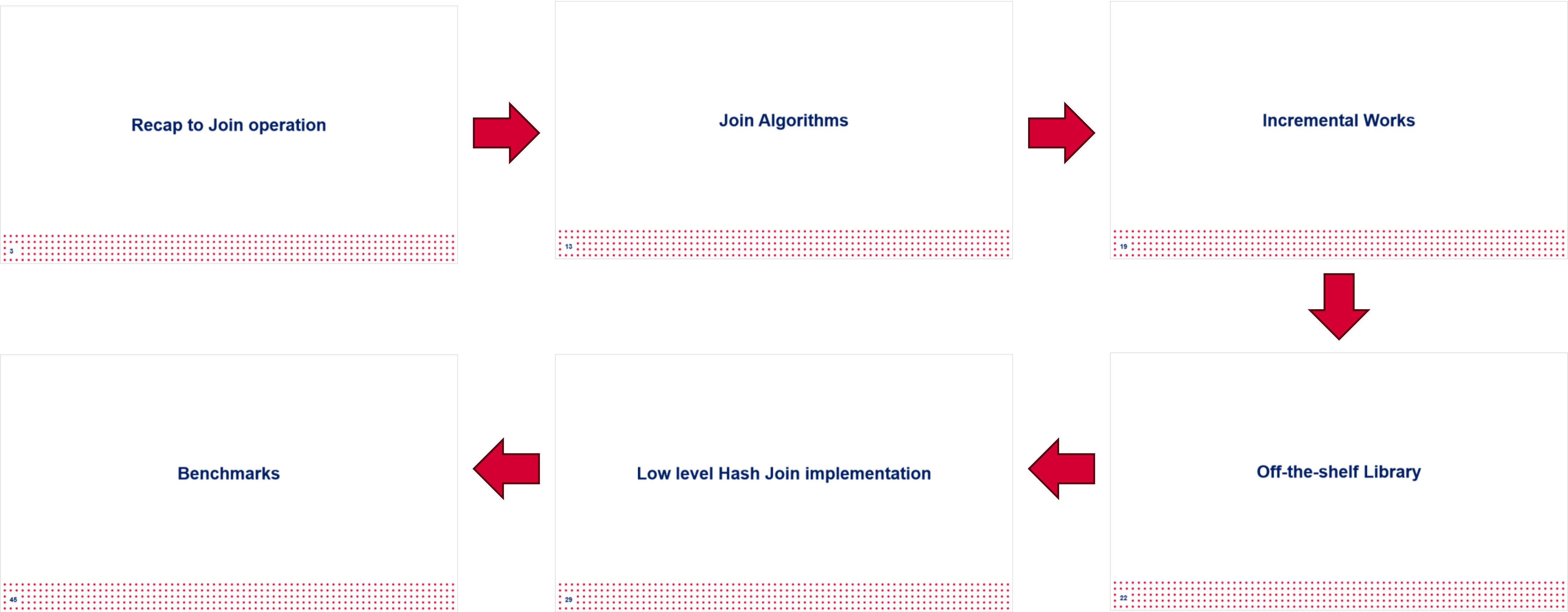
10/23/2023

High Performance Joins

Ahmedur Rahman Shovon
PhD student
Department of Computer Science
Email: ashov@uic.edu
Website: arshovon.com

UNIVERSITY OF ILLINOIS CHICAGO

Roadmap


Recap to Join operation
3
Join Algorithms
13
Incremental Works
19
Off-the-shelf Library
22
Low level Hash Join implementation
29
Benchmarks
45

Recap to Join operation

Recap to Relational Data

- Relation: 2-dimensional structure
- Attribute: Represents characteristics
- Row: Represents unique record
- Join (⋈): Combines data from relations
- Projection (Π): Select specific columns

Relation

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

Tuple (Row)

Attribute (Column)

- 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.

Why Join is Important?

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.

Example of Natural Join ⋈

User (Outer Relation)

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

⋈

Order (Inner Relation)

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

Example of Natural Join ⋈

User (Outer Relation)

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

Order (Inner Relation)

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

User $\bowtie$ **Order**

UserID	UserName	UserEmail	OrderTotal	Items
101	Alice	alice@example.com	12.11	1

Example of Natural Join ⋈

User (Outer Relation)

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

Order (Inner Relation)

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

User ⋈ Order

UserID	UserName	UserEmail	OrderTotal	Items
101	Alice	alice@example.com	12.11	1
102	Bob	bob@example.com	145.66	3

Example of Natural Join ⋈

User (Outer Relation)

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

Order (Inner Relation)

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

User $\bowtie$ **Order**

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

Example of Natural Join ⋈

User (Outer Relation)

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

Order (Inner Relation)

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

User ⋈ Order

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

Example of Natural Join ⋈

User (Outer Relation)

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

Order (Inner Relation)

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

User $\bowtie$ **Order**

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

Duplicates on Join Result

User ⋈ Order

UserID	UserName	UserEmail	OrderTotal	Items
101	Alice	alice@example.com	12.11	1
102	Bob	bob@example.com	145.66	3
103	Eve	eve@example.com	26.69	2
103	Eve	eve@example.com	44.00	2

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

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

Join Algorithms

Join Algorithms

Common Algorithms		
Nested Loop Join (NLJ)	Sort-Merge Join (SMJ)	Hash Join (HJ)

NLJ is suitable for small dataset
SMJ is efficient with pre-sorted data
HJ works on unsorted datasets through hash-based partitioning

- 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 international conference 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 IEEE International Conference on Big Data (Big Data), pages 2541–2550. IEEE, 2015.

Nested Loop Join ⋈

User (Outer Relation)

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

Order (Inner Relation)

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

User ⋈ Order

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

Time complexity is O(N * M)

Example of Sort Merge Join ⋈

User (Outer Relation)

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

Sort

User (Sorted)

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

Order (Inner Relation)

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

Sort

Order (Sorted)

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

Example of Sort Merge Join ⋈

User (Sorted)

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

Merge

Order (Sorted)

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	12.11	1
102	Bob	bob@example.com	145.66	3
103	Eve	eve@example.com	25.69	2
103	Eve	eve@example.com	44.00	2

Time complexity: O(N * log(N)) + O(M * log(M)) (sorting) + O(N + M) (merging)

Example of Hash Join ⋈

User (HashTable)

User (Outer Relation)

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

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

Order (Inner Relation)

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

Build

Probe

User ⋈ Order

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

Time complexity is O(M) + O(N) (building and probing hash table)

Incremental Works

Our efforts on High Performance Relational Algebra

cuDF	GPUJoin	GDLog	mnmgJoin
Started with CPU and GPU based Python libraries	Relational Algebra (RA) operations using CUDA	CUDA based SIMD API for deductive analytics	Multi node multi GPU based engine
Compare GPU capabilities for iterative join operations	High performance GPU hashtable for iterative RA	Novel Hash-Indexed Sorted Array data structure	CUDA + MPI based design (Work in progress)

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

Baseline Engine (Soufflé)

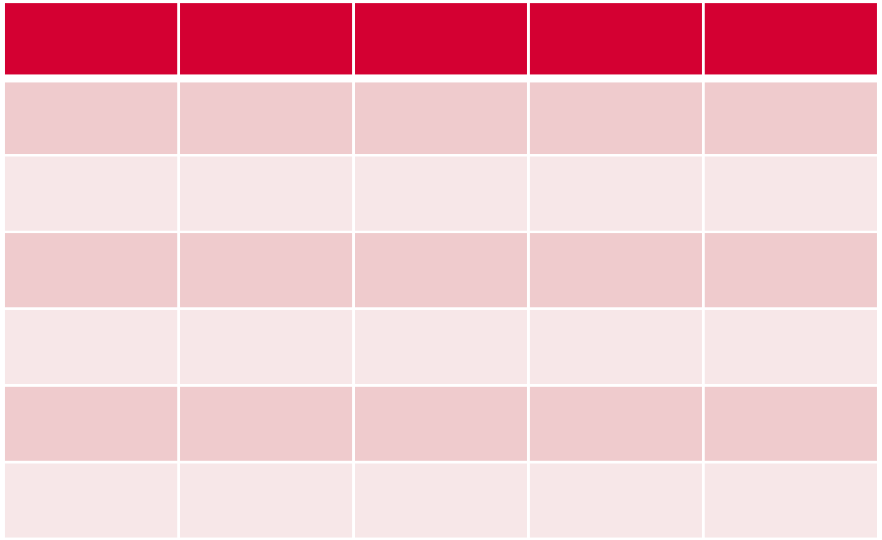
- A state-of-the art in-memory engine
- Uses CPU-based multi-core system for parallel execution of RA operations

- 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. Springer, 422–430

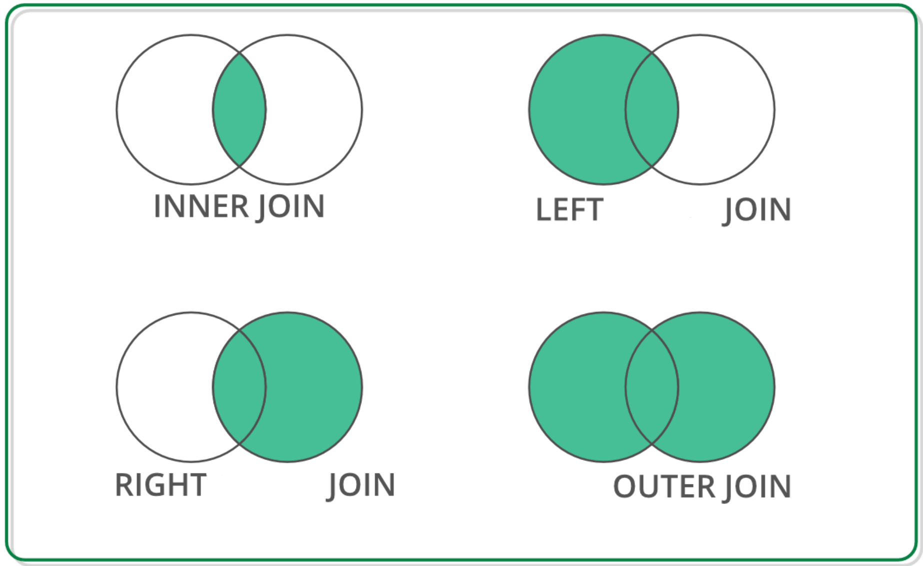
Off-the-shelf Library

Off-the-shelf Data Structure for Join Operation

DataFrame: 2D labeled tabular data structure



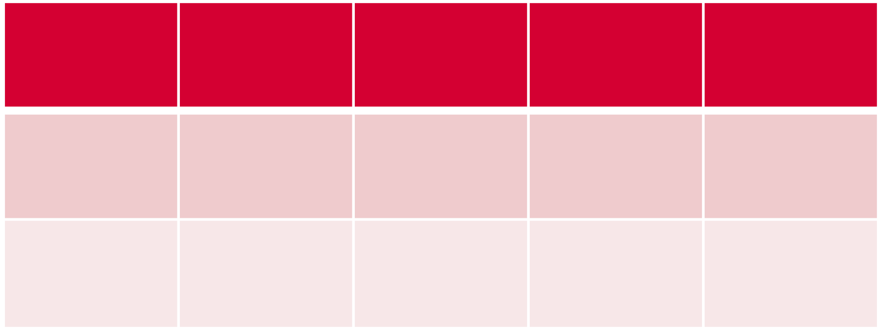
DataFrame has RA primitives APIs



- 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

Off-the-shelf Python Libraries

DataFrame: 2D labeled tabular data structure



CPU

GPU
(NVIDIA cuDF)

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)

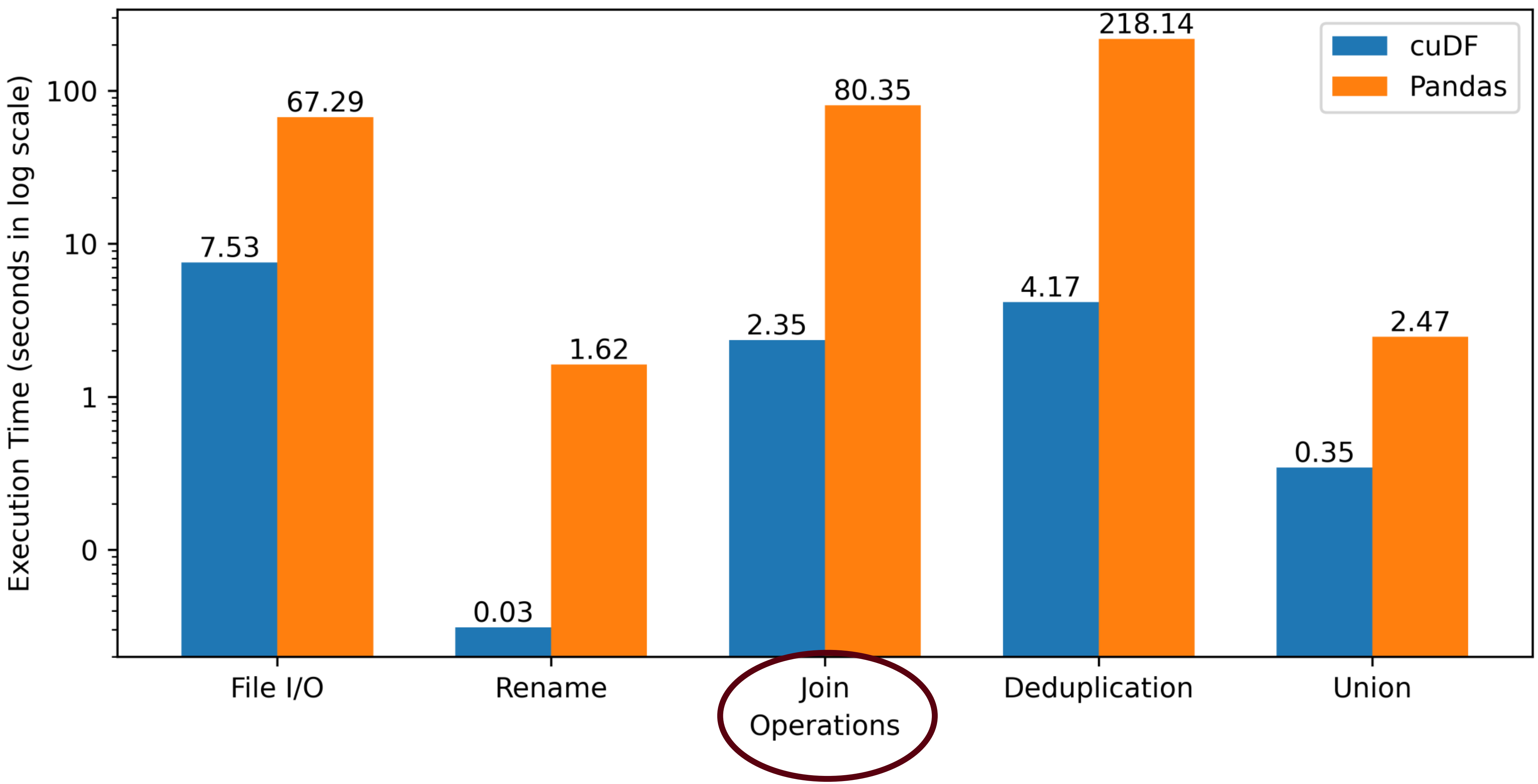
```python
import pandas as pd
import cudf

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

Performance Improvement of using GPU



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

DataFrame Based Join Operations

✓ Advantages

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

- 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

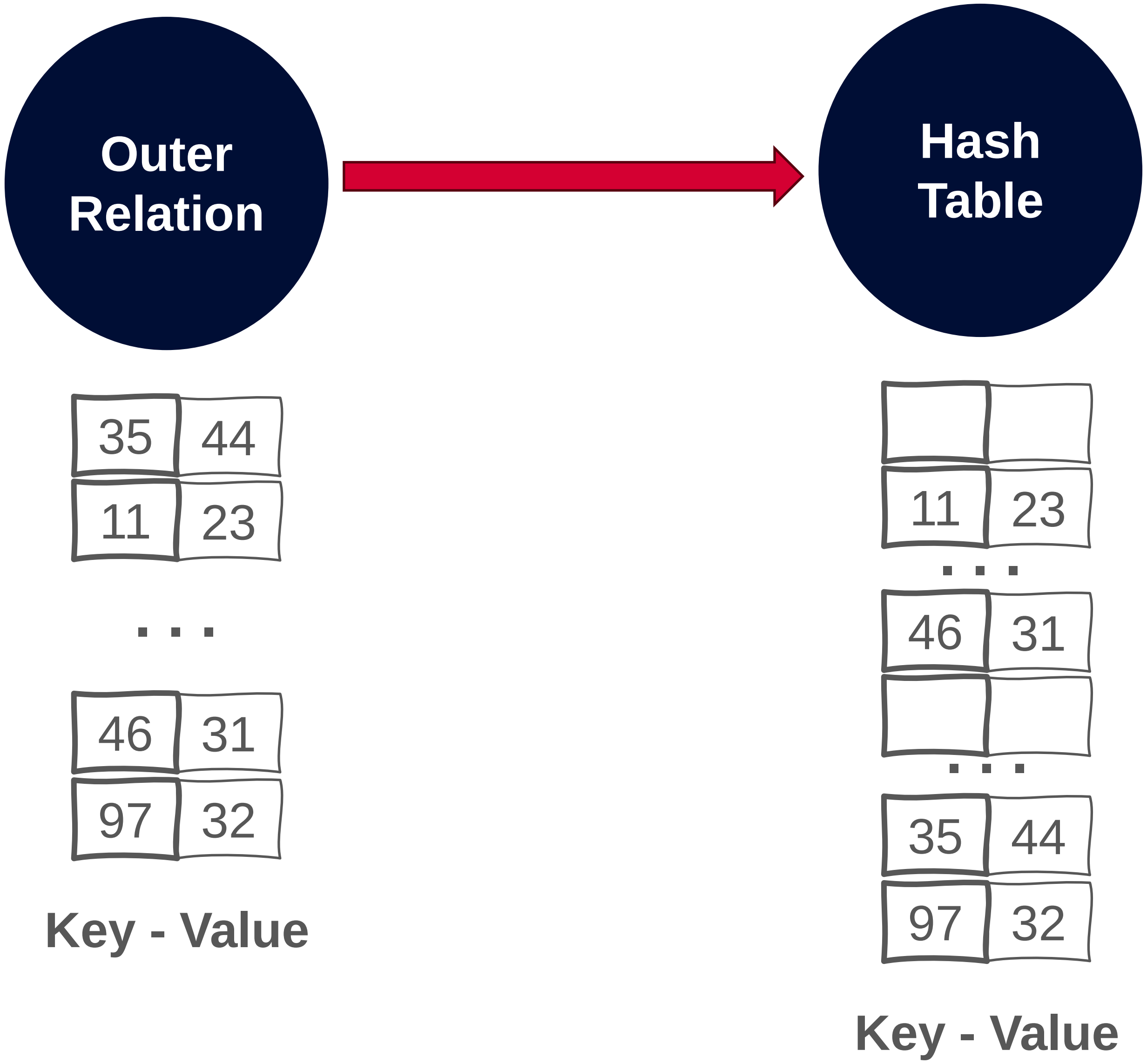
Improvement Opportunity

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

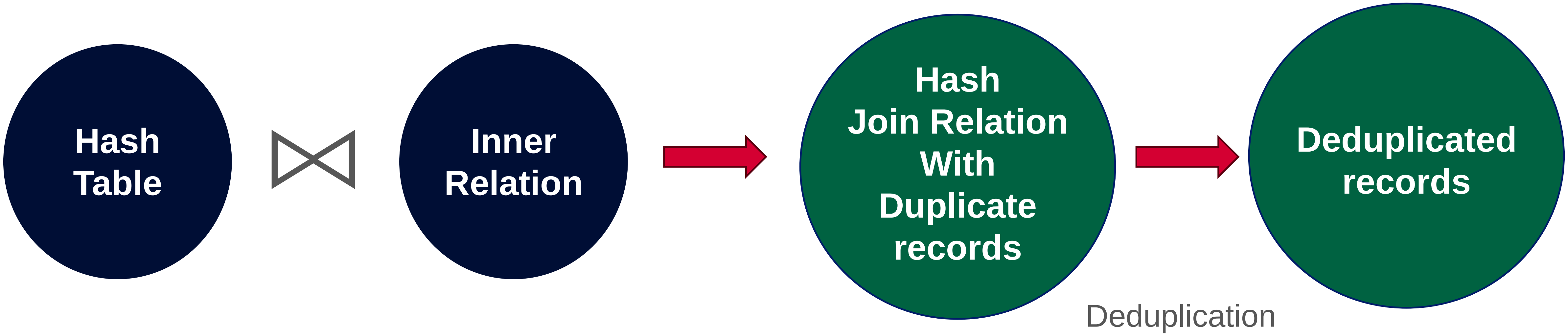
- A. R. Shovon, L. R. Dyken, O. Green, T. Gilray and S. Kumar, "Accelerating Datalog applications with cuDF," 2022 IEEE/ACM Workshop on Irregular Applications: Architectures and Algorithms (IA3), Dallas, TX, USA, 2022, pp. 41-45
- Green, O., Du, Z., Patel, S., Xie, Z., Liu, H., & Bader, D. A. (2021,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

Low level Hash Join implementation

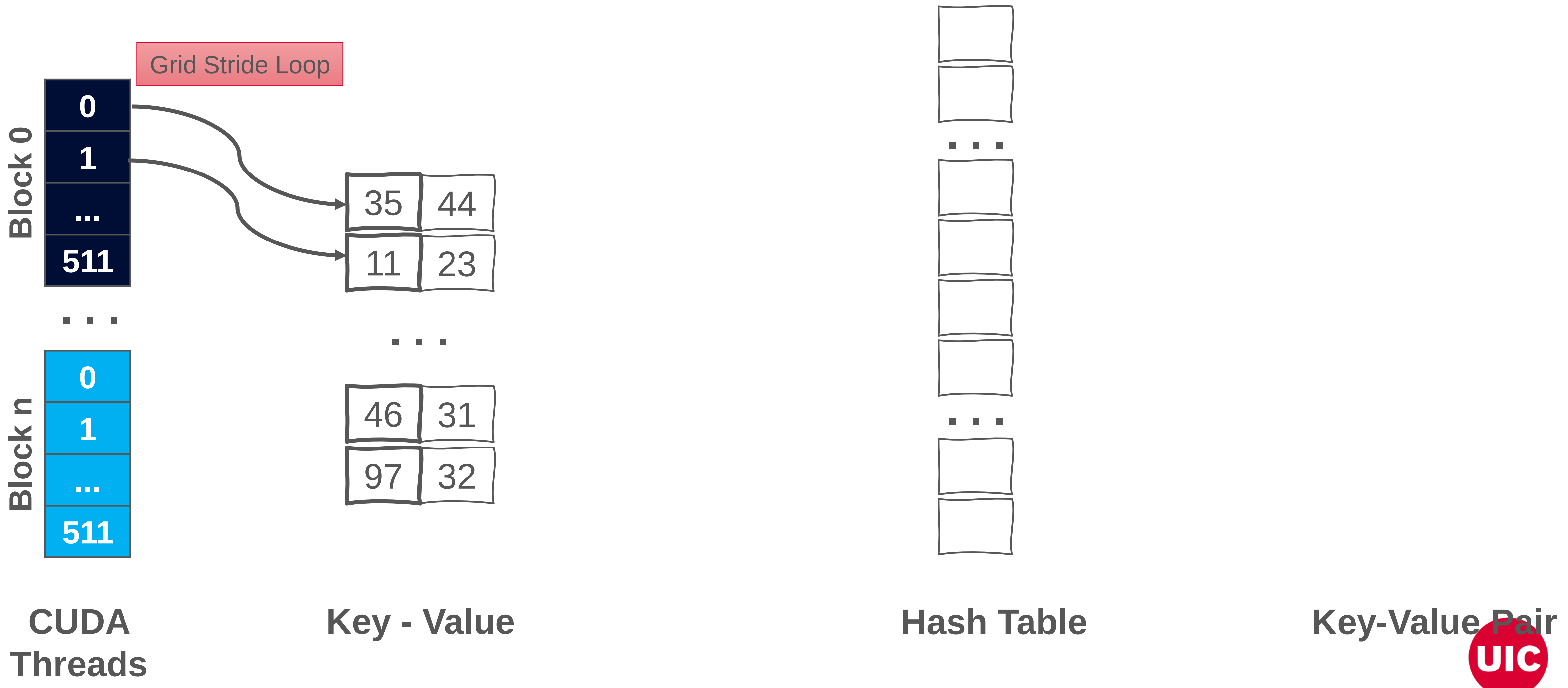
Hash Join Process



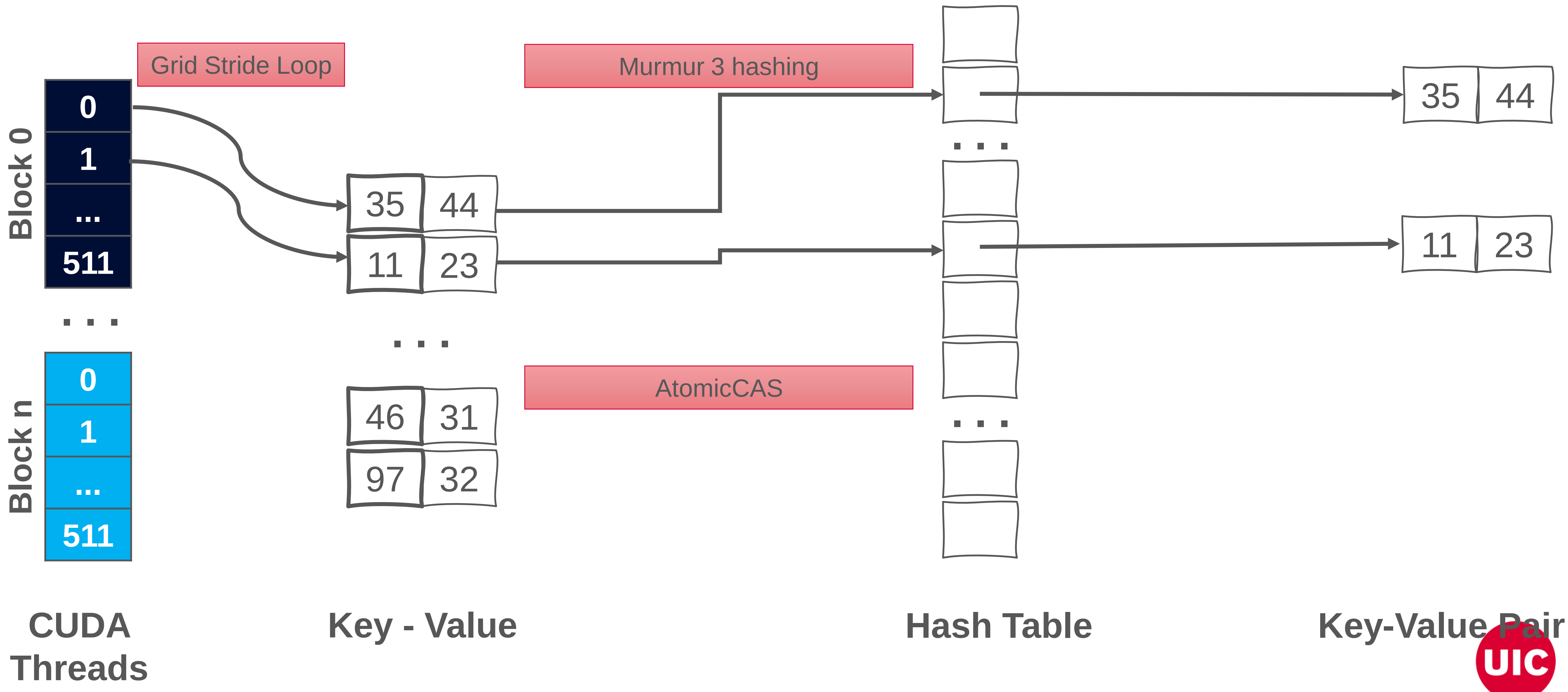
Hash Join Process



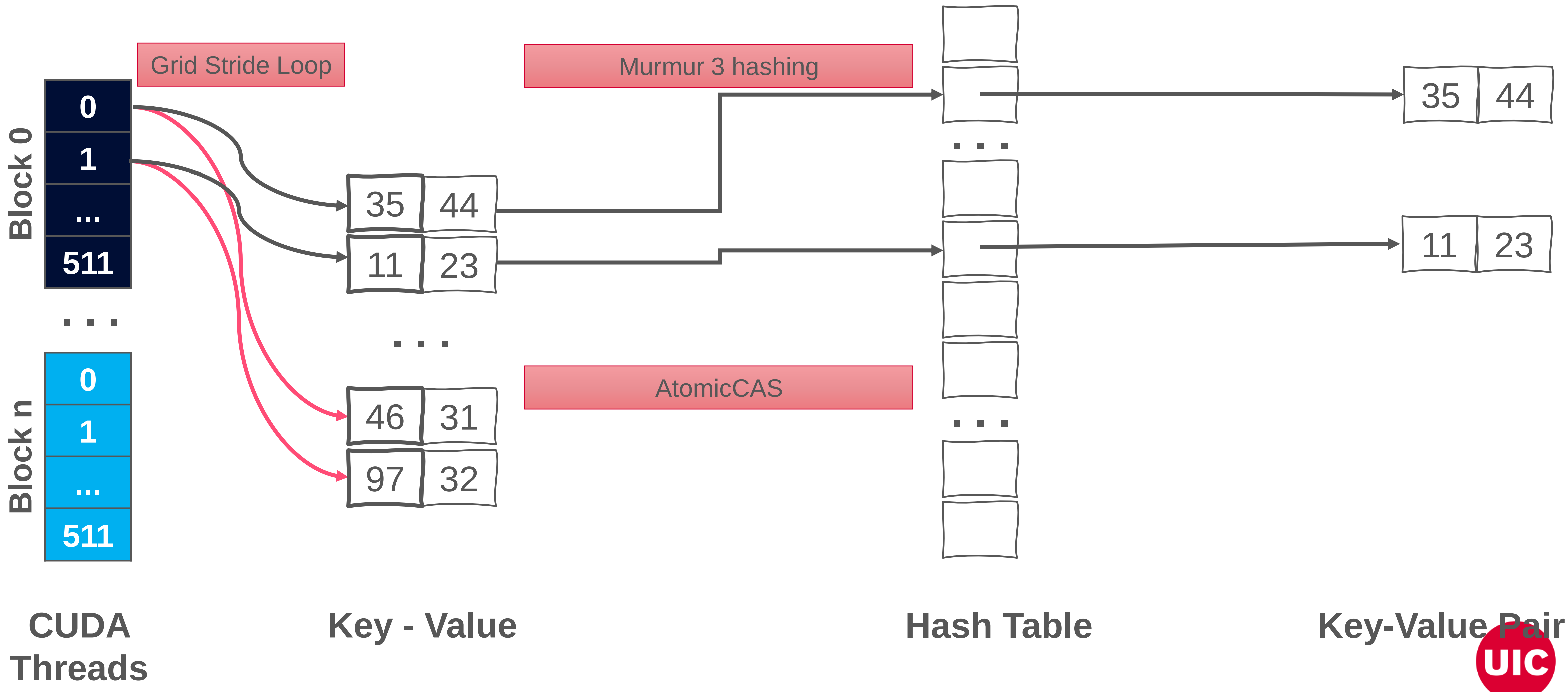
Hash Table (Open Addressing, Linear Probing)



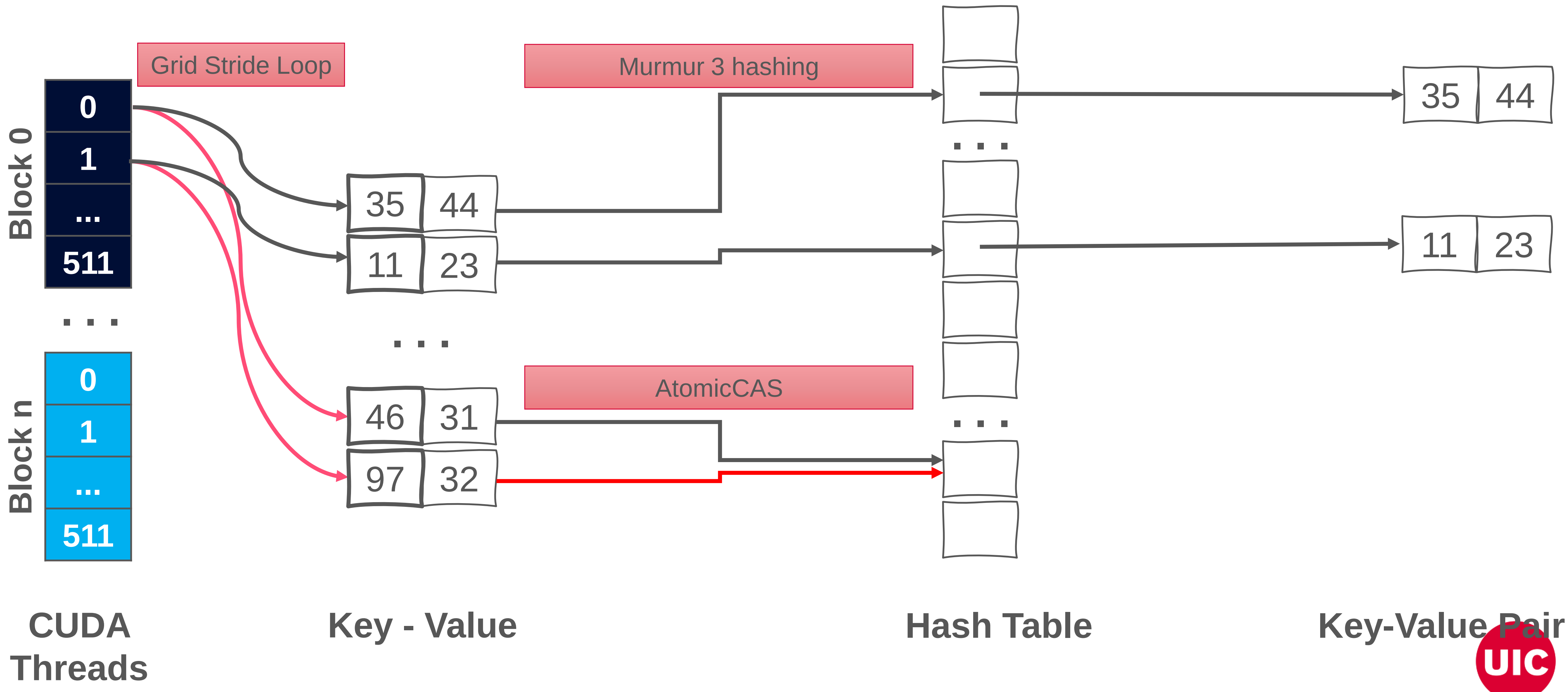
Hash Table (Open Addressing, Linear Probing)



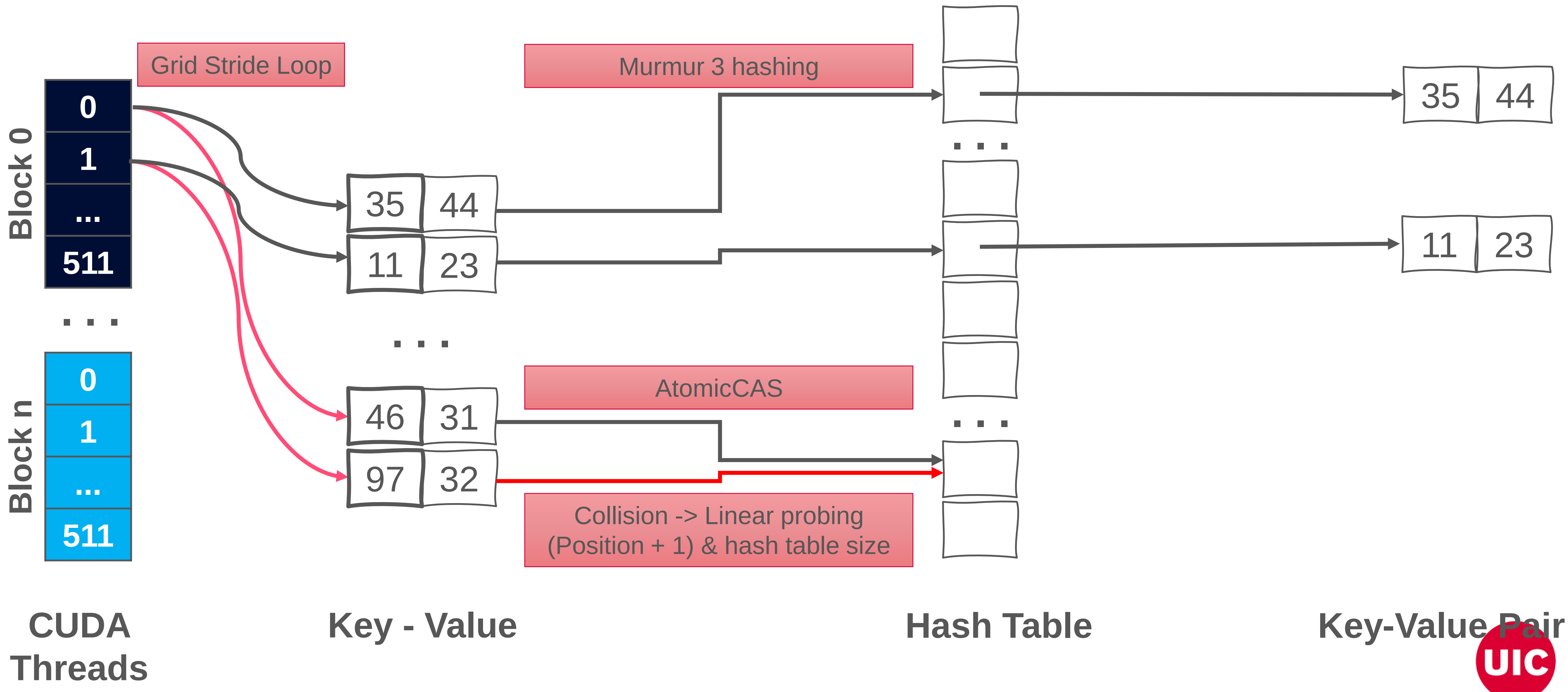
Hash Table (Open Addressing, Linear Probing)



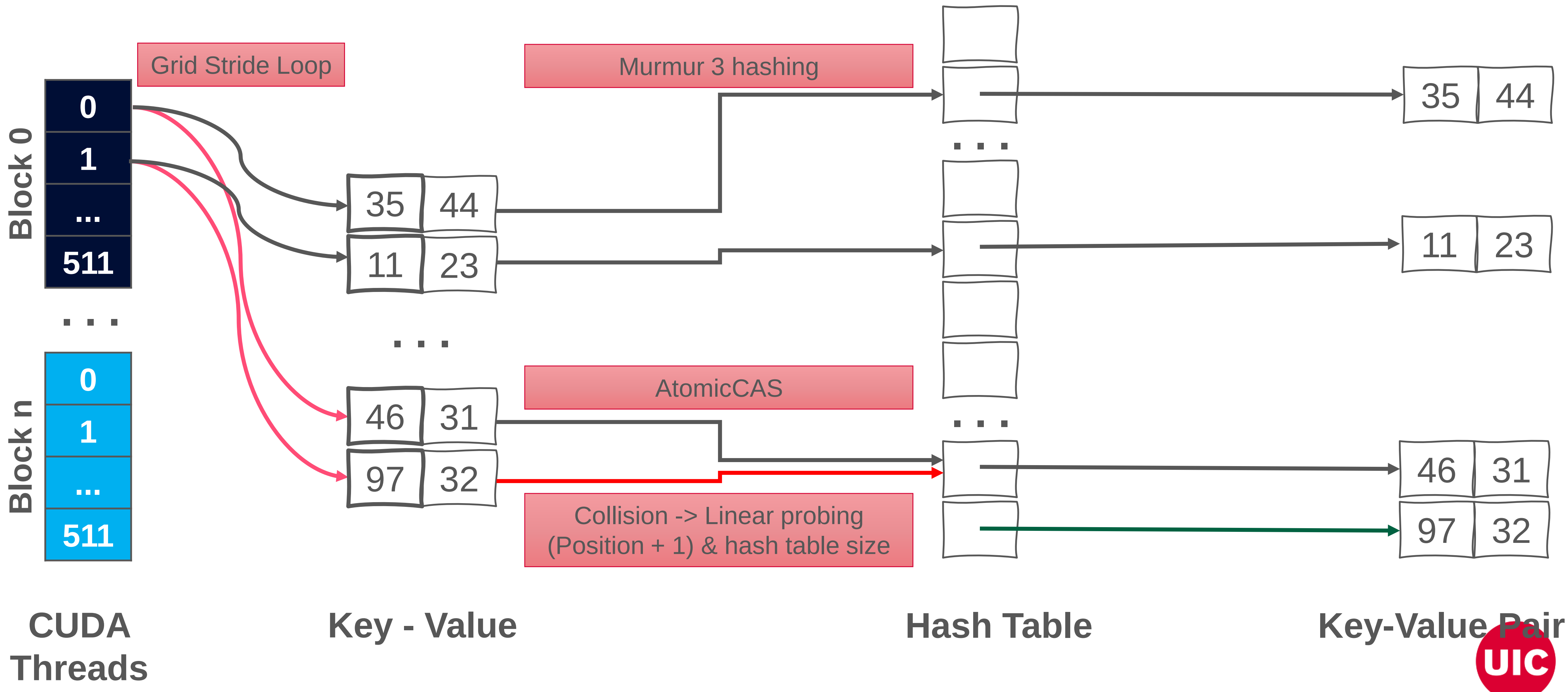
Hash Table (Open Addressing, Linear Probing)



Hash Table (Open Addressing, Linear Probing)

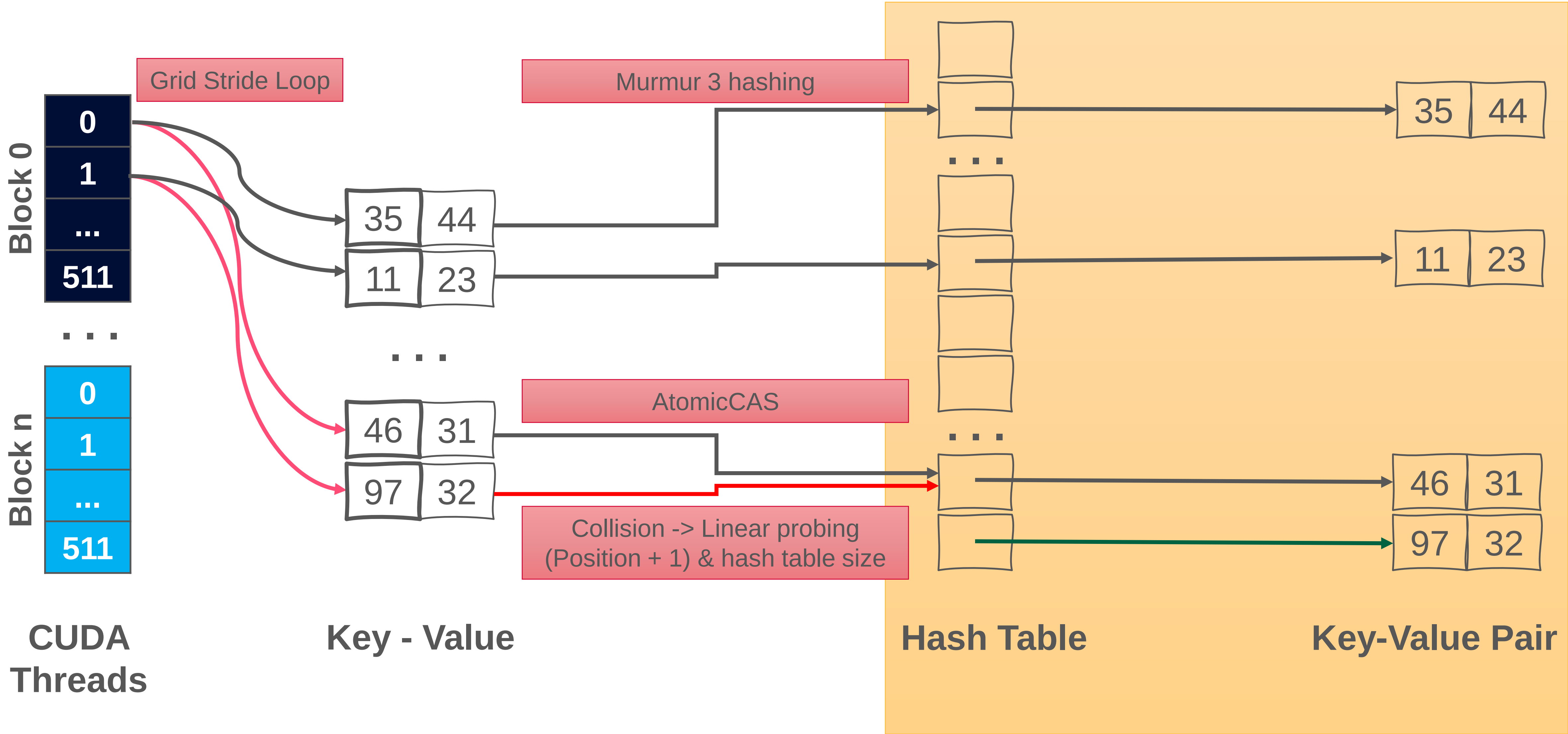


Hash Table (Open Addressing, Linear Probing)



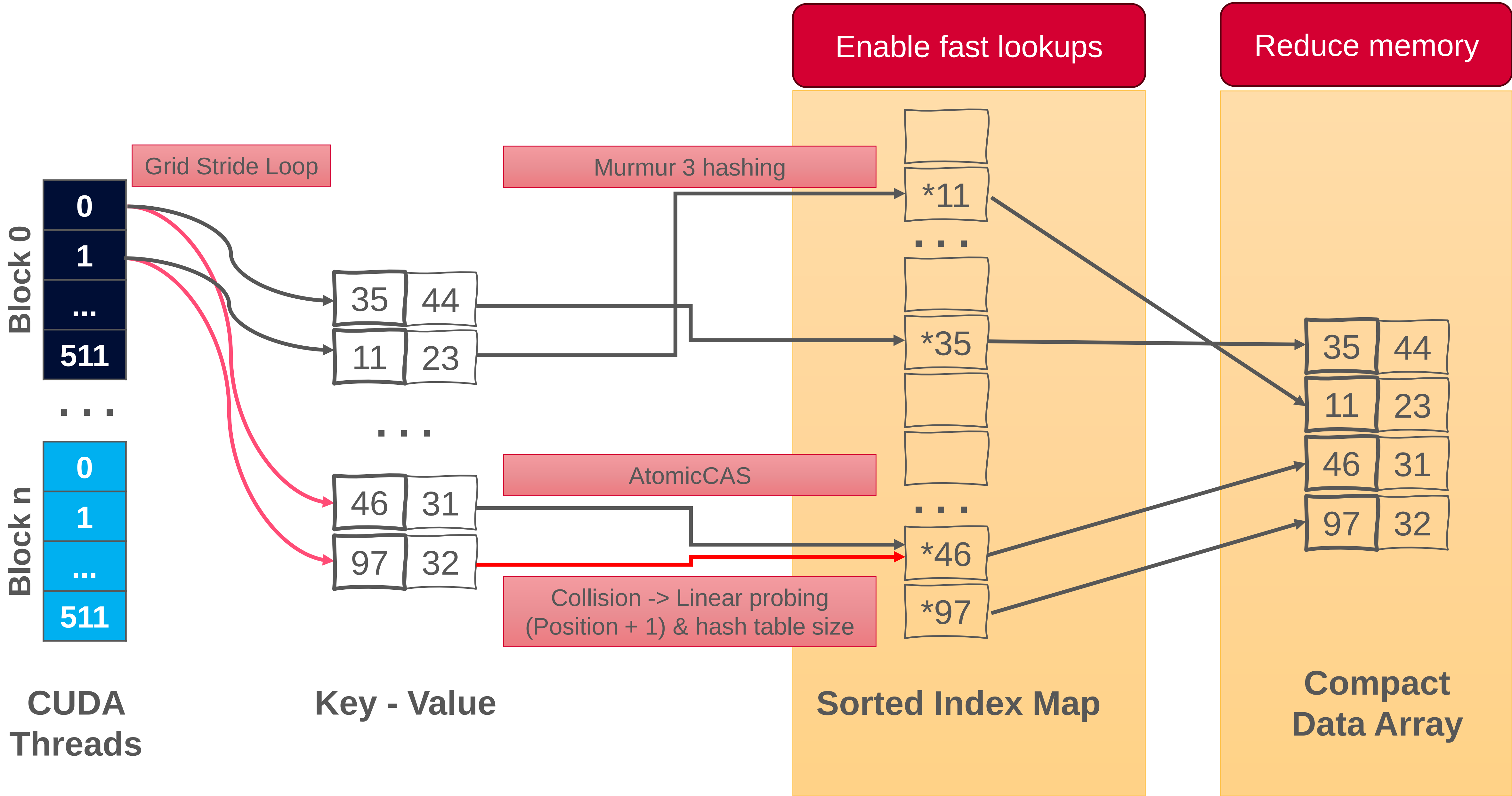
Hash Table (Open Addressing, Linear Probing)

Stores key-value pairs in sparse space
Uses extra memory



How can we improve this?

Hash Indexed Sorted Array (HISA)



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

Performing Hash Join on GPU

Static Hash Table

Key	Value

$\bowtie$

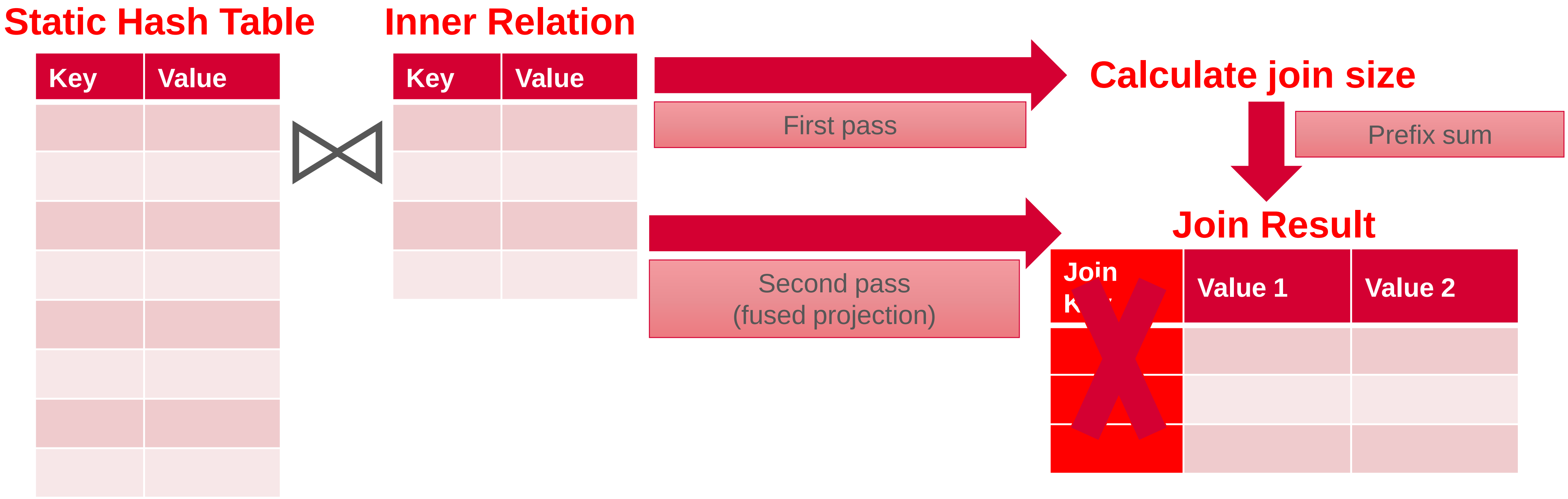
Inner Relation

Key	Value

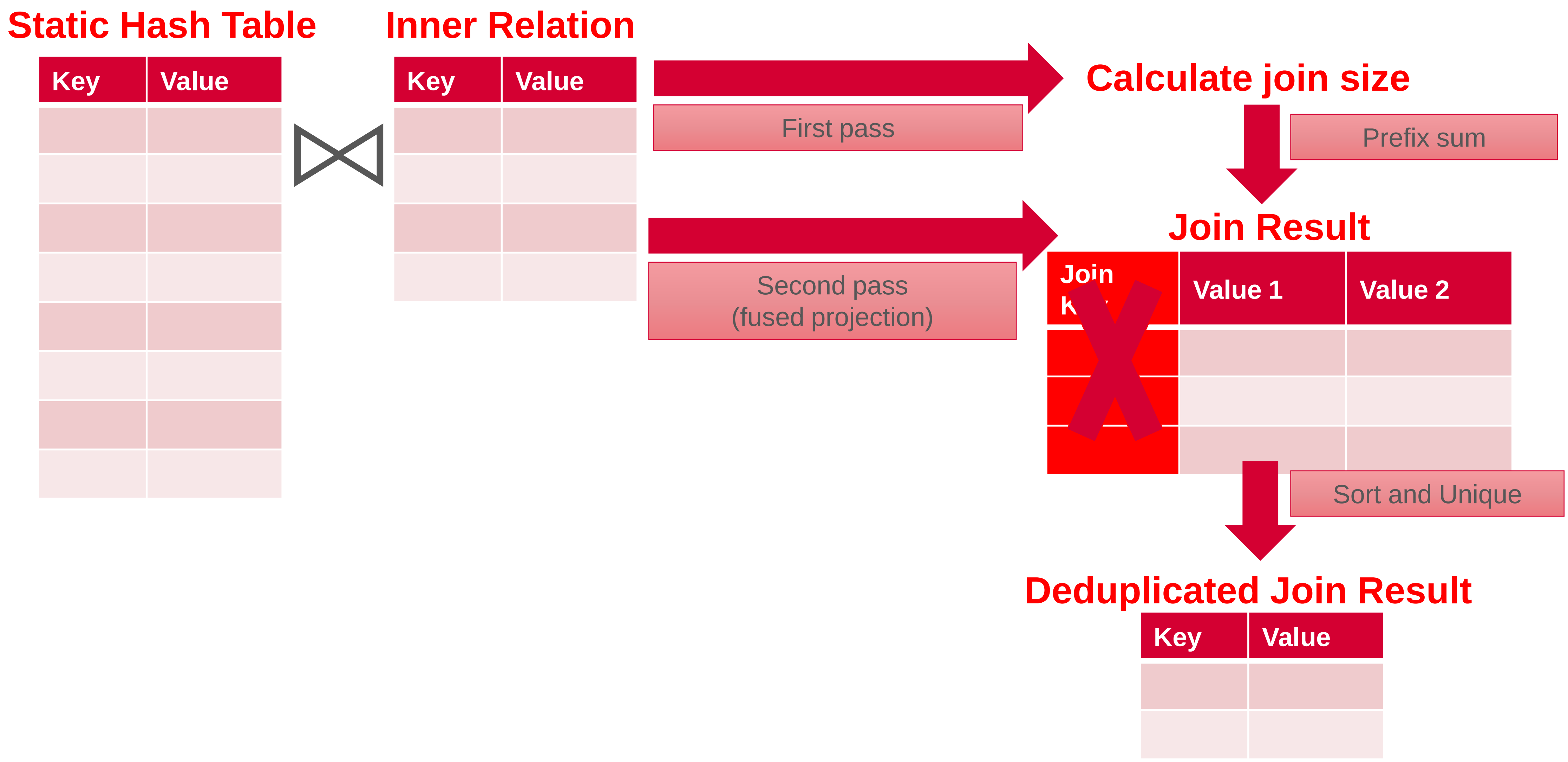
First pass

Calculate join size

Performing Hash Join on GPU

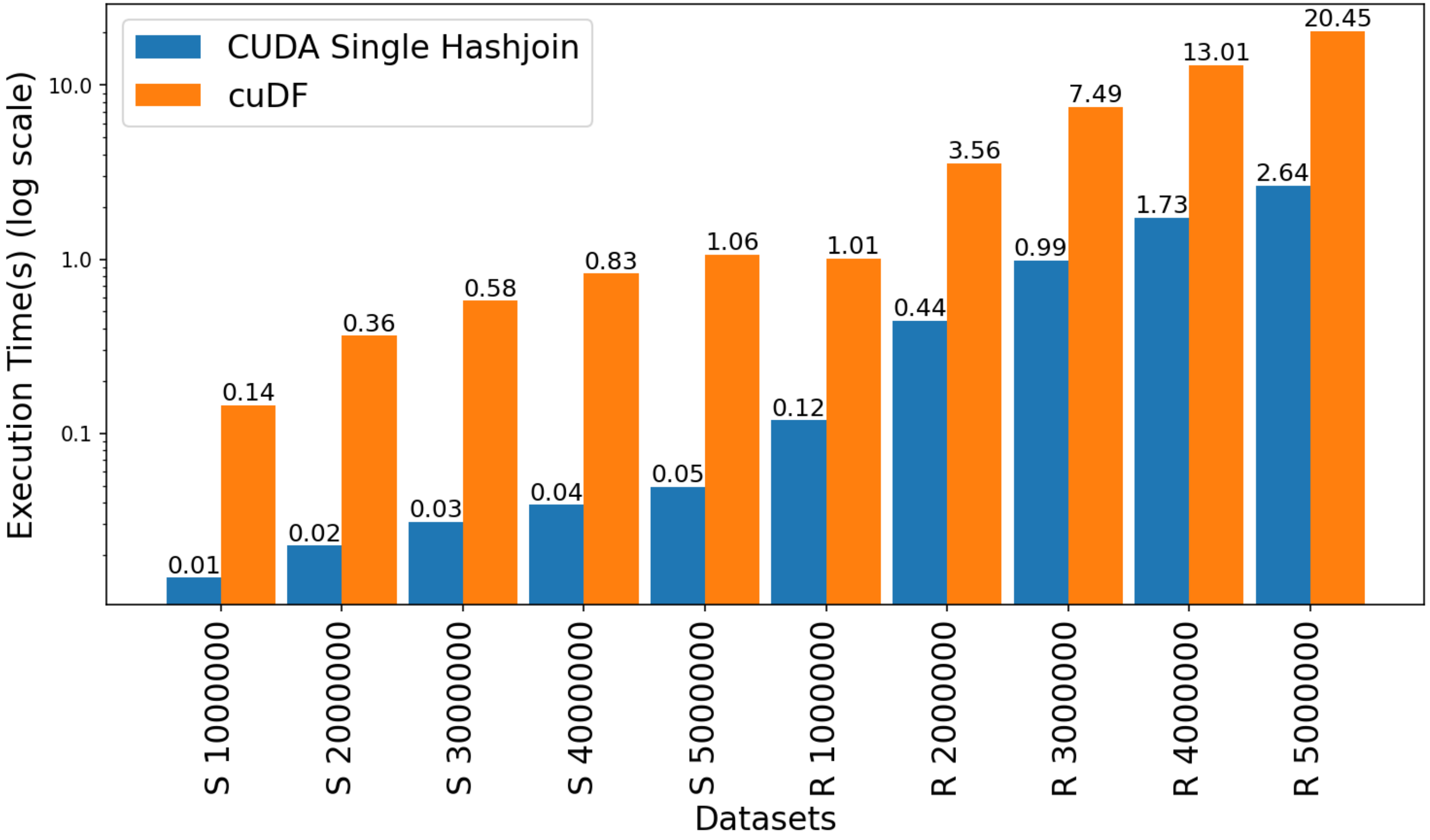


Performing Hash Join on GPU



Benchmarks

Join Performance Comparison: GPUJoin vs cuDF

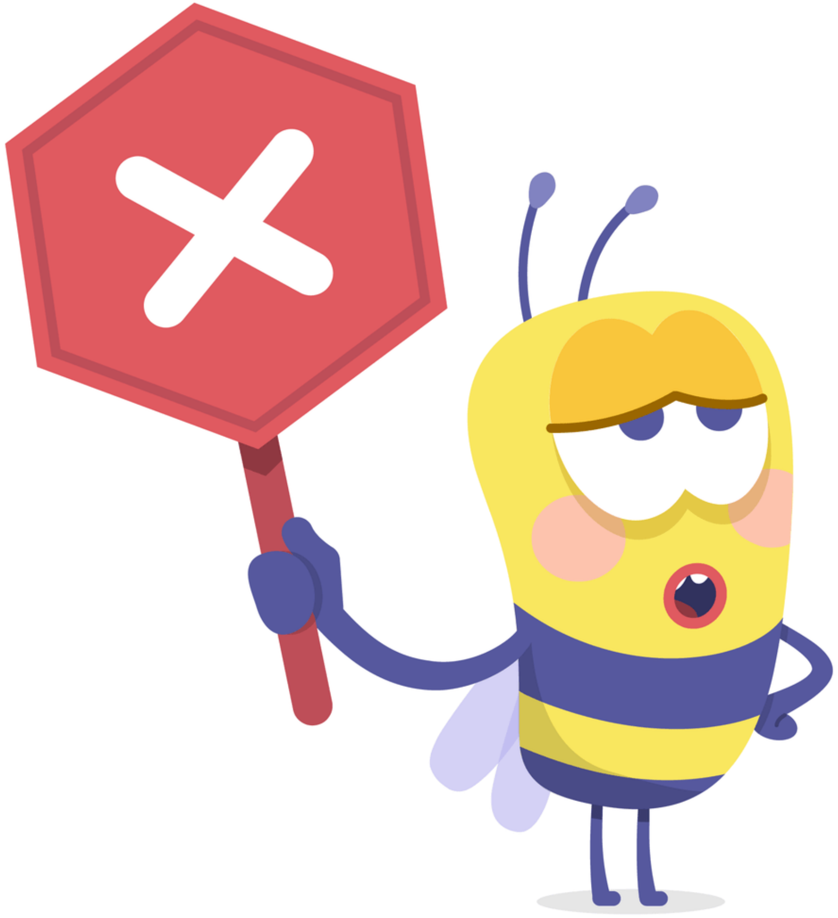


- Leadership Computing Facility, A. (2022). Argonne Leadership Computing Facility. Theta GPU Nodes. URL: https://www.alcf.anl.gov/support-center/theta-gpu-nodes

Performance Enhancement (Reachability)

Dataset name	*Reach* edges	Time (s) GDLOG	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

Limitations

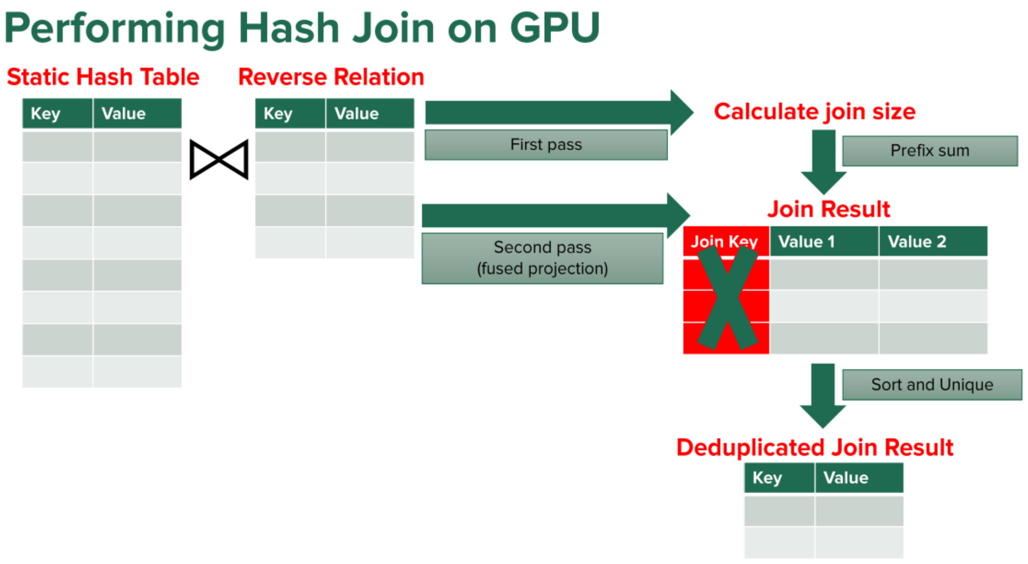


Limited to a single GPU that dictates scaling by available VRAM on the GPU

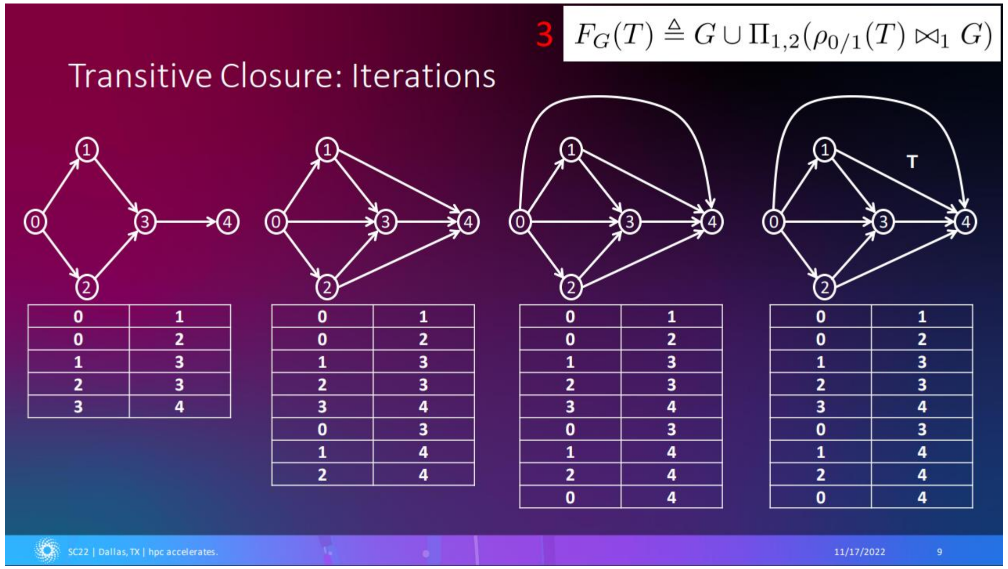
Memory overflow error for larger graphs

Publications

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).



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.



Future Work

- Continue working on developing multi-node multi-GPU backend for Datalog
- Compare performance between CUDA + MPI backend with CUDA aware MPI backend
- Design GPU benchmarking techniques for iterative RA

Thank You

ashov@uic.edu

Declarative Analytics on Heterogeneous Exascale Systems

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

User

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

WHAT

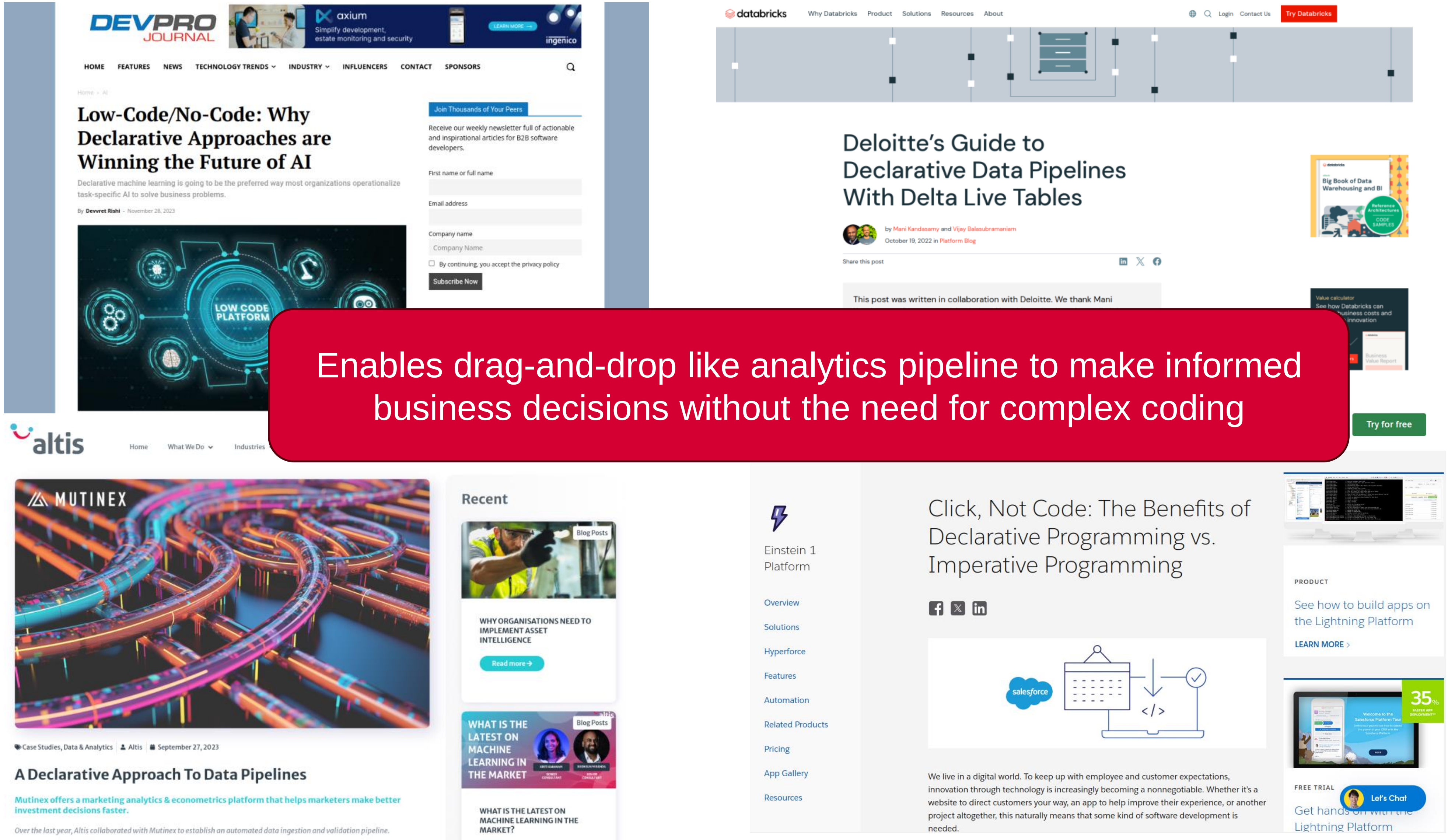
SELECT **UserID** FROM **User** WHERE **Country** = 'USA';

~~HOW~~

Advanced approach: Logic programming (Datalog)

- Makrynioti, N., & Vassalos, V. (2019). Declarative data analytics: A survey. IEEE Transactions on Knowledge and Data Engineering, 33(6), 2392-2411.
- Salesforce. (2024). Click, Not Code: The Benefits of Declarative Programming vs. Imperative Programming retrieved from https://www.salesforce.com/products/platform/best-practices/declarative-programming-vs-imperative-programming/ on 01/24/2026

Declarative Analytics on Heterogeneous Exascale Systems



Enables drag-and-drop like analytics pipeline to make informed business decisions without the need for complex coding

- Makrynioti, N., & Vassalos, V. (2019). Declarative data analytics: A survey. IEEE Transactions on Knowledge and Data Engineering, 33(6), 2392-2411.
- Salesforce. (2024). Click, Not Code: The Benefits of Declarative Programming vs. Imperative Programming retrieved from https://www.salesforce.com/products/platform/best-practices/declarative-programming-vs-imperative-programming/ on 01/24/2026