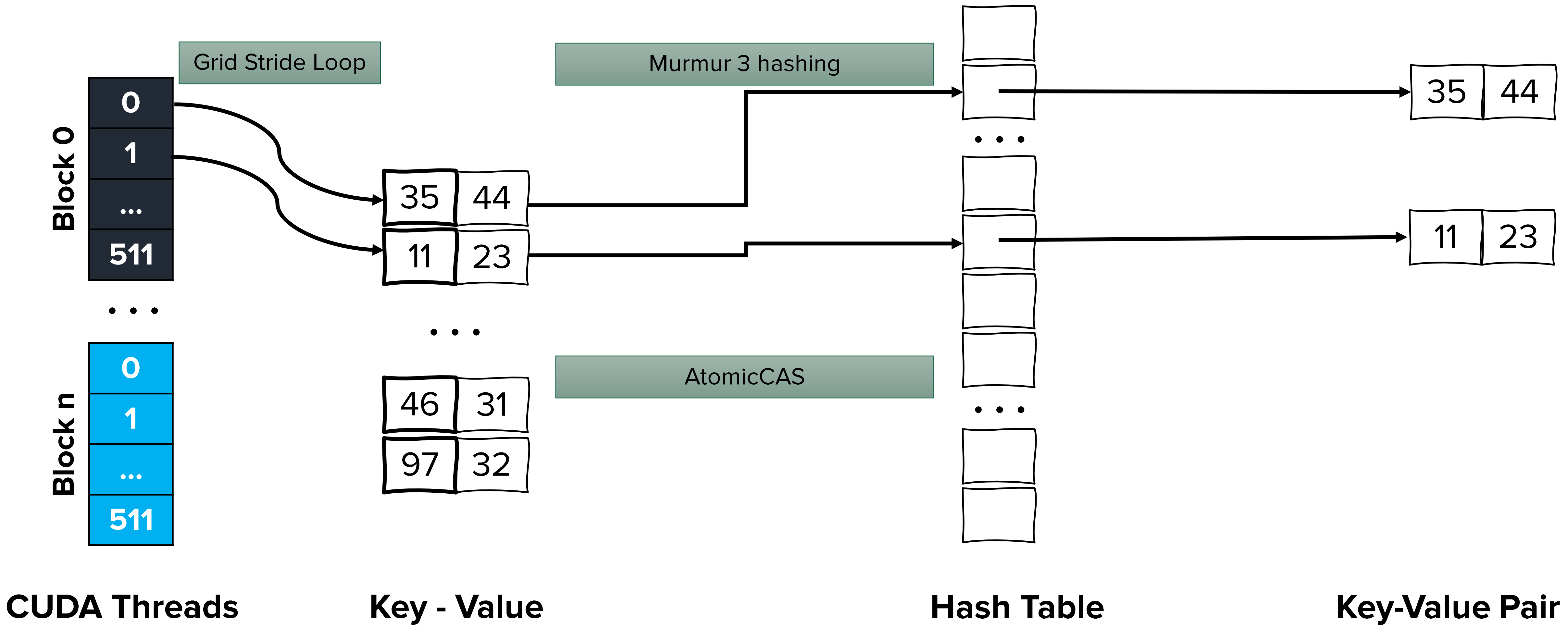
TC Computation in Iterated Relational Algebra



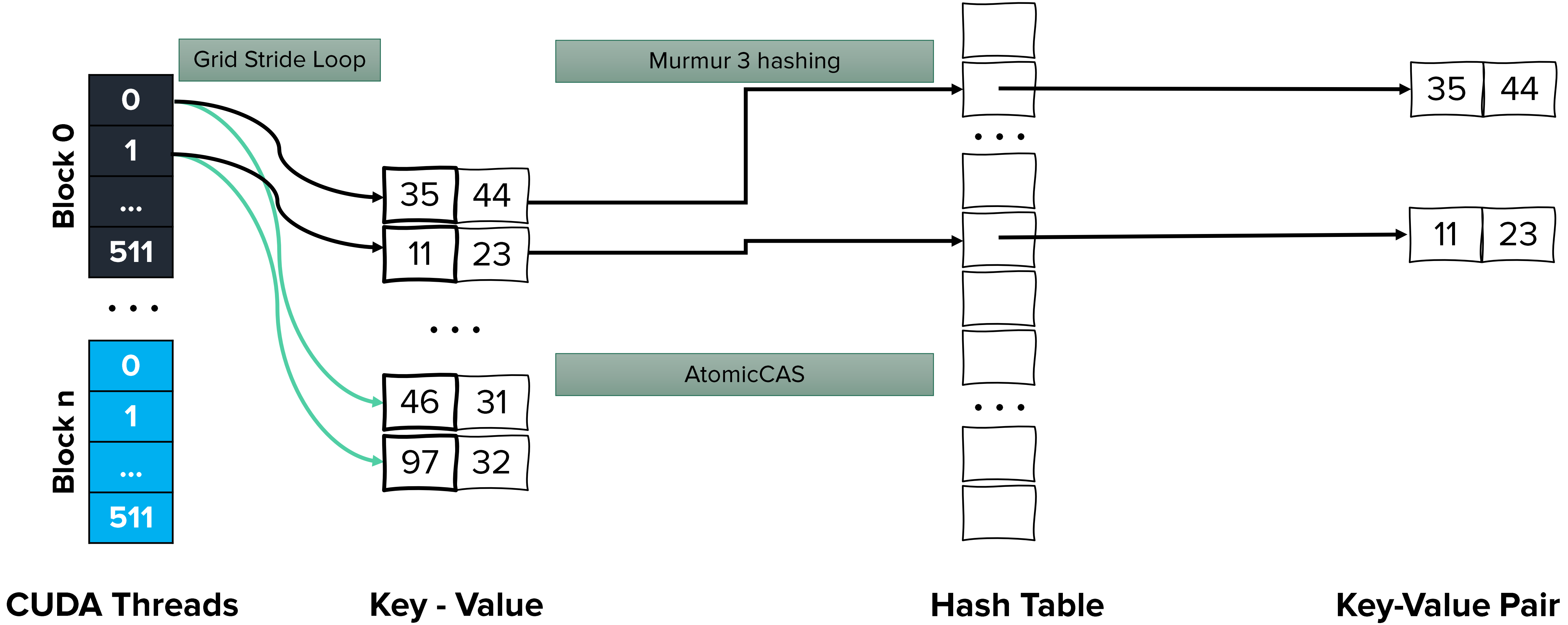
Hash Table (Open Addressing, Linear Probing)



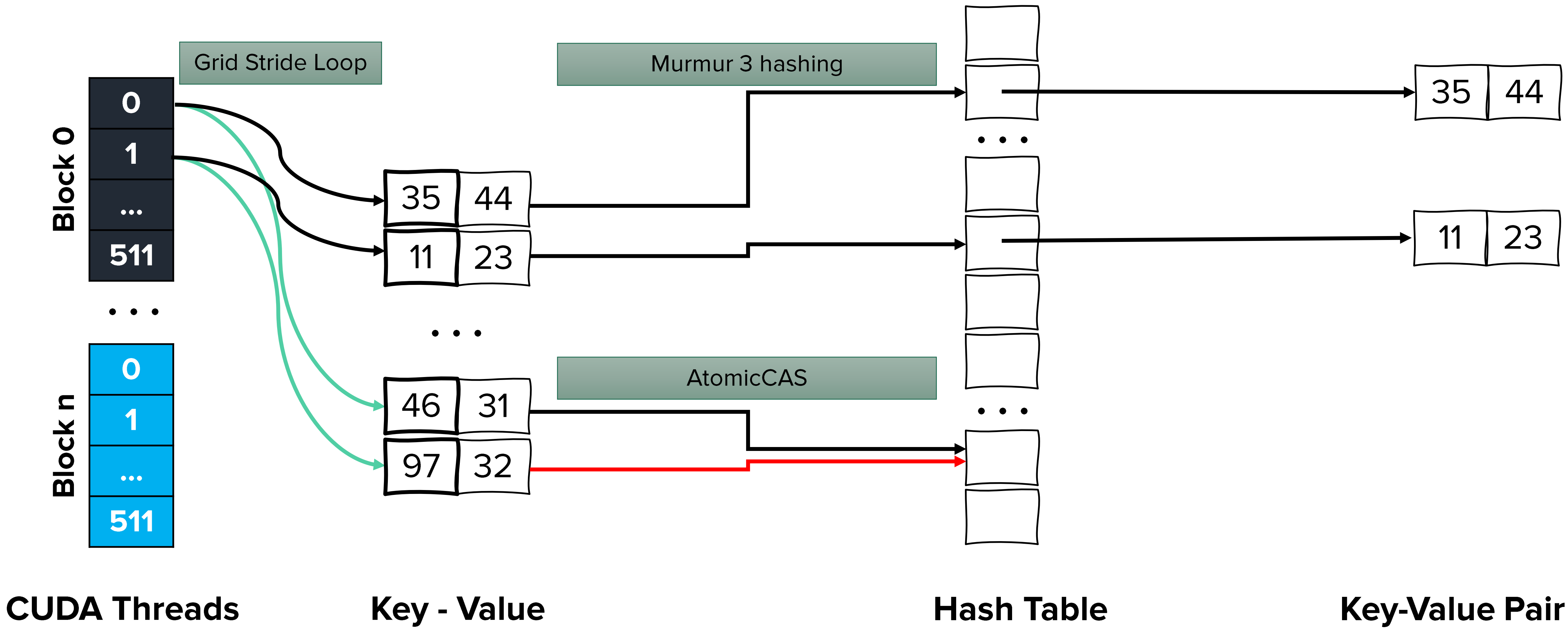
Hash Table (Open Addressing, Linear Probing)



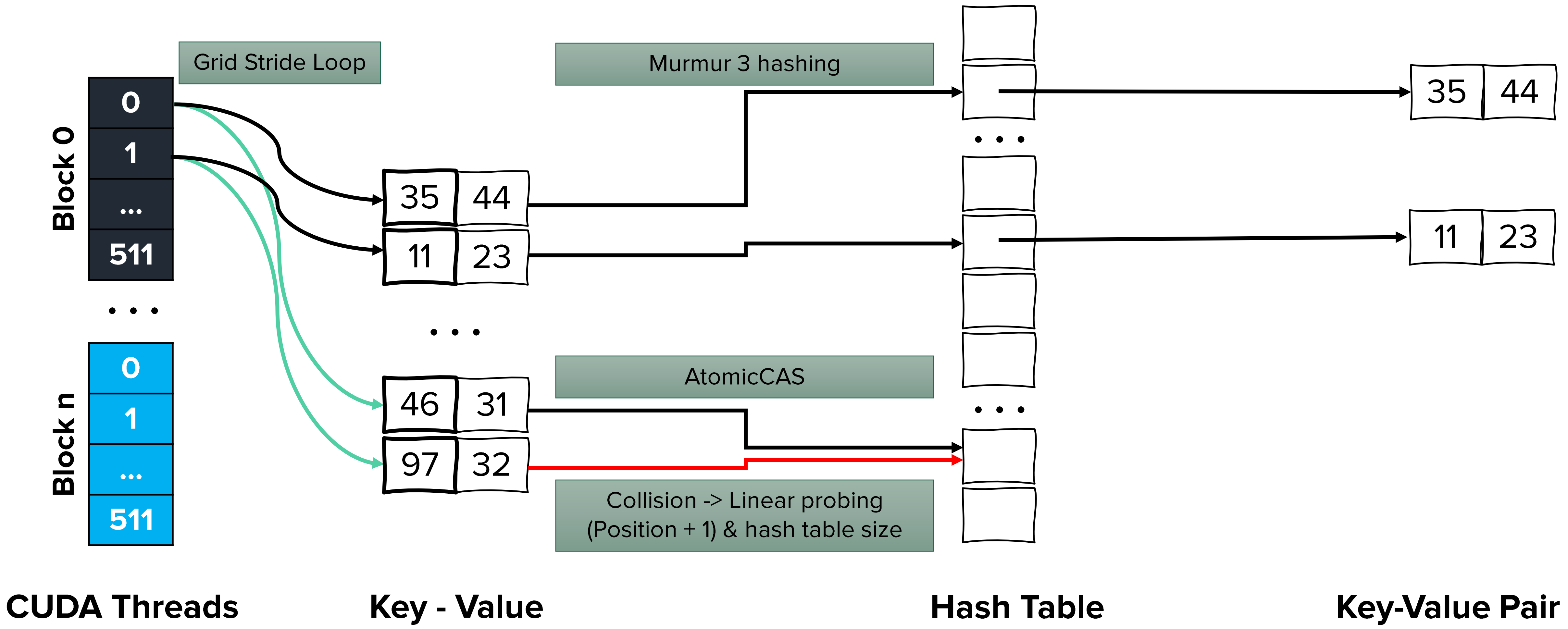
Hash Table (Open Addressing, Linear Probing)



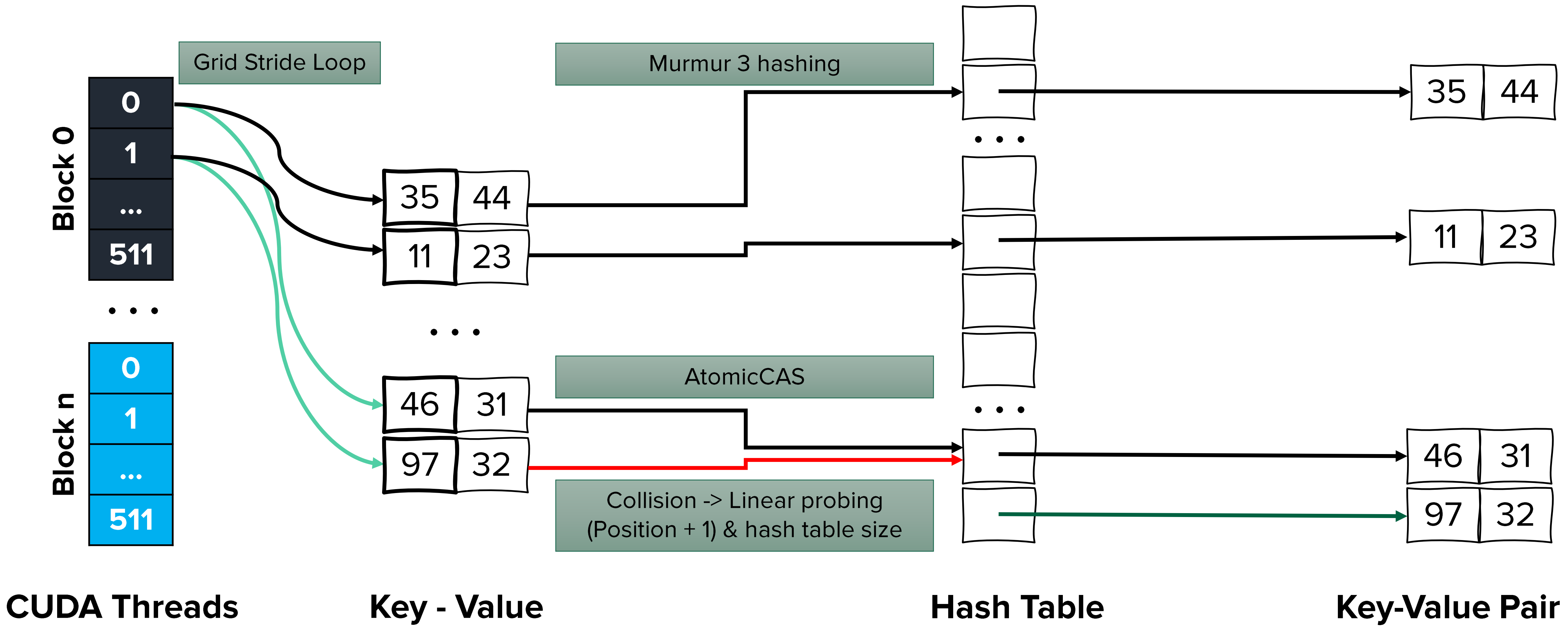
Hash Table (Open Addressing, Linear Probing)



Hash Table (Open Addressing, Linear Probing)



Hash Table (Open Addressing, Linear Probing)



Performing Hash Join on GPU

Static Hash Table

Key	Value

$\bowtie$

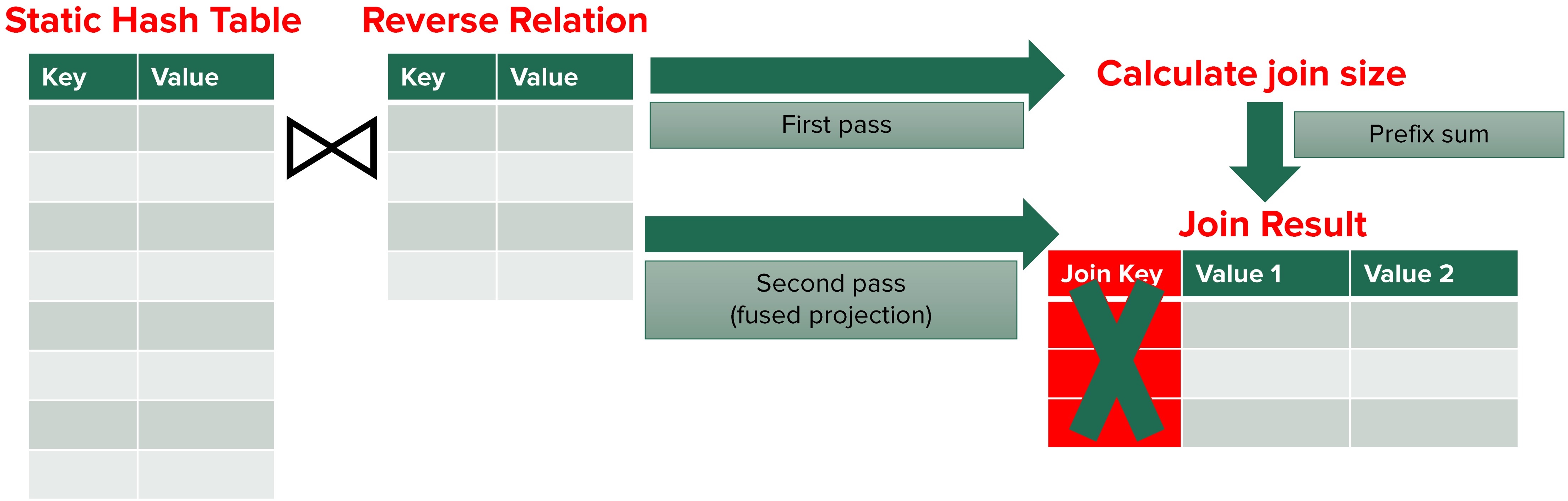
Reverse Relation

Key	Value

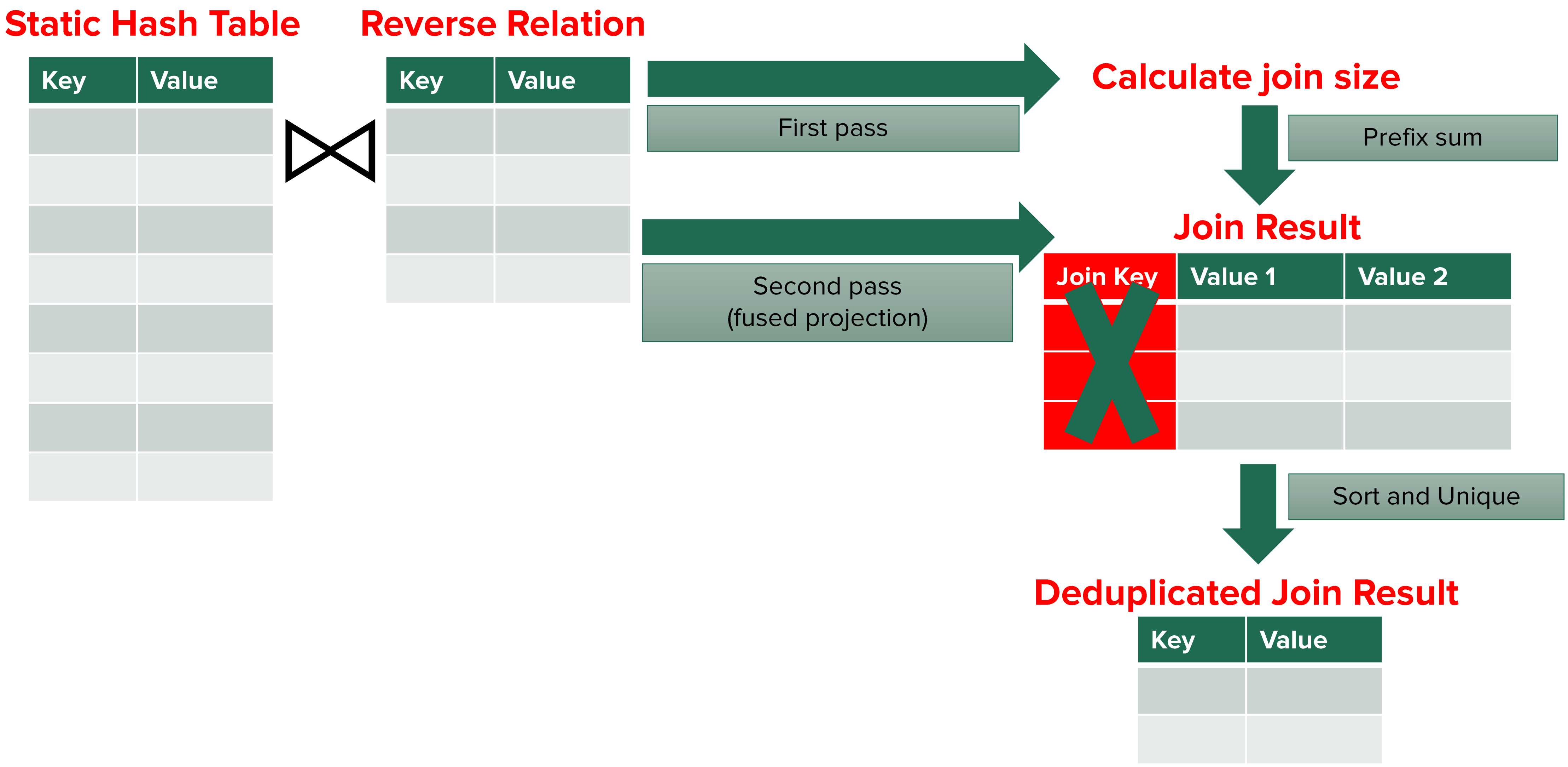
First pass

Calculate join size

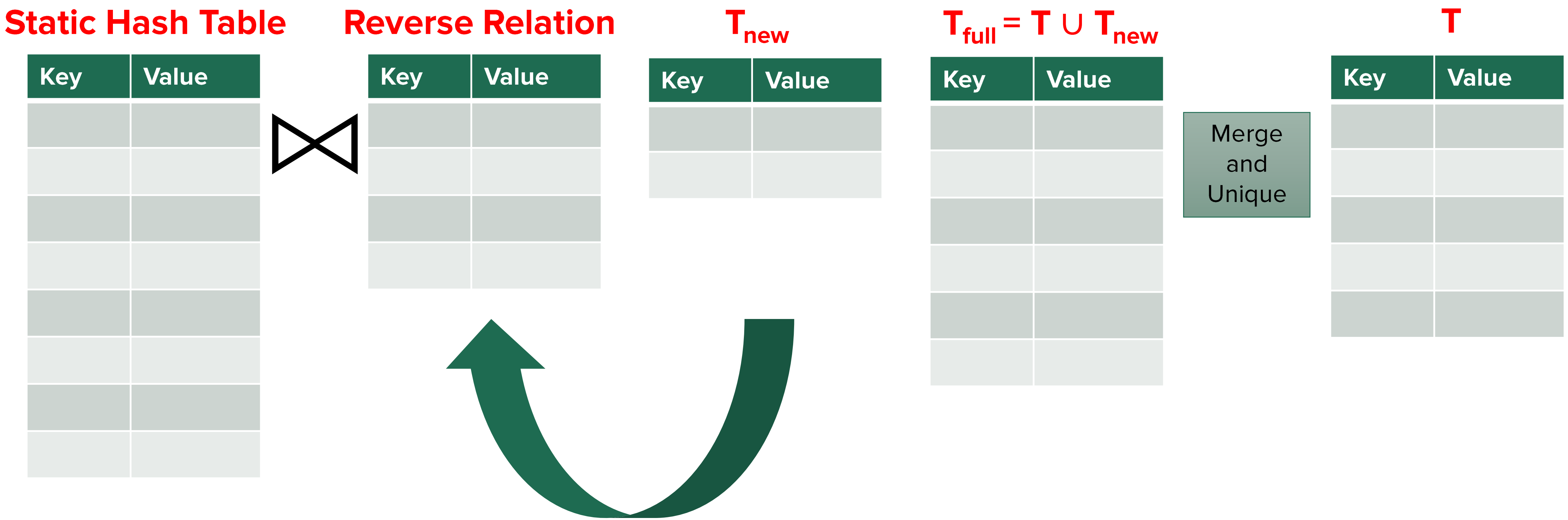
Performing Hash Join on GPU



Performing Hash Join on GPU



Transitive closure computation (single iteration)



Process continues until there is no new facts are discovered in an iteration

Table of Contents

Experiment Platform and Datasets

ThetaGPU supercomputer from Argonne National Lab

CPU: AMD EPYC 7742 processors with 3.31GHz clock speed, 128 cores

GPU

- NVIDIA A100 Tensor Core GPU with 40GB GPU memory
- 108 multiprocessors on device (SM)

Environment

- CUDA version 11.4, 3,456 x 512 (blocks per grid x threads per block)
- Souffle version 2.3 with 128 threads
- cuDF package inside conda environment

Datasets

- Stanford large network dataset collection
- SuiteSparse matrix collection
- Road network real datasets collection

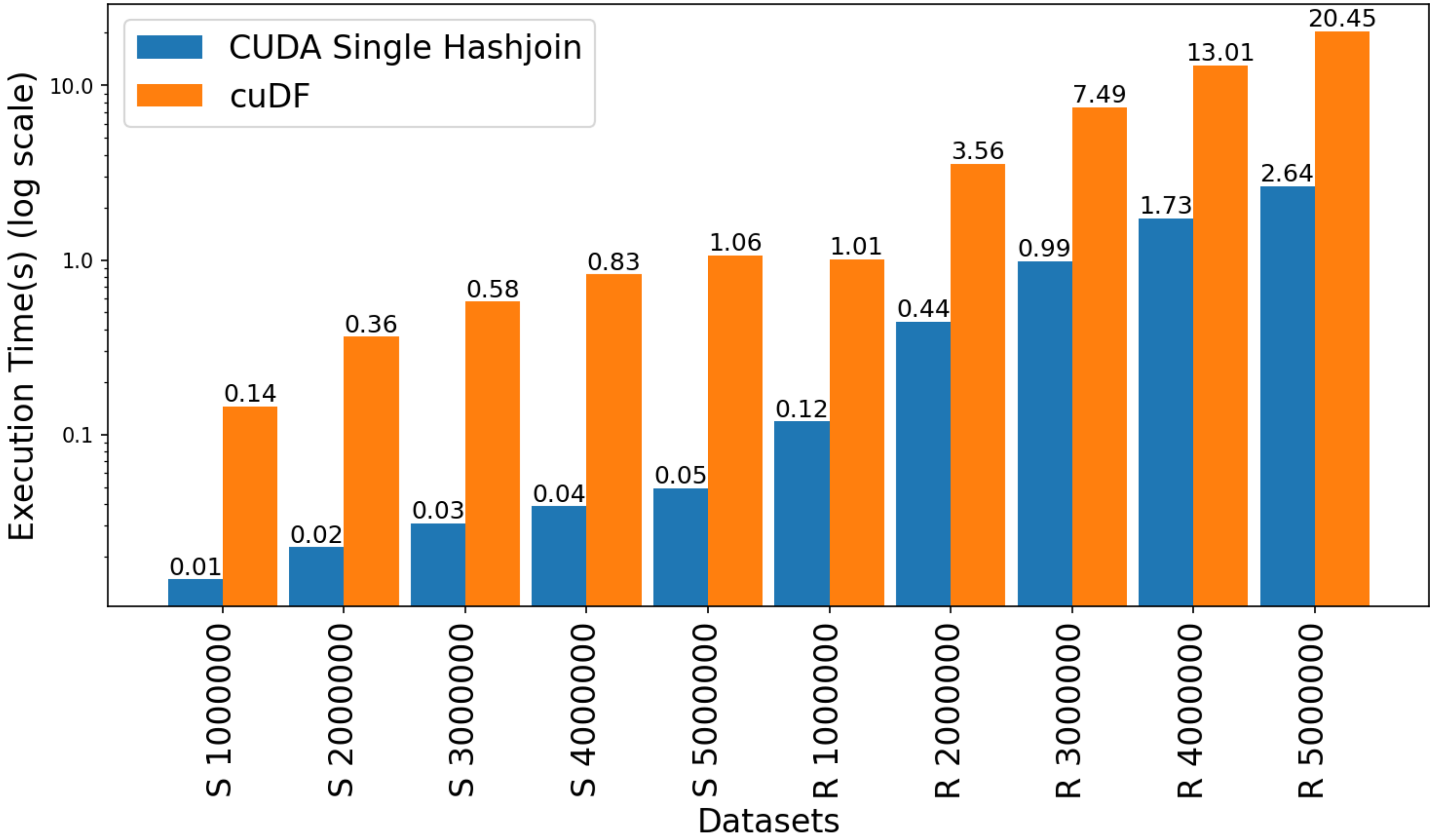
- Leskovec, J., & Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection.

Table of Contents

Hash Table Performance

- Build rate:
  - Random synthetic graph: 400 million keys/second
  - String graph: 4 billion keys/second
- Load factors are varied to ensure less memory overhead

Join Performance Comparison: CUDA vs cuDF



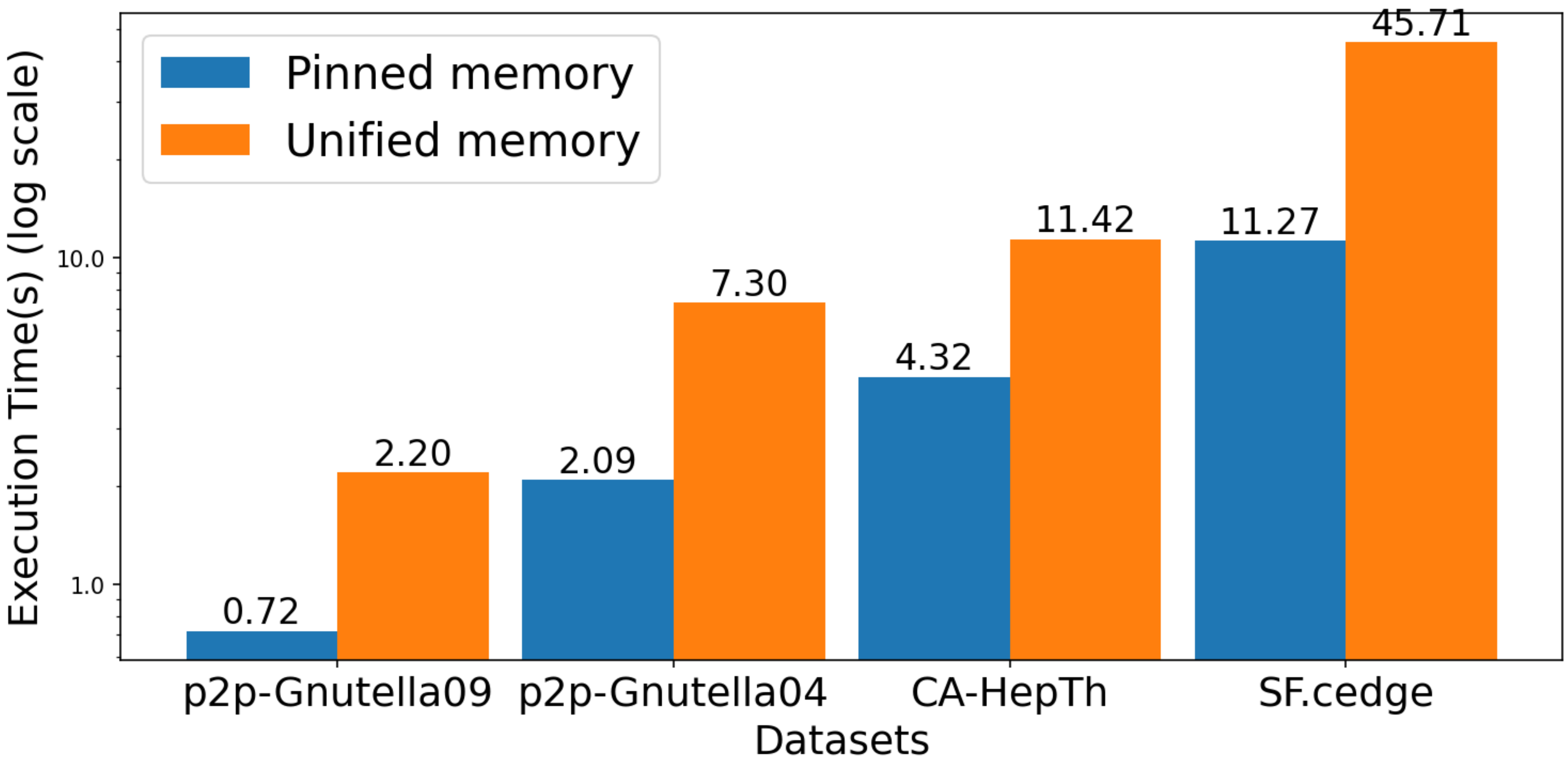
- Leadership Computing Facility, A. (2022). Argonne Leadership Computing Facility. Theta GPU Nodes. URL: https://www.alcf.anl.gov/support-center/theta-gpu-nodes

CUDA Advantages over Dataframe

- Fuse operations
- Thread-block configuration
- Memory management
- Optimize data structure

- Jason. Sanders. CUDA by example : an introduction to general-purpose GPU programming. AddisonWesley, Upper Saddle River, NJ, 2011.
- John Cheng, Max Grossman, and Ty McKercher. Professional CUDA c programming. John Wiley & Sons, 2014

TC Performance Comparison: Memory Schemes



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

TC Performance Comparison: CUDA vs Soufflé vs cuDF

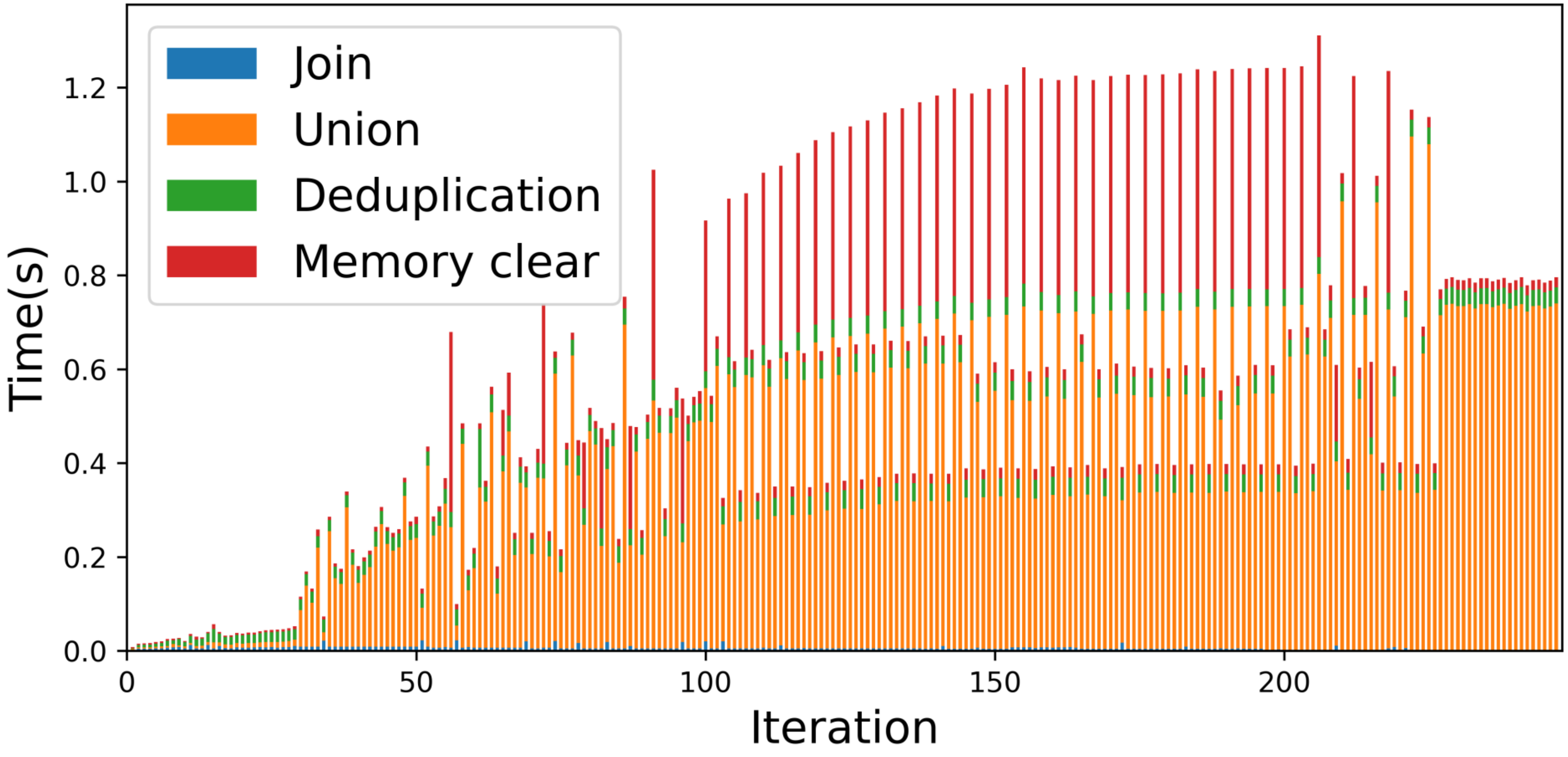
Dataset	Type	Rows	TC size	Iterations	CUDA Hashjoin(s)	Soufflé(s)	cuDF(s)
fe_ocean	U	409,593	1,669,750,513	247	138.237	536.233	Out of Memory
p2p-Gnutella31	D	147,892	884,179,859	31	Out of Memory	128.917	Out of memory
usroads	U	165,435	871,365,688	606	364.554	222.761	Out of Memory
fe_body	U	163,734	156,120,489	188	47.758	29.07	Out of Memory
loc-Brightkite	U	214,078	138,269,412	24	15.88	29.184	Out of Memory
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	0.72	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

Cases Where Souffle Outperforms CUDA

Overflows GPU memory when higher workload/iteration

Underperforms when less work for GPU/iteration

Operations Breakdown per Iteration (fe_ocean)

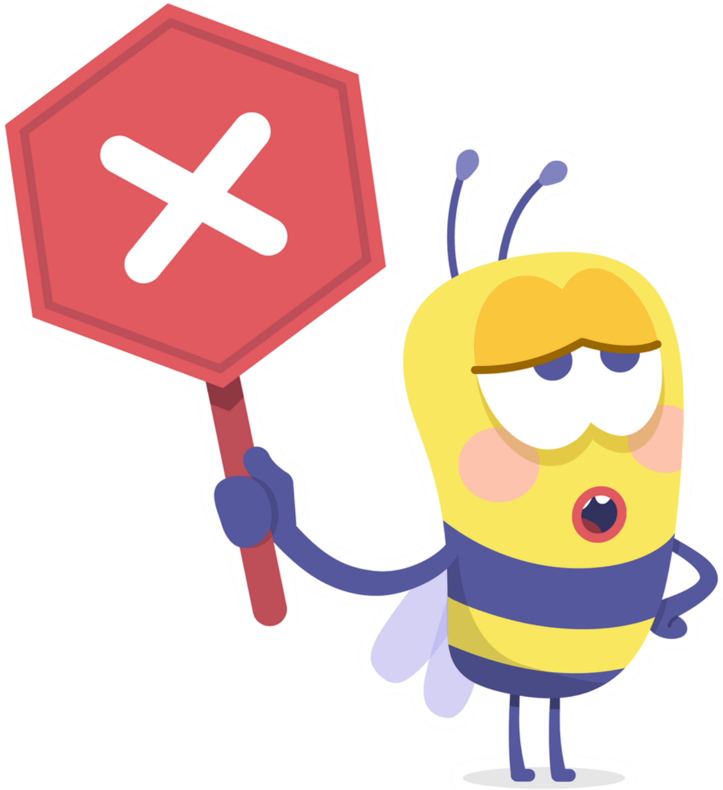


- Leadership Computing Facility, A. (2022). Argonne Leadership Computing Facility. Theta GPU Nodes. URL: https://www.alcf.anl.gov/support-center/theta-gpu-nodes

Contributions

- High Performance GPU hash table for iterative RA
- Operations optimization (fuse join and projection)
- Overcome deduplication challenge
- Efficient GPU memory management (pinned and buffer clearance)

Limitations



Limited to a single GPU that dictates scaling by available VRAM on the GPU

Memory overflow error for larger graphs

Open addressing based hash table causes memory overhead

Table of Contents

Future Work

Develop	Multi-node multi-GPU backend for Datalog to perform iterated relational algebra operations tailored for GPU
Compare	Different Parallel Programming Models performance on iterative relational join
Extend	State-of-the-art multi-node CPU-based Datalog-like language SLOG to leverage our GPU-based solutions

https://github.com/harp-lab/usenixatc23

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

Thank you!

HARP Lab

High-performance Automated Reasoning and Programming Lab

https://github.com/harp-lab/

THE UNIVERSITY OF ALABAMA AT BIRMINGHAM.

Appendix

DataFrame Based Datalog Applications

✓ Advantages

- ✓ Abstract memory management
- ✓ Abstract thread block configuration
- ✓ Same API signatures for CPU and GPU
- ✓ Easy-to-code interface

✗ Limitations

- ✗ No fusing
- ✗ Memory and computation overhead
- ✗ No consecutive operation
- ✗ Memory limitation

- 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.
- Team, R. D. (2018). RAPIDS: Collection of libraries for end to end GPU data science. NVIDIA, Santa Clara, CA, USA. https://rapids.ai

Datalog Example

Facts:

parents(x,y).
children(y,x).

extensional

Rules:

grandparent(x,y) :- parents(x,z), parents(z,y).

intensional

head ***body***

- Michael Stonebraker. Readings in database systems. Morgan Kaufmann Publishers Inc., 1988
- Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressivepower of logic programming. ACM Comput. Surv., 33(3):374–425, sep 2001.
- David Maier, K Tuncay Tekle, Michael Kifer, and David S Warren. Datalog: concepts, history, andoutlook. In Declarative Logic Programming: Theory, Systems, and Applications, pages 3–100. 2018.

Parallel Join

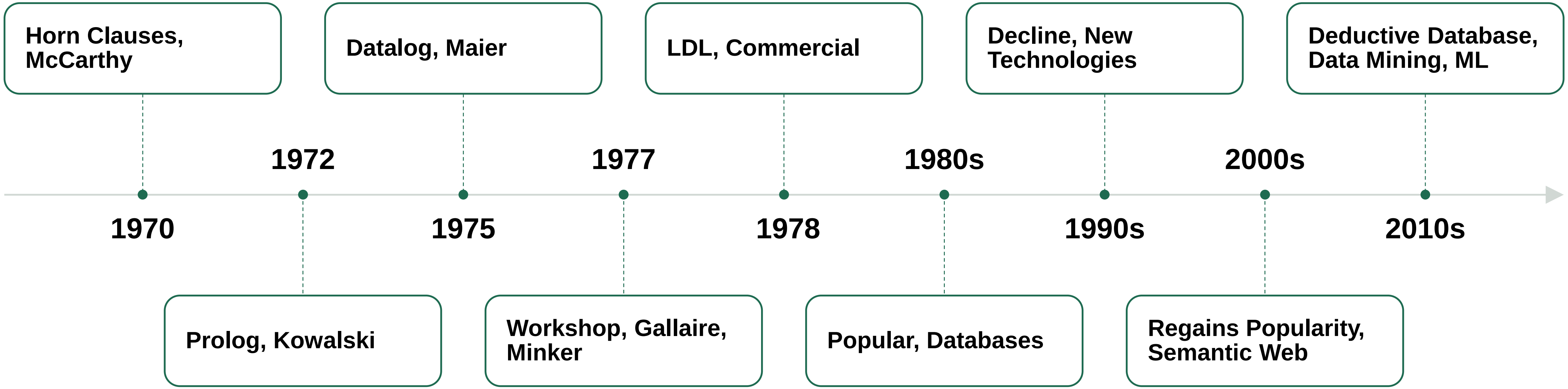
What: Perform relational join operation simultaneously on a number of processors or machines

When: Useful when input data is enormous and the join is computationally costly

How: Divide the data into partitions and assign each partition to a different processor

- Daniel Zinn, Haicheng Wu, Jin Wang, Molham Aref, and Sudhakar Yalamanchili. General-purpose join algorithms for large graph triangle listing on heterogeneous systems. In Proceedings of the 9th Annual Workshop on General Purpose Processing Using Graphics Processing Unit, pages 12–21, 2016.

Datalog Timeline



- Stefano Ceri, Georg Gottlob, Letizia Tanca, et al. What you always wanted to know about datalog(andnever dared to ask). IEEE transactions on knowledge and data engineering, 1(1):146–166, 1989
- David Maier, K Tuncay Tekle, Michael Kifer, and David S Warren. Datalog: concepts, history, andoutlook. In Declarative Logic Programming: Theory, Systems, and Applications, pages 3–100. 2018.
- Shan Shan Huang, Todd Jeffrey Green, and Boon Thau Loo. Datalog and emerging applications:An interactive tutorial. In Proceedings of the 2011 ACM SIGMOD International Conference onManagement of Data, SIGMOD ’11, page 1213–1216, New York, NY, USA, 2011. Association forComputing Machinery.

Datalog Applications

- Modern Datalog implementations: Soufflé, SLOG, LogicBlox, RadLog, PRAM
- Domains
  - Deductive database
  - Machine learning
  - Software analysis
  - Business analytics

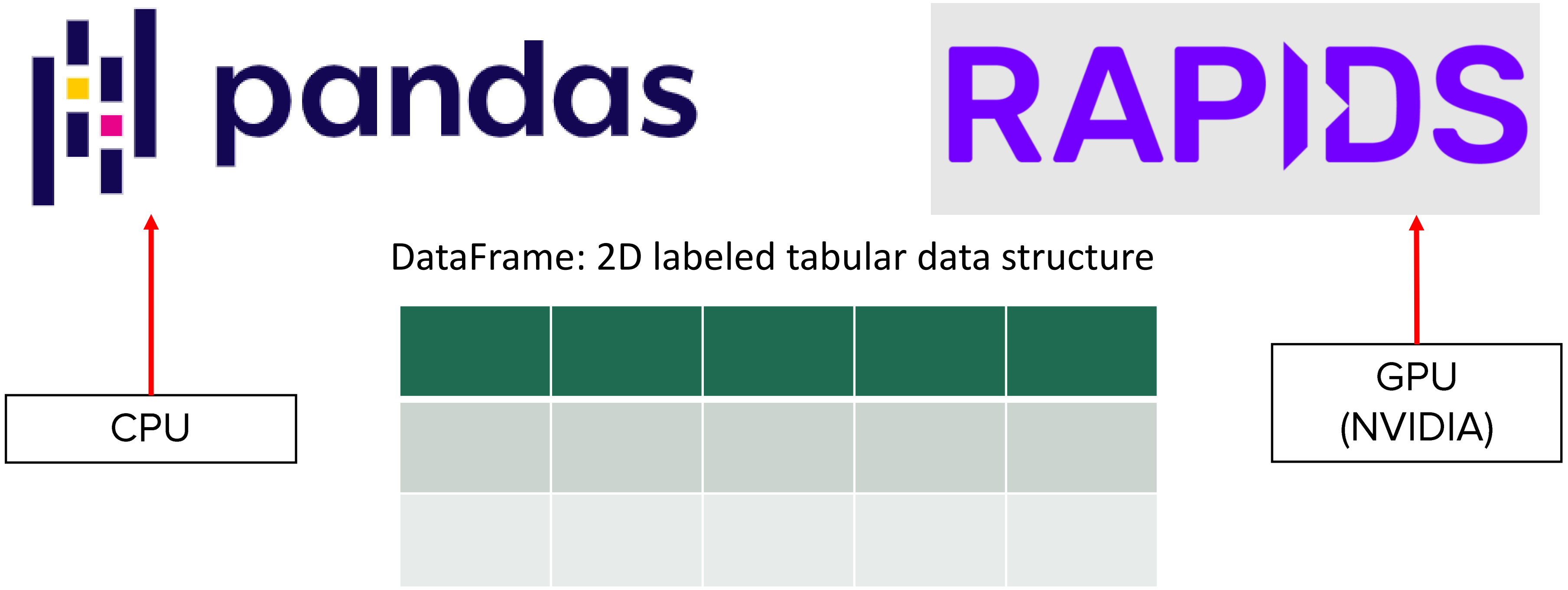
- Martin Bravenboer and Yannis Smaragdakis. Strictly declarative specification of sophisticated points-toanalyses. In Proceedings of the 24th ACM SIGPLAN conference on Object oriented programming systems languages and applications, pages 243–262, 2009.
- Jiwon Seo, Stephen Guo, and Monica S Lam. Socialite: Datalog extensions for efficient social networkanalysis. In 2013 IEEE 29th International Conference on Data Engineering (ICDE), pages 278–289.IEEE, 2013

Algorithm for TC computation using CUDA

- Open-Addressing based hash table
- Two pass approach to perform hash join on the GPU
- Deduplication using sort and unique, merge and unique

```
procedure TRANSITIVECLOSURE(Graph G)
    R ← HashTable(G)
    result ← Sort(G)
    T_Δ ← G
    repeat
        joinSizePerRow ← JoinSize(R, T_Δ)
        joinOffset ← Scan(joinSizePerRow)
        Initialize(joinResult, totalJoinSize)
        joinResult ← Join((R, T_Δ), joinOffset)
        joinResult ← Sort(joinResult)
        joinResult ← RemoveDuplicates(joinResult)
        totalUniqueJoinSize ← Size(joinResult)
        FreeMemory(T_Δ)
        T_Δ ← Copy(joinResult, totalUniqueJoinSize)
        unionSize ← resultSize + totalUniqueJoinSize
        Initialize(unionResult, unionSize)
        unionResult ← MergeSortedArrays(result, joinResult)
        unionResult ← RemoveDuplicates(unionResult)
        uniqueUnionSize ← Size(unionResult)
        oldUnionSize ← Size(result)
        FreeMemory(result)
        result ← Copy(unionResult, uniqueUnionSize)
        FreeMemory(joinOffset)
        FreeMemory(joinResult)
        FreeMemory(unionResult)
    until oldUnionSize ≠ uniqueUnionSize
    FreeMemory(R)
    FreeMemory(result)
    FreeMemory(T_Δ)
    return result
end procedure
```

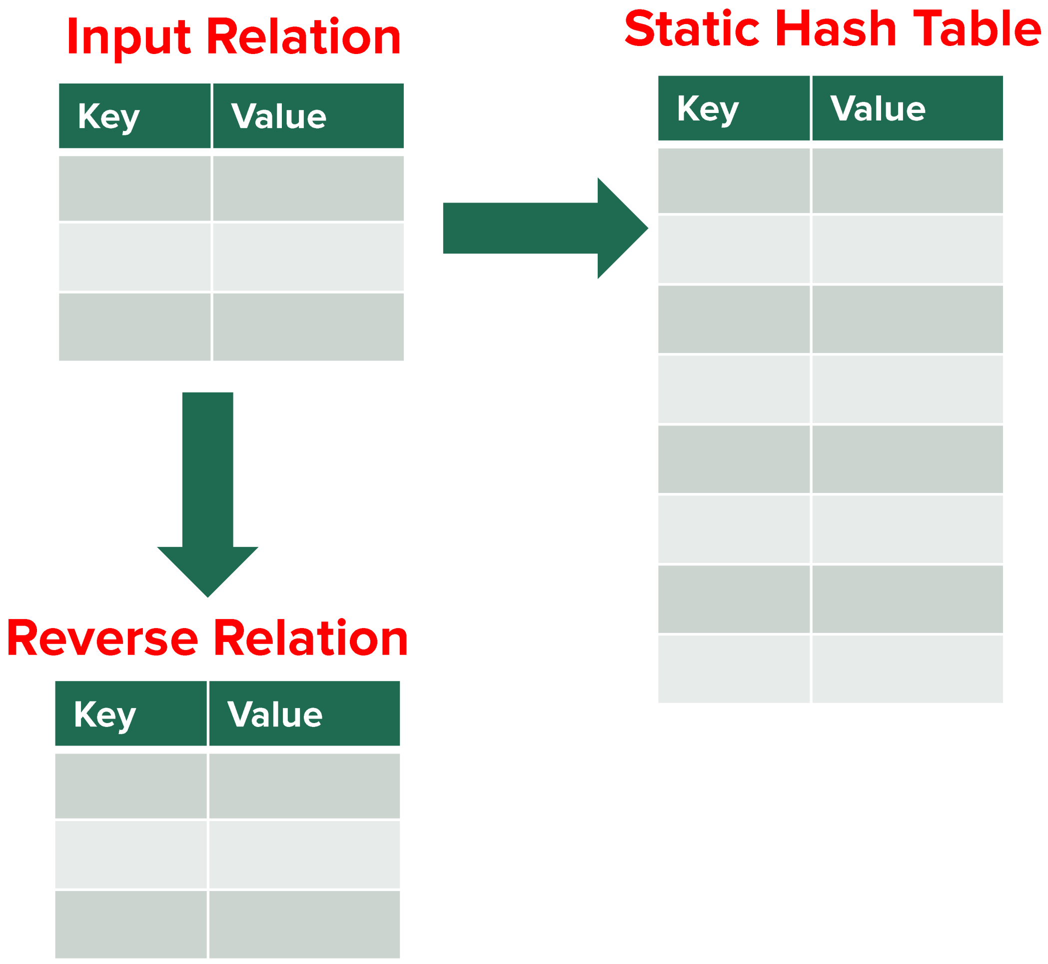
Off-the-shelf Data Structure



Both supports RA primitives (e.g. join, aggregation, rename, deduplication, and projection)

- 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.

Hash Join Initialization on GPU



Why Join is Important in RA?

COMBINE DATA FROM MULTIPLE TABLES

FIND PATTERNS IN DATA

CLEAN DATA

CREATE NEW DATA SETS

- Daniel Zinn, Haicheng Wu, Jin Wang, Molham Aref, and Sudhakar Yalamanchili. General-purpose join algorithms for large graph triangle listing on heterogeneous systems. In Proceedings of the 9th Annual Workshop on General Purpose Processing Using Graphics Processing Unit, pages 12–21, 2016.

Soufflé

- A variant of Datalog for static analysis using OpenMP
- State-of-the-art implementation for multi-core CPU systems with single-node
- Translates Datalog programs to optimized C++ programs
- Supports limited number of threads for task-level parallelism
- Cannot provide data parallelism

- Herbert Jordan, Bernhard Scholz, and Pavle Suboti´c. Souffl´e: On synthesis of program analyzers. InInternational Conference on Computer Aided Verification, pages 422–430. Springer, 2016.
- Thomas Gilray, Sidharth Kumar, and Kristopher Micinski. Compiling data-parallel datalog. InProceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction, CC 2021,page 23–35, New York, NY, USA, 2021. Association for Computing Machinery.

Parallel Join (Continue)

Design	Implement	Optimize
Consider partition, load balancing, communication	Challenging due to the uncertain output size	Efficient joins requires sorting or indexing

- Daniel Zinn, Haicheng Wu, Jin Wang, Molham Aref, and Sudhakar Yalamanchili. General-purpose join algorithms for large graph triangle listing on heterogeneous systems. In Proceedings of the 9th Annual Workshop on General Purpose Processing Using Graphics Processing Unit, pages 12–21, 2016.

Hybrid Join Algorithm

- Guo et al. proposed PHYJ: SMJ with HJ join
- Reduced host-to-device and device-to-host
- Fused data communication with GPU execution
- On a single GPU achieved up to 1.72X speedup
- Can handle skewed data
- No information on multiple GPUs or distributed systems

- Chengxin Guo, Hong Chen, Feng Zhang, and Cuiping Li. Parallel hybrid join algorithm on gpu. 2019IEEE 21st International Conference on High Performance Computing and Communications; IEEE17th International Conference on Smart City; IEEE 5th International Conference on Data Science andSystems (HPCC/SmartCity/DSS), pages 1572–1579, 2019.
- Hongzhi Wang, Ning Li, Zheng ke Wang, and Jianing Li. Gpu-based efficient join algorithms on hadoop.The Journal of Supercomputing, 77:292 – 321, 2020.

Join on GPUs: Benchmark

- Rui et al. assessed NINLJ, INLJ, SMJ, and HJ on modern GPU
- Modern GPUs can lead to 20X speedup VS 7X speedup of old GPUs
- Not suitable for HPC systems with multiple GPU environments
- New GPU architecture is introduced (Nvidia Hopper architecture)

- Bingsheng He, Ke Yang, Rui Fang, Mian Lu, Naga Govindaraju, Qiong Luo, and Pedro Sander.Relational joins on graphics processors. In Proceedings of the 2008 ACM SIGMOD internationalconference on Management of data, pages 511–524, 2008.
- Ran Rui, Hao Li, and Yi-Cheng Tu. Join algorithms on gpus: A revisit after seven years. In 2015 IEEEInternational Conference on Big Data (Big Data), pages 2541–2550. IEEE, 2015.
- Anne C Elster and Tor A Haugdahl. Nvidia hopper gpu and grace cpu highlights. Computing in Science& Engineering, 24(2):95–100, 2022.

Join on GPUs: LogiQL

- Wu et al. presents Red Fox high-performance accelerator core for LogiQL queries
- Outperforms multi-threaded CPU-based implementations
- Novel: multi-predicate join algorithm (worst-case optimal) on GPU
- Issue: deduplication of tuples and maintaining join result in sorted order

- Haicheng Wu, Gregory Diamos, Tim Sheard, Molham Aref, Sean Baxter, Michael Garland, andSudhakar Yalamanchili. Red fox: An execution environment for relational query processing on gpus. InProceedings of Annual IEEE/ACM International Symposium on Code Generation and Optimization,pages 44–54, 2014.
- Haicheng Wu. Acceleration and execution of relational queries using general purpose graphics processingunit (GPGPU). PhD thesis, Georgia Institute of Technology, 2015.

Join on GPUs: Relational Learning Framework

- Expedites rule coverage on GPUs for healthcare records data
- Outperforms 75% of applications over multi-core CPU systems
- Duplicate tuples not efficiently managed and GPU memory overflows

- Carlos Alberto Mart´ınez-Angeles, Haicheng Wu, Inˆes Dutra, V´ıtor Santos Costa, and Jorge BuenabadCh´avez. Relational learning with gpus: Accelerating rule coverage. International Journal of ParallelProgramming, 44(3):663–685, 2016

Join on GPUs: Control Flow Analysis (CFA)

Parallel functional CFA encoded in Datalog utilizes RA as the foundation on GPU

Extended Red Fox combining GPU parallelism with multi-node multi-core HPC

Proposed partitioned global address space (PGAS) programming model

- THOMAS GILRAY and SIDHARTH KUMAR. Toward parallel cfa with datalog, mpi, and cuda. InScheme and Functional Programming Workshop, 2017.

Join on GPUs: WarpDrive

- Jünger et al. presented a single-node multi-GPU hashing for hashjoin
- Attained better memory coalescing
- Hashtable insertion rate:
  - 1.4B keys/sec (single GPU)
  - 4.3B keys/sec (4 GPUs)
- 32 bit keys only with no deduplication
- Incremental study: WarpCore supports 64 bit keys

- Daniel J¨unger, Christian Hundt, and Bertil Schmidt. Warpdrive: Massively parallel hashing on multigpu nodes. In 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pages441–450. IEEE, 2018.
- Daniel J¨unger, Robin Kobus, Andr´e M¨uller, Christian Hundt, Kai Xu, Weiguo Liu, and Bertil Schmidt.Warpcore: A library for fast hash tables on gpus. In 2020 IEEE 27th International Conference on HighPerformance Computing, Data, and Analytics (HiPC), pages 11–20, 2020.