01/26/2024

Declarative Analytics on Heterogeneous Exascale Systems

Ahmedur Rahman Shovon

WCP Committee:

Dr. Michael Papka (Chair)

Dr. Sidharth Kumar (Advisor)

Dr. Pedram Rooshenas

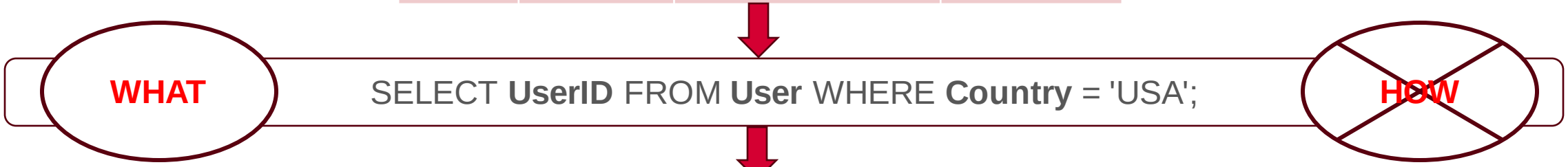
UIC UNIVERSITY OF ILLINOIS CHICAGO

Declarative Analytics on Heterogeneous Exascale Systems

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

User

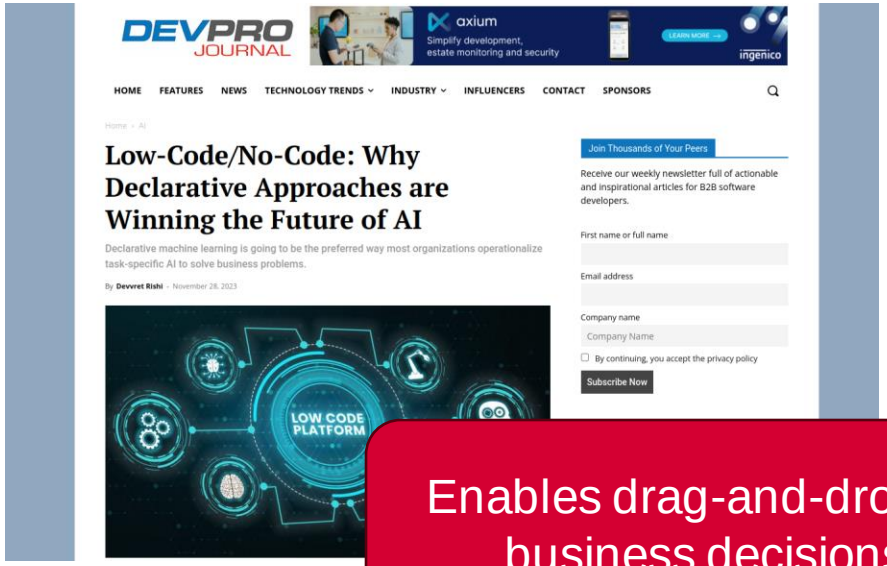
UserID	UserName	UserEmail	Country
101	Alice	alice@example.com	USA
102	Bob	bob@example.com	USA
103	Eve	eve@example.com	Australia

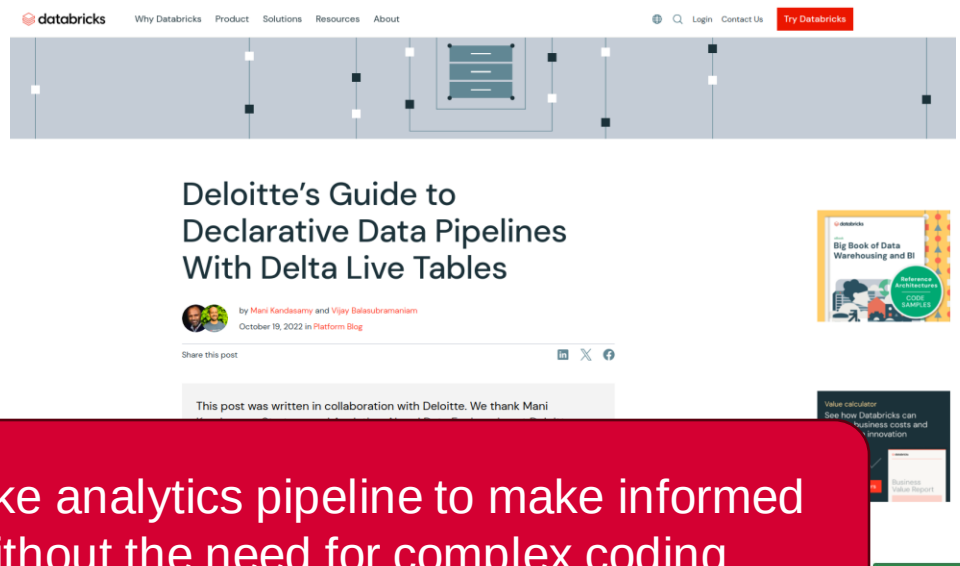


Advanced approach: Logic programming (Datalog)

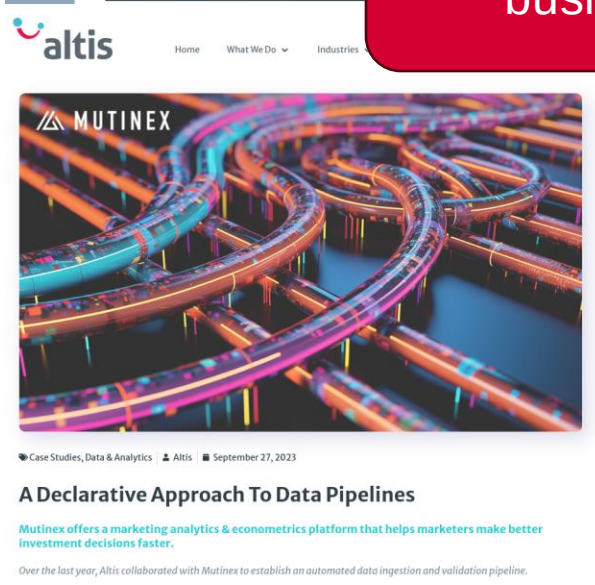
- Makrynioti, N., & Vassalos, V. (2019). Declarative data analytics: A survey. IEEE Transactions on Knowledge and Data Engineering, 33(6), 2392-2411.
- Salesforce. (2024). Click, Not Code: The Benefits of Declarative Programming vs. Imperative Programming retrieved from https://www.salesforce.com/products/platform/best-practices/declarative-programming-vs-impera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

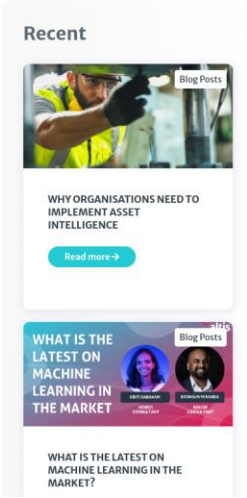
Declarative Analytics on Heterogeneous Exascale Systems

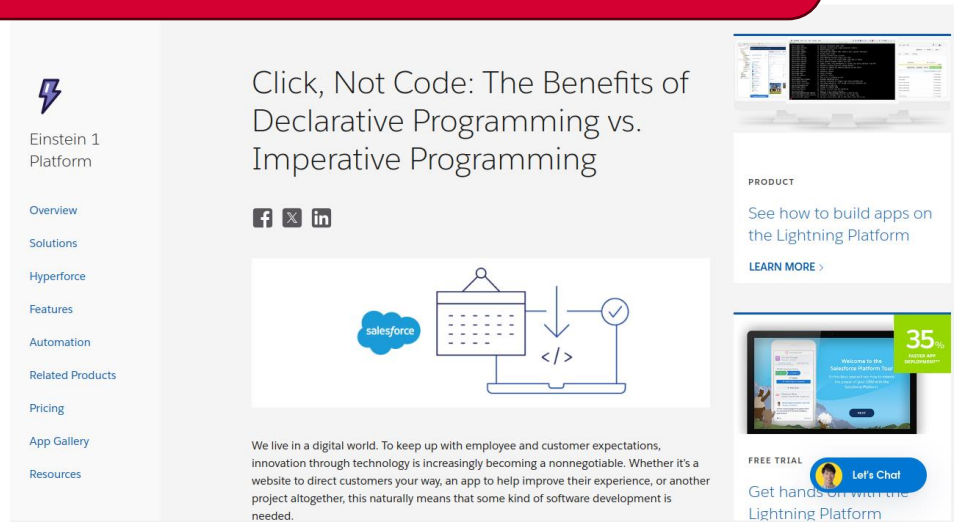




Enables drag-and-drop like analytics pipeline to make informed business decisions without the need for complex coding







- 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

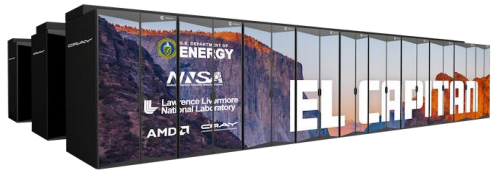
Table of Contents

What is Exascale?

Ability of a computer system to perform 1 Exaflop or more

1 Exaflop = 1 quintillion flop (10^{18})
or 10^9 Gigaflop

Capacity?
Perform as many calculations as about 50 million laptops!
Speed?
A trillion times faster than a PC!



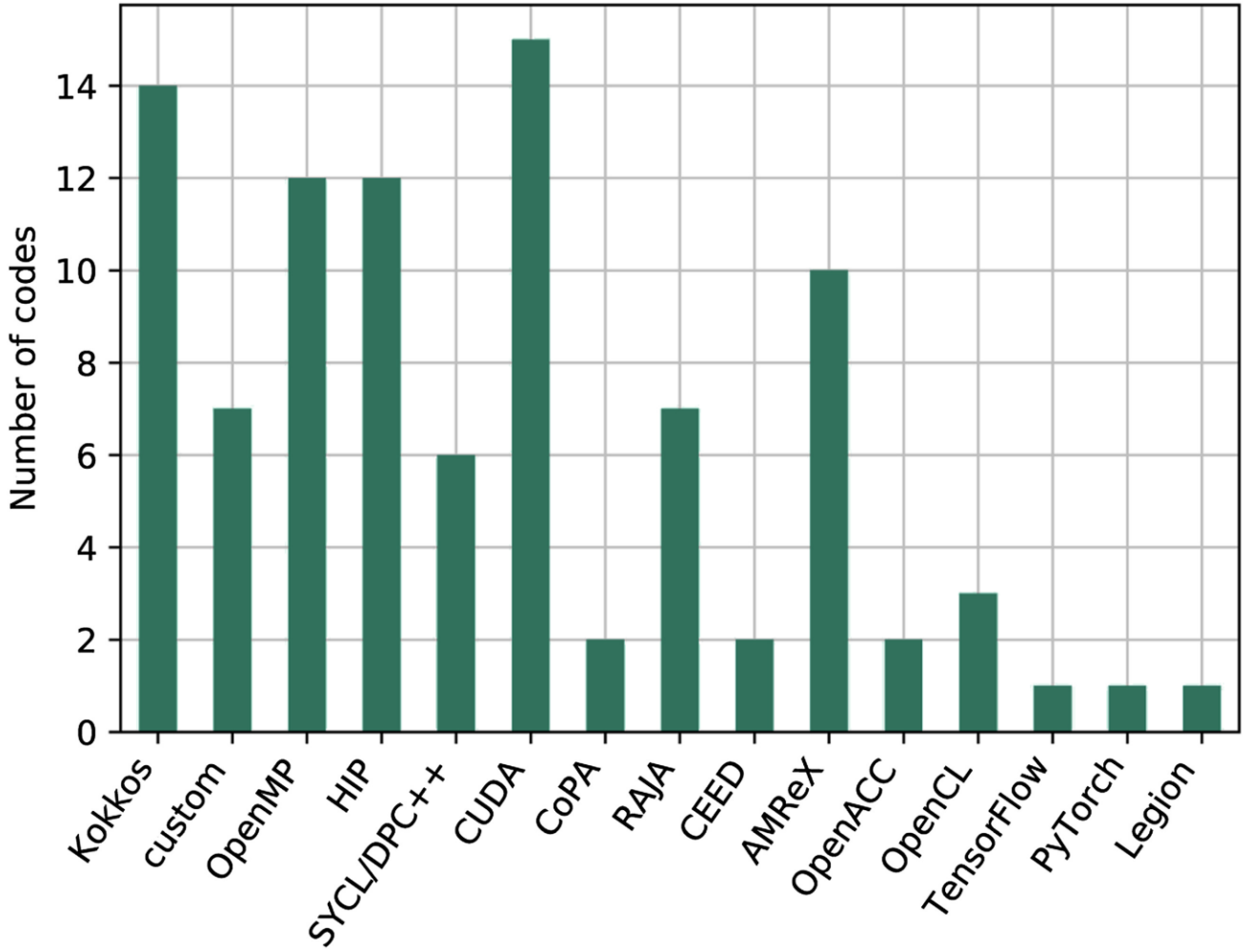
Rank	Site	Computer	Cores	HPL (Eflop/s)
1	DOE/SC/ORNL	Frontier	8,730,112	1.194
2	DOE/SC/ANL	Aurora (Half)	4,742,808	0.585
3	Microsoft Azure	Eagle	1,123,200	0.561
4	RIKEN	Fugaku	7,630,848	0.442
5	EuroHPC/CSC	LUMI	2,752,704	0.379

TOP500 HPL (High-Performance Linpack (HPL) benchmark of high-performance computing (HPC) from November 2023

- Merritt, R. (2022, September 16). What is an exaflop? NVIDIA Blog. Retrieved April 9, 2023, from https://blogs.nvidia.com/blog/2022/07/26/what-is-an-exaflop/
- Strohmaier, E. et al. (2023, November). TOP500 Lists. HPL. Retrieved January 20, 2024, from https://www.top500.org/lists/top500/2023/11/
- Webstudios, G. (2021, August 31). Why countries are competing to build supercomputers? Griffon Webstudios. Retrieved April 9, 2023, from https://griffonwebstudios.com/build-super-computers/

Heterogeneous Exascale Computing Models

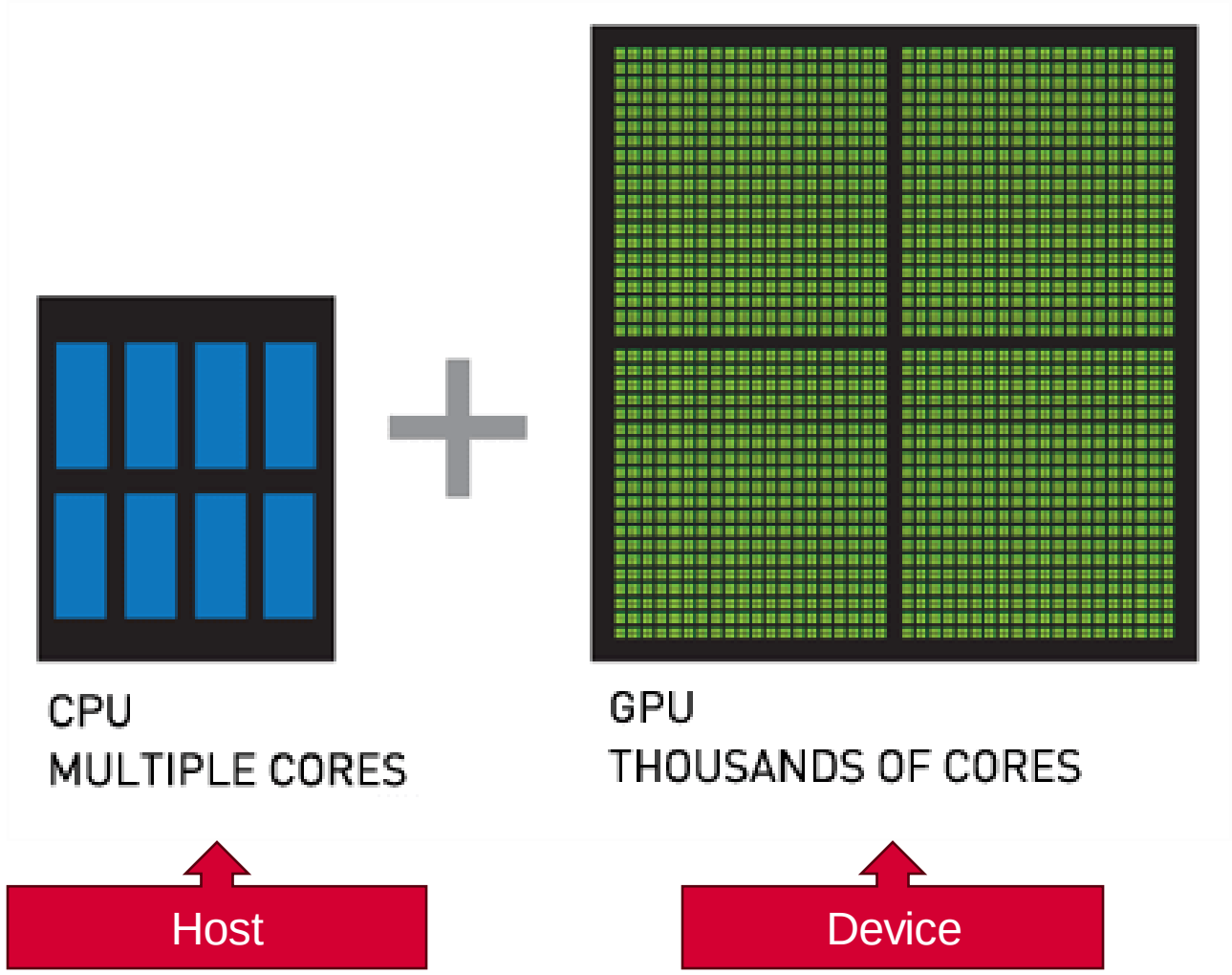
- 22 combinations of GPU
- Most common: CUDA
- Aurora: SYCL (primary), HIP
- Frontier: HIP (primary), SYCL
- El Capitan: HIP (primary), SYCL



• Evans, T. M., Siegel, A., Draeger, E. W., Deslippe, J., Francois, M. M., Germann, T. C., ... & Martin, D. F. (2022). A survey of software implementations used by application codes in the Exascale Computing Project. The International Journal of High Performance Computing Applications, 36(1), 5-12.

GPGPU

- General Purpose computing using GPU
- Work together with CPU
- GPU is designed for parallel processing
- Major manufacturers: Nvidia, AMD, Intel
- Power efficiency TFlop per Watt



GPGPU Advantages

Massive parallel processing:
Scientific simulations

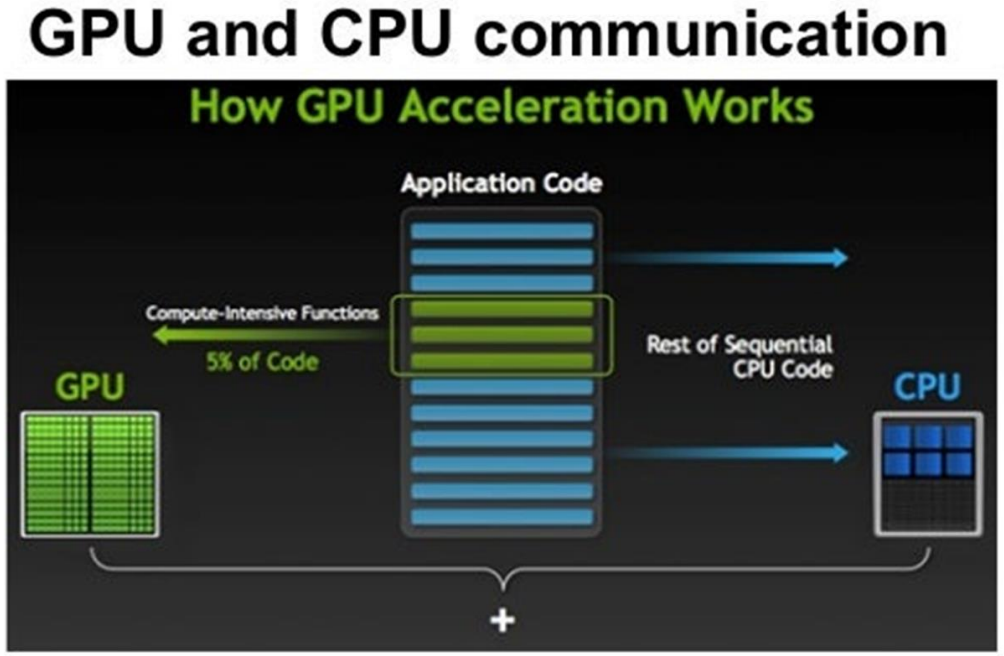
Efficient large dataset handling:
Machine learning algorithms

Real-time processing:
Gaming and streaming

Accelerated financial modeling:
Risk assessment and pricing

GPU Programming Model

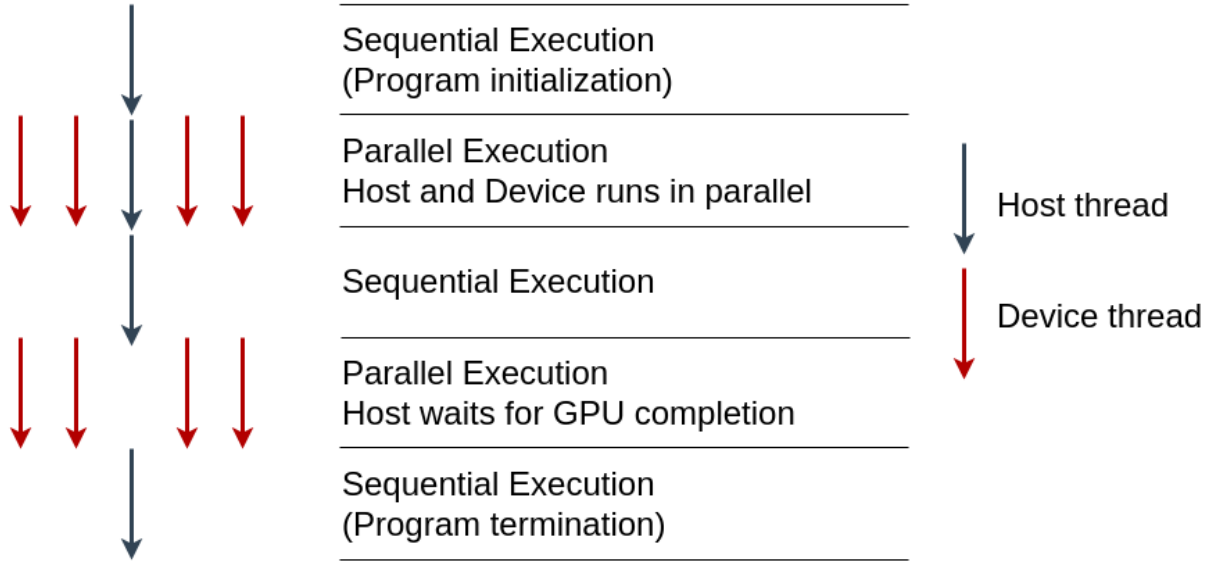
- **CUDA** - proprietary to Nvidia GPUs but most mature and established
- **HIP** - targets AMD GPUs, portable
- **SYCL** - open standard for cross-platform portability
- **DPC++** - Intel's implementation of SYCL
- **OneAPI** - Intel's unified programming model across devices



- Jason. Sanders. CUDA by example : an introduction to general-purpose GPU programming. Addison-Wesley, Upper Saddle River, NJ, 2011.
- Jianbin Fang, Chun Huang, Tao Tang, and Zheng Wang. Parallel programming models for heterogeneous many-cores: a comprehensive survey. CCF Transactions on High Performance Computing, 2(4):382–400, 2020.

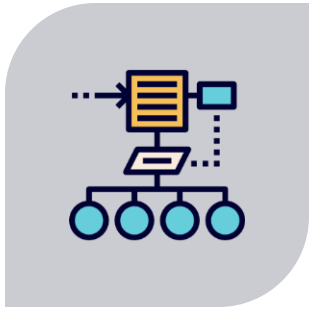
GPU Programming Model

- Globally Sequential Locally Parallel programming pattern
- Invokes parallel kernels that execute across a set of threads
- Each thread executes an instance of the kernel

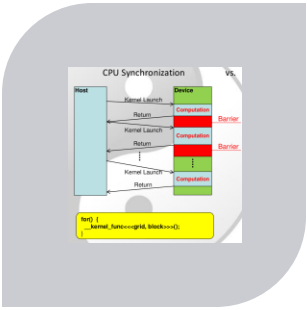


- Jason. Sanders. CUDA by example : an introduction to general-purpose GPU programming. Addison-Wesley, Upper Saddle River, NJ, 2011.
- Jianbin Fang, Chun Huang, Tao Tang, and Zheng Wang. Parallel programming models for heterogeneous many-cores: a comprehensive survey. CCF Transactions on High Performance Computing, 2(4):382–400, 2020.

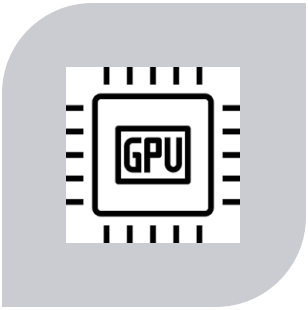
GPGPU Challenges



ALGORITHM ADAPTATION
SEQUENTIAL TO PARALLEL



SYNCHRONIZATION
PUTTING BARRIER



MEMORY MANAGEMENT
H2D AND D2H DATA TRANSFER

PORTABILITY
DIFFERENT GPU DEVICES

- CUDA to SYCL migration retrieved from https://oneapi-sycl-hackathon.devpost.com/ on 01/22/2024
- Levinas, M. (2020, November). What is GPU computing and how is it applied today? What Is GPU Computing And How Is It Applied Today? Retrieved from https://blog.cherryservers.com/what-is-gpu-computing

Table of Contents

Datalog: Bottom-Up Logic Programming Language

A lightweight logic-programming language for deductive-database systems

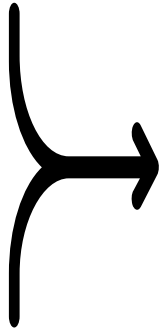
Running the Datalog program extends data from input database creating the output database with all data transitively derivable via the program rules

- 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).

Datalog Example

Facts:

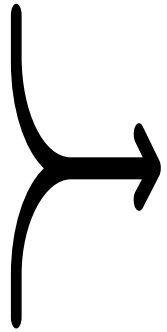
parents(x,y).
children(y,x).



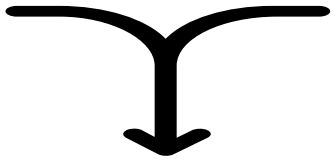
extensional

Rules:

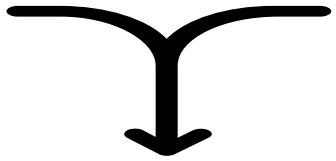
grandparent(x,y) :- parents(x,z), parents(z,y).



intensional



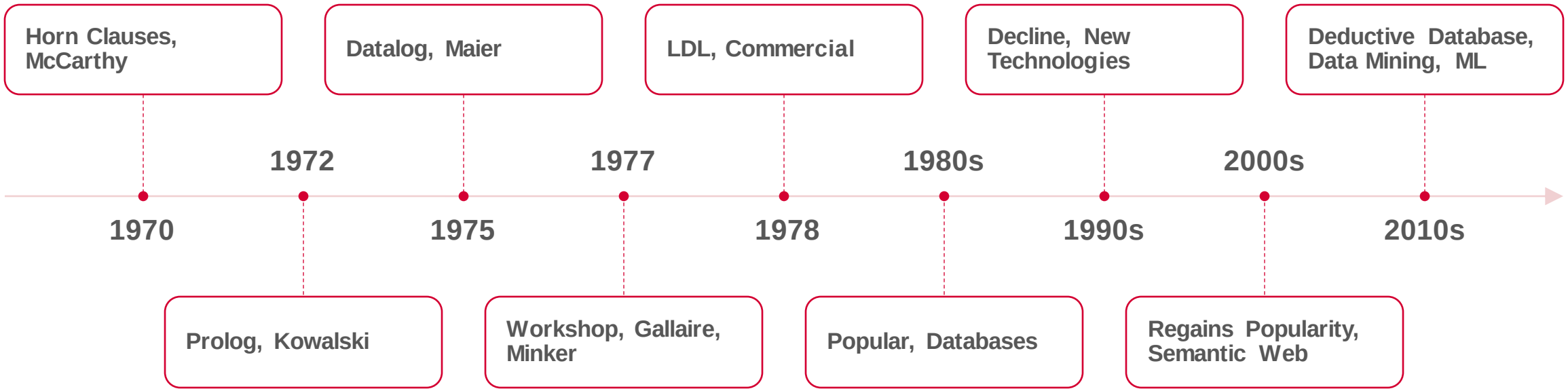
head



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.

Datalog Timeline



- Stefano Ceri, Georg Gottlob, Letizia Tanca, et al. What you always wanted to know about datalog (and never 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, and outlook. 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 on Management of Data, SIGMOD '11, page 1213–1216, New York, NY, USA, 2011. Association for Computing Machinery.

Declarative Analytics Domains

Focuses on the desired output

Utilizes modern Datalog implementation

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 socialnetworkanalysis. In 2013 IEEE 29th International Conference on Data Engineering (ICDE), pages 278–289.IEEE, 2013

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 International Workshop, 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 network analysis. In 2013 IEEE 29th International Conference on Data Engineering (ICDE), pages 278–289. IEEE, 2013.

Assessment of Datalog Solvers

Solver	Technique	Data parallelism	Scalability	Limitations
LogicBlox	Shared memory	No	High	Limited number of threads
Soufflé	Multi core	No	Limited	Internal locking, single node
RadLog	Map Reduce	Yes	Limited	Inadequate for HPC
PRAM	Data parallel	Yes	High	Worse than Soufflé on AWS
SLOG	Data parallel	Yes	High	No GPU usage

- Molham Aref, Balder ten Cate, Todd J. Green, Benny Kimelfeld, Dan Olteanu, Emir Pasalic, Todd L. Veldhuizen, and Geoffrey Washburn. Design and implementation of the logicblox system. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD '15, page 1371–1382, New York, NY, USA, 2015. Association for Computing Machinery
- Alexander Shkapsky, Mohan Yang, Matteo Interlandi, Hsuan Chiu, Tyson Condie, and Carlo Zaniolo. Big data analytics with datalog queries on spark. In Proceedings of the 2016 International Conference on Management of Data, pages 1135–1149, 2016.
- Herbert Jordan, Bernhard Scholz, and Pavle Subotić. Soufflé: On synthesis of program analyzers. In International Conference on Computer Aided Verification, pages 422–430. Springer, 2016.
- 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.
- Thomas Gilray, Arash Sahebolamri, Sidharth Kumar, and Kristopher Micinski. Higher-order, data parallel structured deduction. arXiv preprint arXiv:2211.11573, 2022.

UIC

Table of Contents

Bottom-Up Logic Programming with Datalog

Datalog

Iterated 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

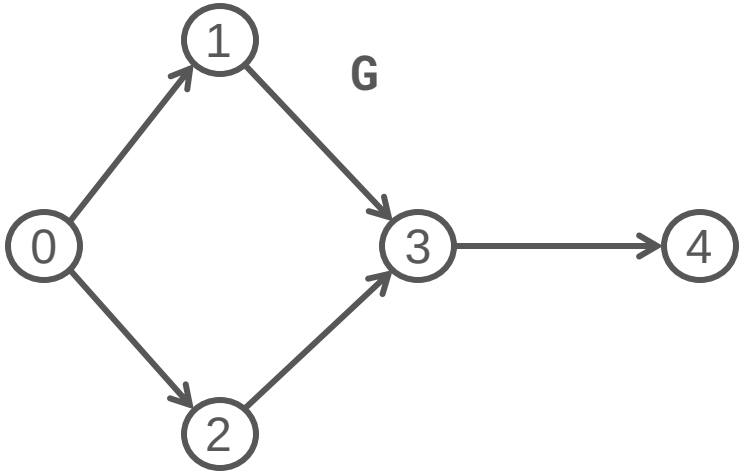
$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$

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.

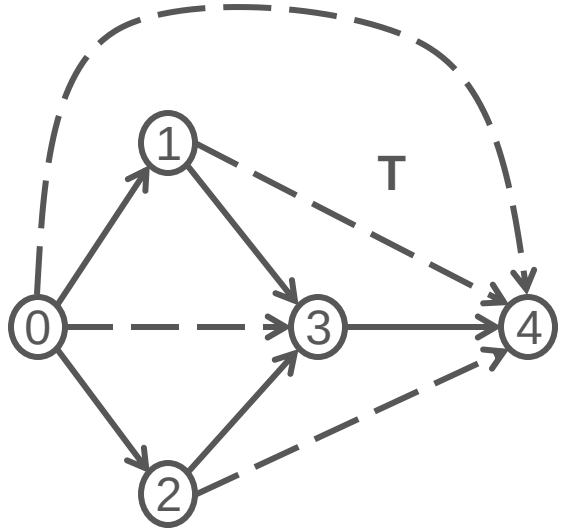
UIC

Transitive Closure: Logical Inference for Graphs



0	1
0	2
1	3
2	3
3	4

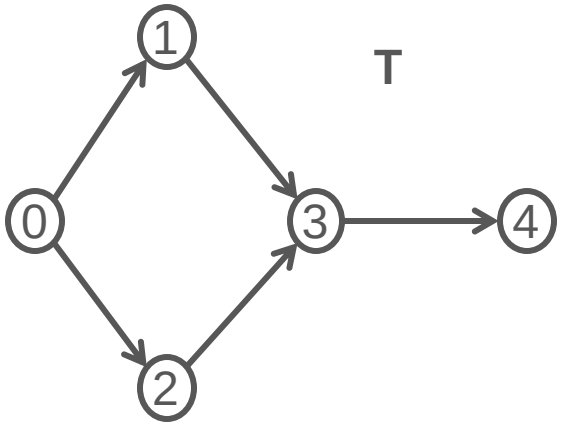
```
T(x,y) <- G(x,y)
T(x,z) <- T(x,y),G(y,z)
```



0	1
0	2
1	3
2	3
3	4
0	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



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

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

1	0
2	0
3	1
3	2
4	3

$\bowtie$

G

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

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

1	0
2	0
3	1
3	2
4	3

$\bowtie$

G

0	1
0	2
1	3
2	3
3	4

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

1	0	3

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

Transitive Closure: Iterations

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

1	0
2	0
3	1
3	2
4	3

$\bowtie$

G

0	1
0	2
1	3
2	3
3	4

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

1	0	3
2	0	3

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

Transitive Closure: Iterations

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

1	0
2	0
3	1
3	2
4	3

$\bowtie$

G

0	1
0	2
1	3
2	3
3	4

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

1	0	3
2	0	3
3	1	4

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

Transitive Closure: Iterations

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

1	0
2	0
3	1
3	2
4	3

$\bowtie$

G

0	1
0	2
1	3
2	3
3	4

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

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

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

Transitive Closure: Iterations

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

1	0
2	0
3	1
3	2
4	3

$\bowtie$

G

0	1
0	2
1	3
2	3
3	4

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

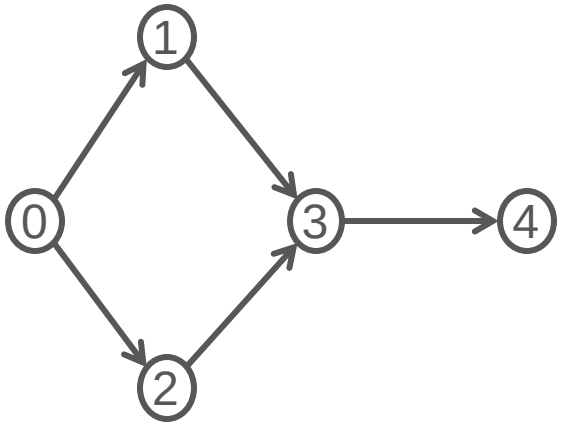
1	0	3
2	0	3
3	1	4
3	2	4

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

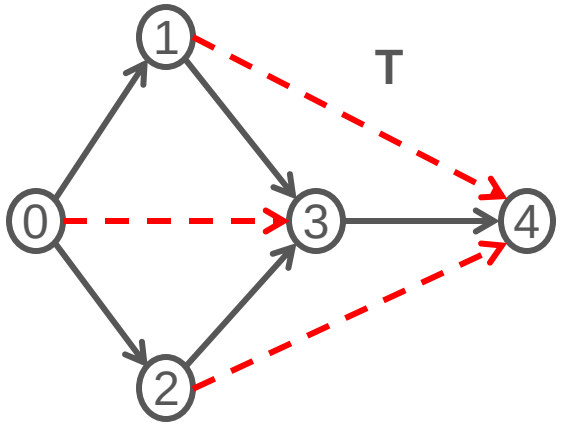
0	3
1	4
2	4

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

Transitive Closure: Iterations 1



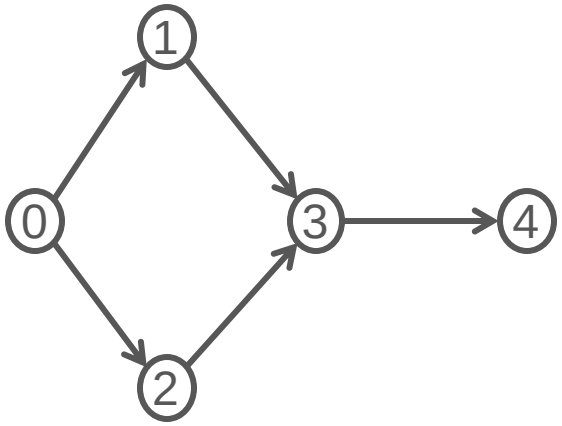
0	1
0	2
1	3
2	3
3	4



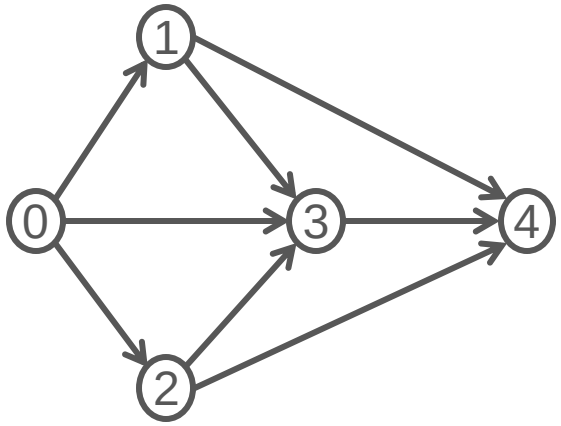
0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4

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

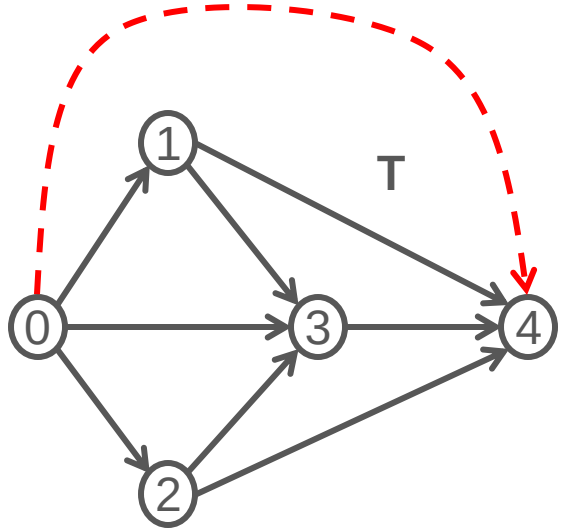
Transitive Closure: Iterations 2



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

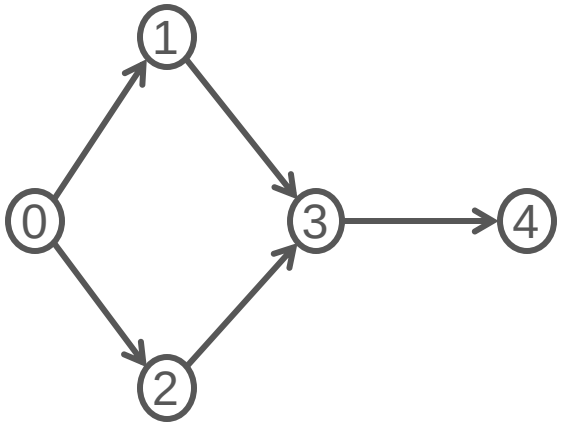


0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4
0	4

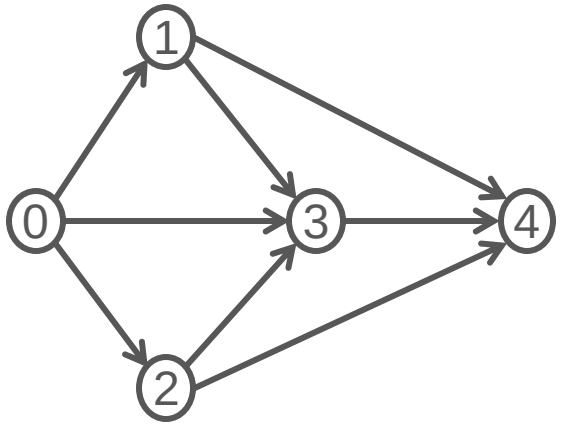
Transitive Closure: Iterations

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

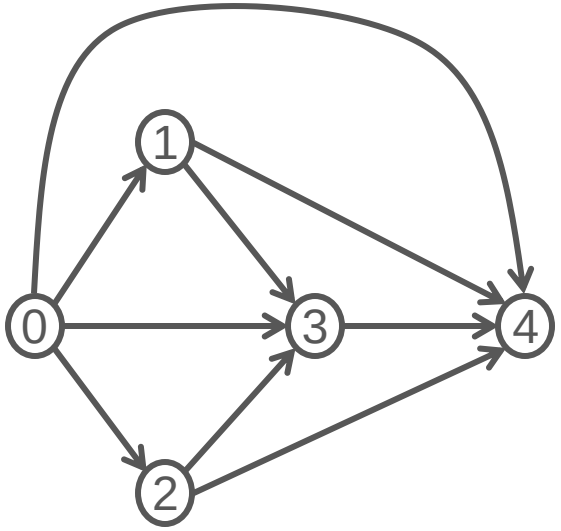
3 Union Projection Join



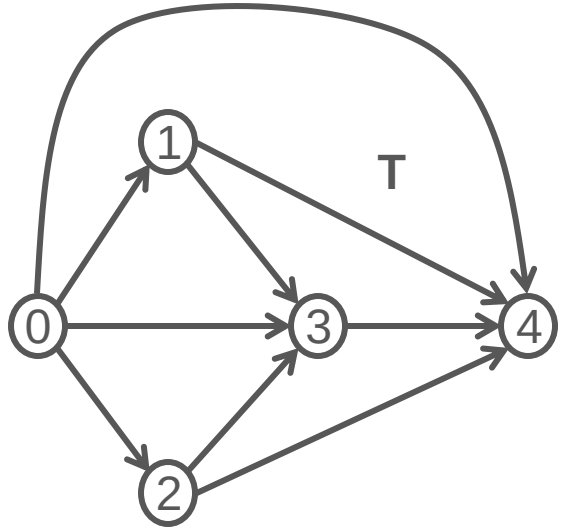
0	1
0	2
1	3
2	3
3	4



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

Table of Contents

How can we do join in parallel?

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
101	25.69	2
102	145.66	3
103	12.11	1
103	44.00	2

Parallel Join ⋈

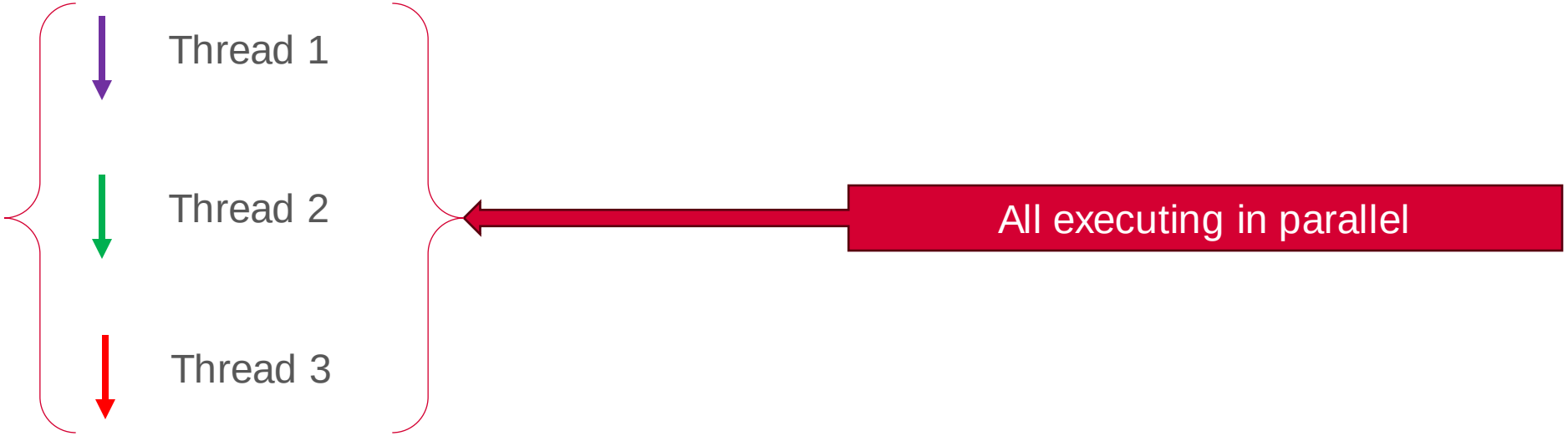
User (Outer Relation)

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

Order (Inner Relation)

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

⋈



Parallel Join

Parallel Join

Algorithms

Hardwares

Load balancing

Comparison of Join Algorithms

Algorithm	Pros	Cons	Dataset
Non indexed nested loop	Simple	High time complexity, low efficiency	Small
Indexed nested loop	Index for faster performance	High memory usage	Small to medium, indexed tables
Sort merge	Scalable, handles non-equality joins	Requires sorting, high memory usage	Medium
Hash	Scalable, handles equality joins	Requires partitioning, cannot do non-equality	Large
Hybrid (sort + hash)	Reduce communication cost, Handles skewed data	Complex, requires tuning	Large

- 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.
- Ran Rui, Hao Li, and Yi Cheng Tu. Efficient join algorithms for large database tables in a multi-gpu environment. Proceedings of the VLDB Endowment, 14:708–720, 2020.
- 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.
- Claude Barthels, Ingo M¨uller, Timo Schneider, Gustavo Alonso, and Torsten Hoefler. Distributed join algorithms on thousands of cores. Proc. VLDB Endow., 10(5):517–528, jan 2017
- Chengxin Guo, Hong Chen, Feng Zhang, and Cuiping Li. Parallel hybrid join algorithm on gpu. 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pages 1572–1579, 2019

Parallel Join

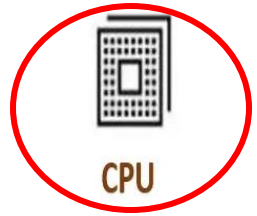
Parallel Join

Algorithms

Hardwares

Load balancing

Join on CPUs: Multi-core HJ and SMJ



- Barthels et al. compared HJ and SMJ on distributed multi-core systems using 4096 processor cores (MPI)
- Communication inefficiencies can significantly impede performance
- More cores not always means better performance, sometimes degrade
- Bounded by the low computation capacity of CPUs and higher communication costs

- Claude Barthels, Ingo M¨uller, Timo Schneider, Gustavo Alonso, and Torsten Hoefler. Distributed join algorithms on thousands of cores. Proc. VLDB Endow.,10(5):517–528, jan 2017.
- Chengxin Guo, Hong Chen, Feng Zhang, and Cuiping Li. Distributed join algorithms on multi-cpu clusters with gpu direct rdma. Proceedings of the 48th International Conference on Parallel Processing,2019.

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 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.
- 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: 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.

Join on GPUs: Secondary storage

- Zinn et al. presented multi-GPUs join with secondary storage support
- Focused on triangle counting problem
- Allow overfitting global GPU memory with SSD storage
- Leapfrog Triejoin (LFTJ) algorithm: a worst-case-optimal algorithm
- No mention of multi-node heterogeneous systems

• Daniel Zinn, Haicheng Wu, Jin Wang, Molham Aref, and Sudhakar Yalamanchili. General-purpose joinalgorithms 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.

UIC

Parallel Join

Parallel Join

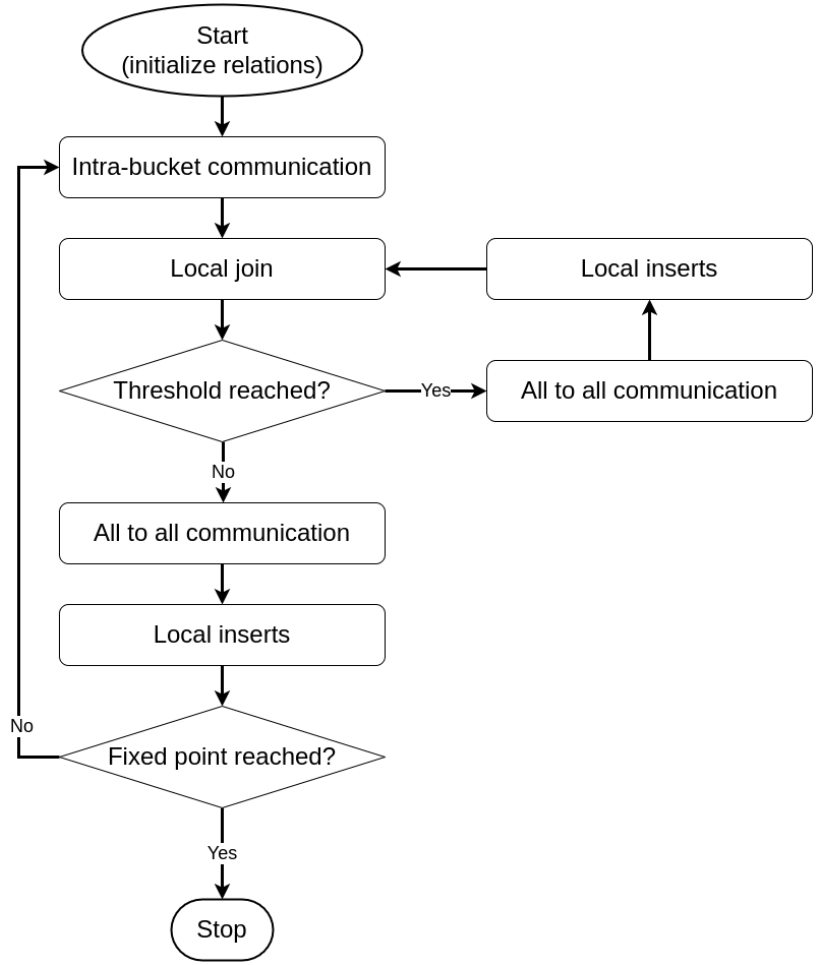
Algorithms

Hardwares

Load balancing

Load Balancing Multi-node Joins

- 2 types of load imbalance issues:
  - Spatial: across multiple computing nodes
  - Temporal: distributing newly discovered tuples
  - Solutions: Bucket refinement,consolidation,Iteration rollover
- Range of favorable scaling exists: 256 to 32K processes
- Performance deteriorates beyond that range



Handling temporal load imbalance during the join

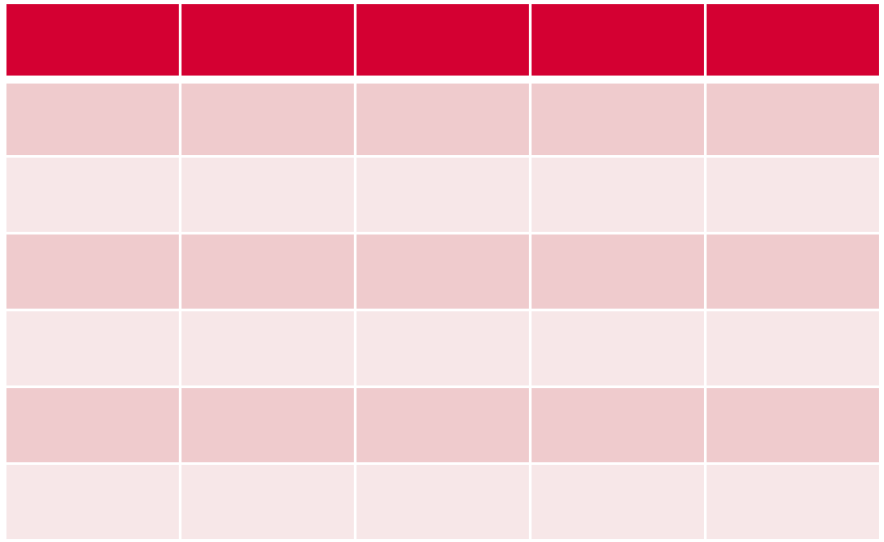
- Sidharth Kumar and Thomas Gilray. Distributed relational algebra at scale. In International Conference on High Performance Computing, Data, and Analytics (HiPC). IEEE, 2019.
- Sidharth Kumar and Thomas Gilray. Load-balancing parallel relational algebra. In International Conference on High Performance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

Table of Contents

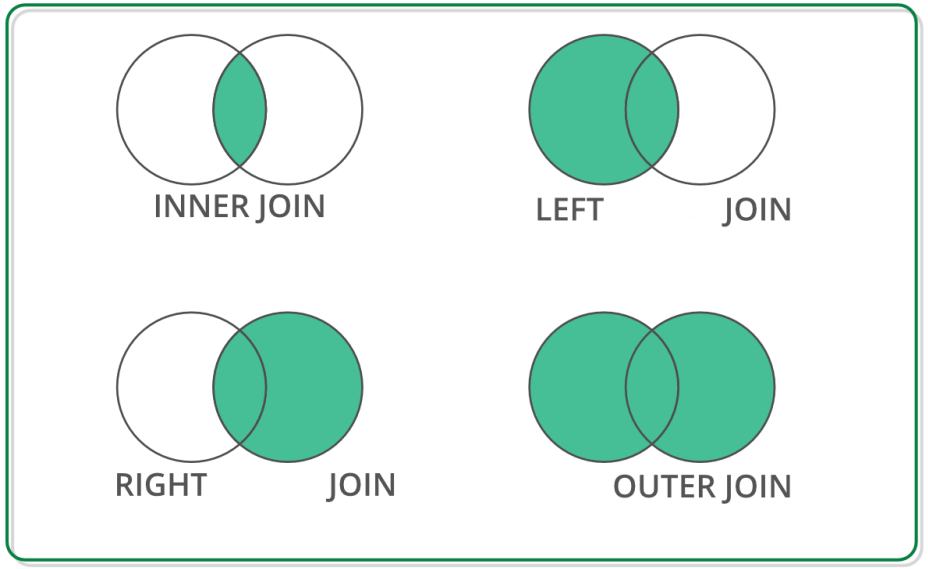
First approach:
Off-the-shelf RA Data Structure

Off-the-shelf Data Structure for RA Operation

DataFrame: 2D labeled tabular data structure



DataFrame has RA primitives APIs

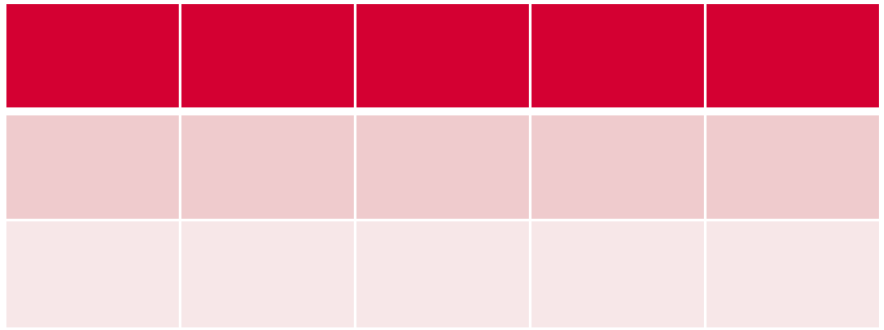


- Reback, J., McKinney, W., Van Den Bossche, J., Augspurger, T., Cloud, P., Klein, A., ... & Seabold, S. (2020). pandas-dev/pandas: Pandas 1.0. 5. Zenodo.
- Chen, D. Y. (2017). Pandas for everyone: Python data analysis. Addison-Wesley Professional.
- Singh, R. (2020, July 1). Merging DataFrames with 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

CPU

DataFrame: 2D labeled tabular data structure



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.

UIC

CPU (Pandas) and GPU (cuDF)

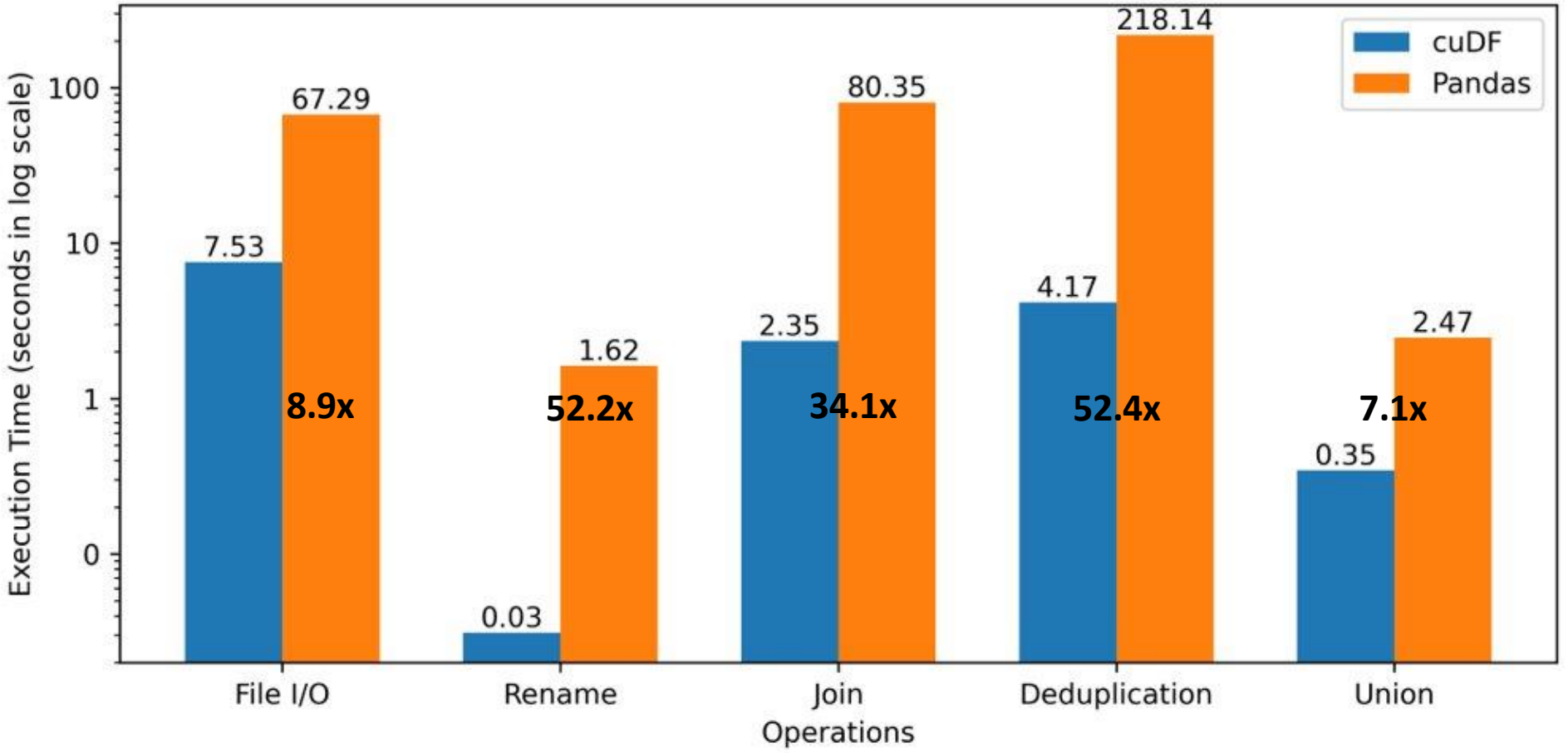
```
import pandas as pd
import cudf

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

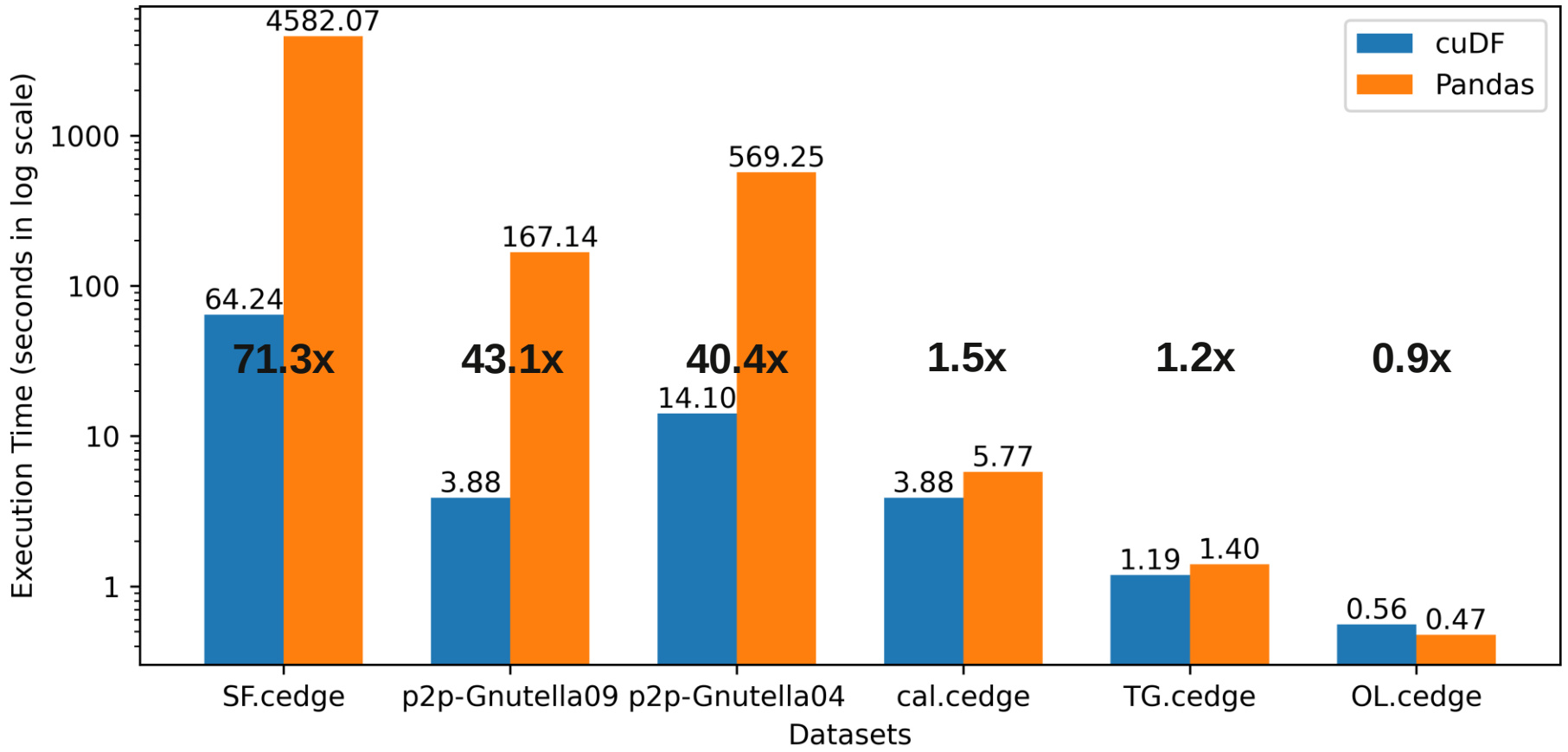
- https://github.com/harp-lab/GPUJoin/edit/main/IA3_2022/basic_operations.py

Performance Improvement: RA operations



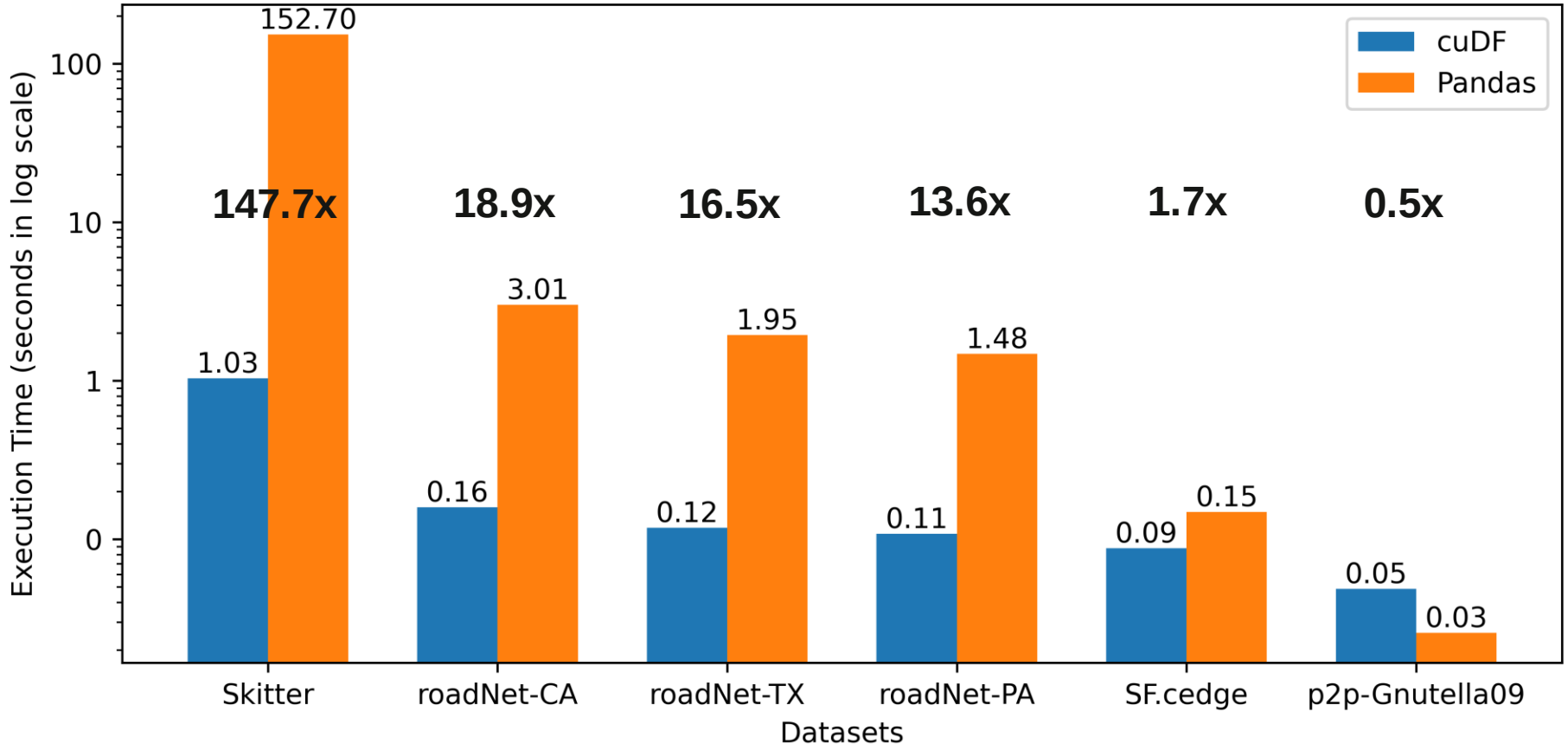
- A. R. Shovon, L. R. Dyken, O. Green, T. Gilray and S. Kumar, "Accelerating Datalog applications with cuDF," 2022 IEEE/ACM Workshop on Irregular Applications: Architectures and Algorithms (IA3), Dallas, TX, USA, 2022, pp. 41-45
- Leadership Computing Facility, A. (2022). Argonne Leadership Computing Facility. Theta GPU Nodes. URL: https://www.alcf.anl.gov/support-center/theta-gpu-nodes

Performance Improvement: Transitive Closure



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

Performance Improvement: Triangle Counting



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

DataFrame Based Datalog Applications

✓ 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

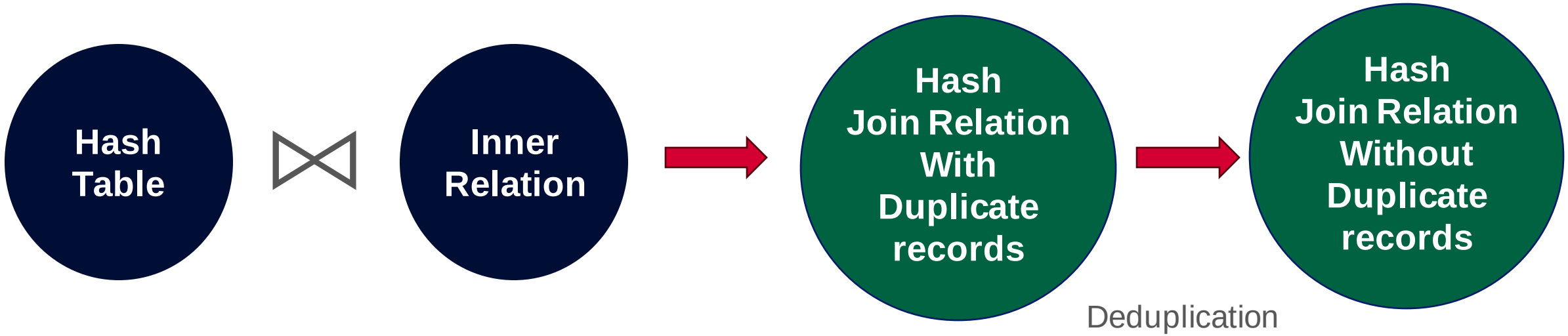
Improvement Opportunity for TC

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

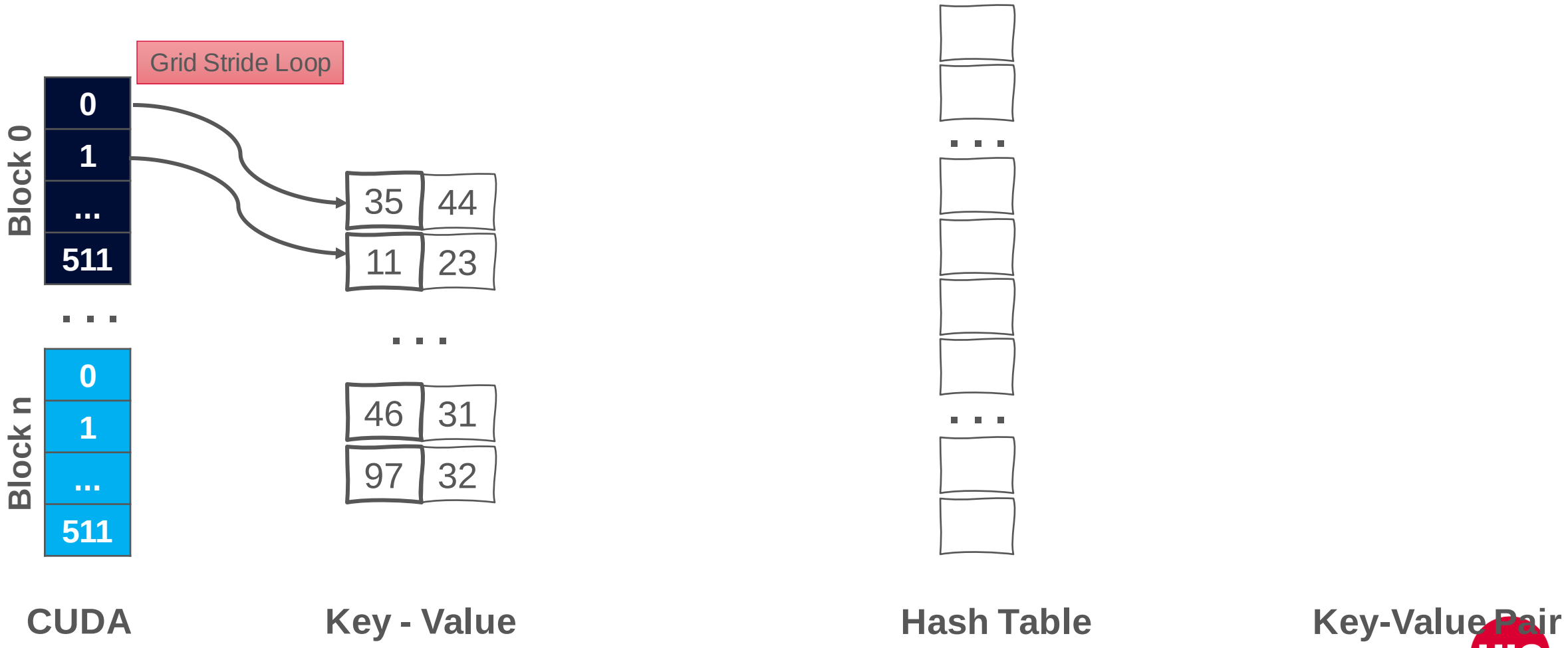
- A. R. Shovon, L. R. Dyken, O. Green, T. Gilray and S. Kumar, "Accelerating Datalog applications with cuDF," 2022 IEEE/ACM Workshop on Irregular Applications: Architectures and Algorithms (IA3), Dallas, TX, USA, 2022, pp. 41-45
- Green, O., Du, Z., Patel, S., Xie, Z., Liu, H., & Bader, D. A. (2021,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

Second approach:
Low level iterative hash join

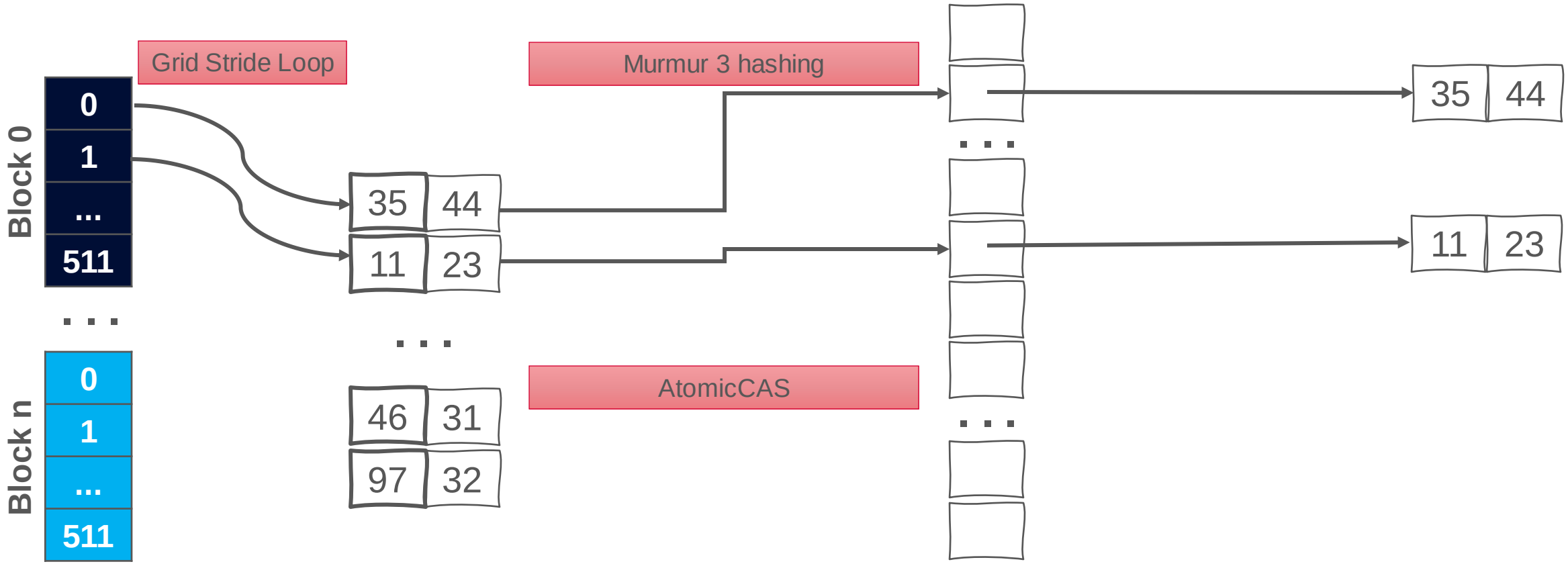
Hash Join Process



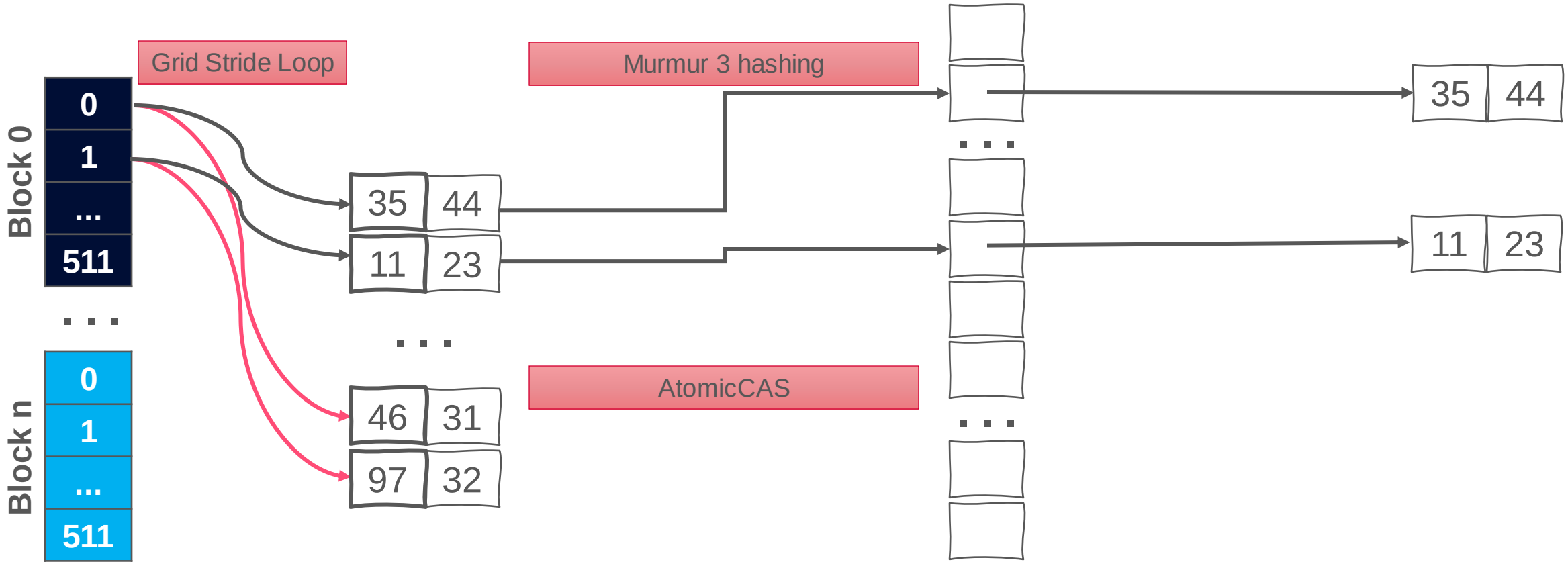
Hash Table (Open Addressing, Linear Probing)



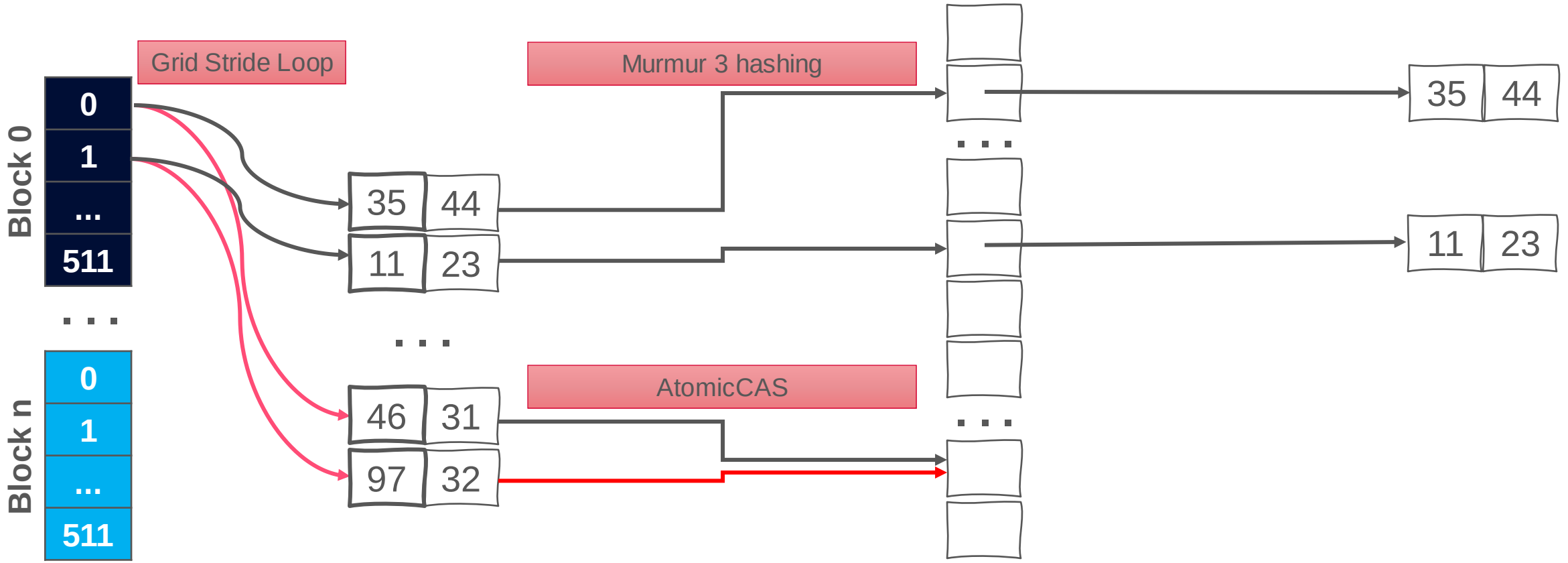
Hash Table (Open Addressing, Linear Probing)



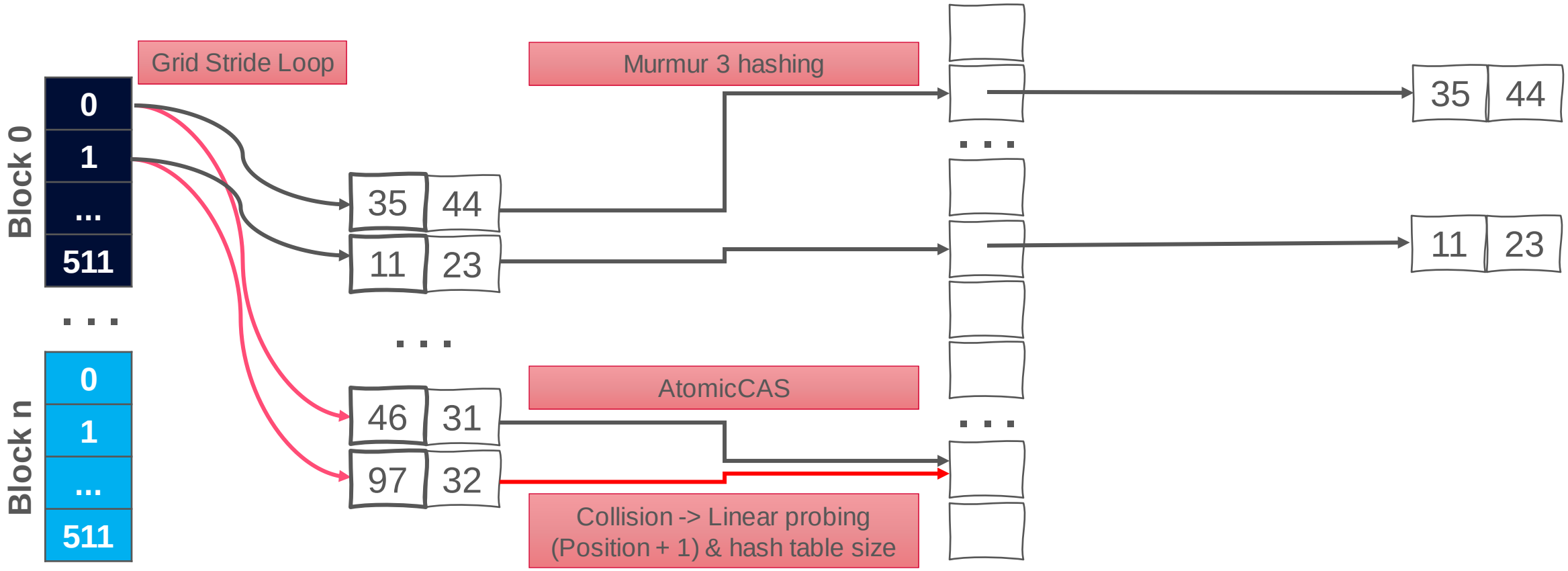
Hash Table (Open Addressing, Linear Probing)



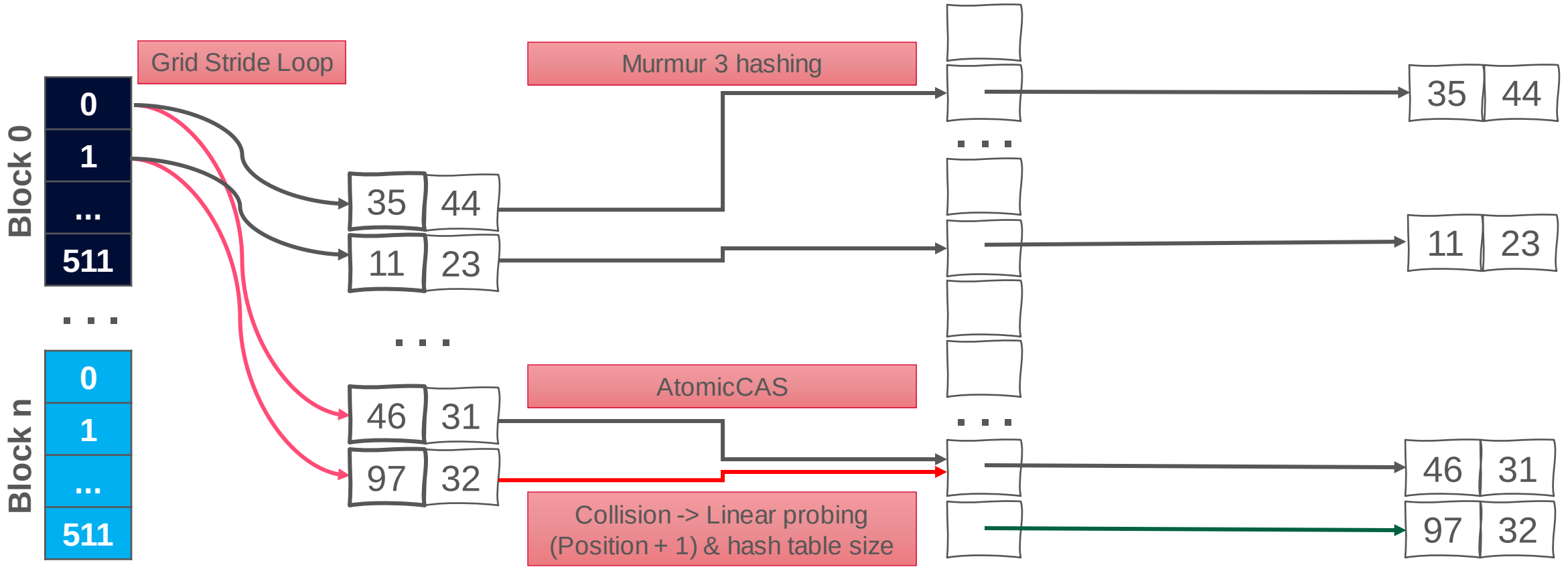
Hash Table (Open Addressing, Linear Probing)



Hash Table (Open Addressing, Linear Probing)



Hash Table (Open Addressing, Linear Probing)



Performing Hash Join on GPU

Static Hash Table

Key	Value

⋈

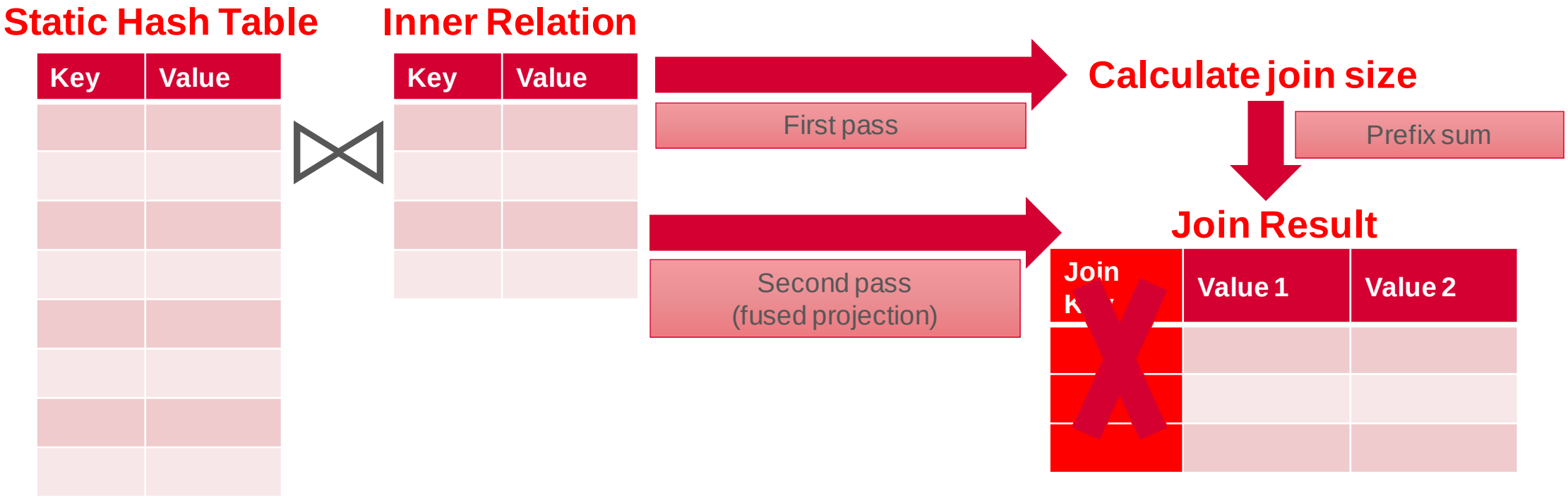
Inner Relation

Key	Value

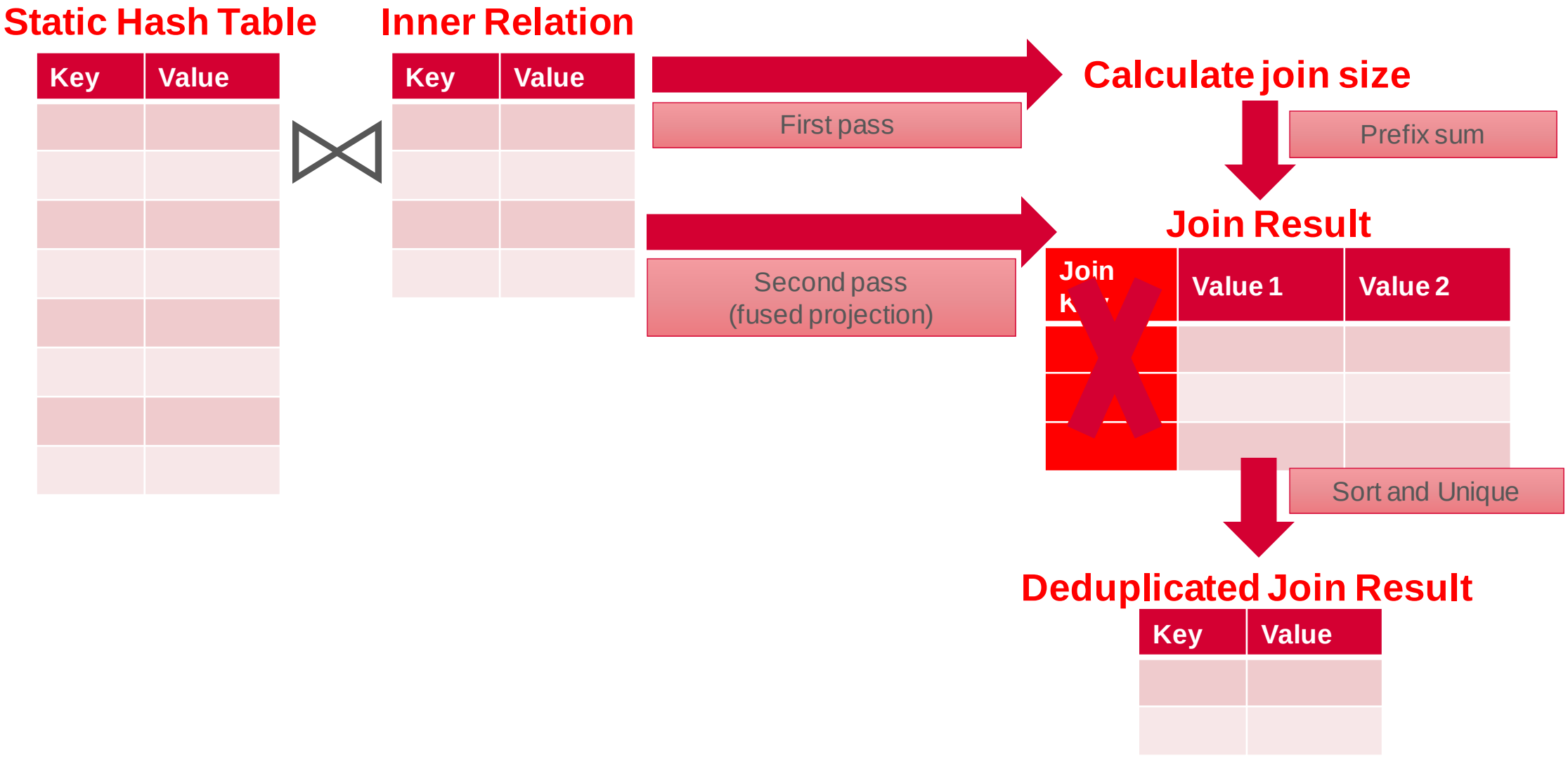
First pass

Calculate join size

Performing Hash Join on GPU



Performing Hash Join on GPU



CUDA Advantages over cuDF

- 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

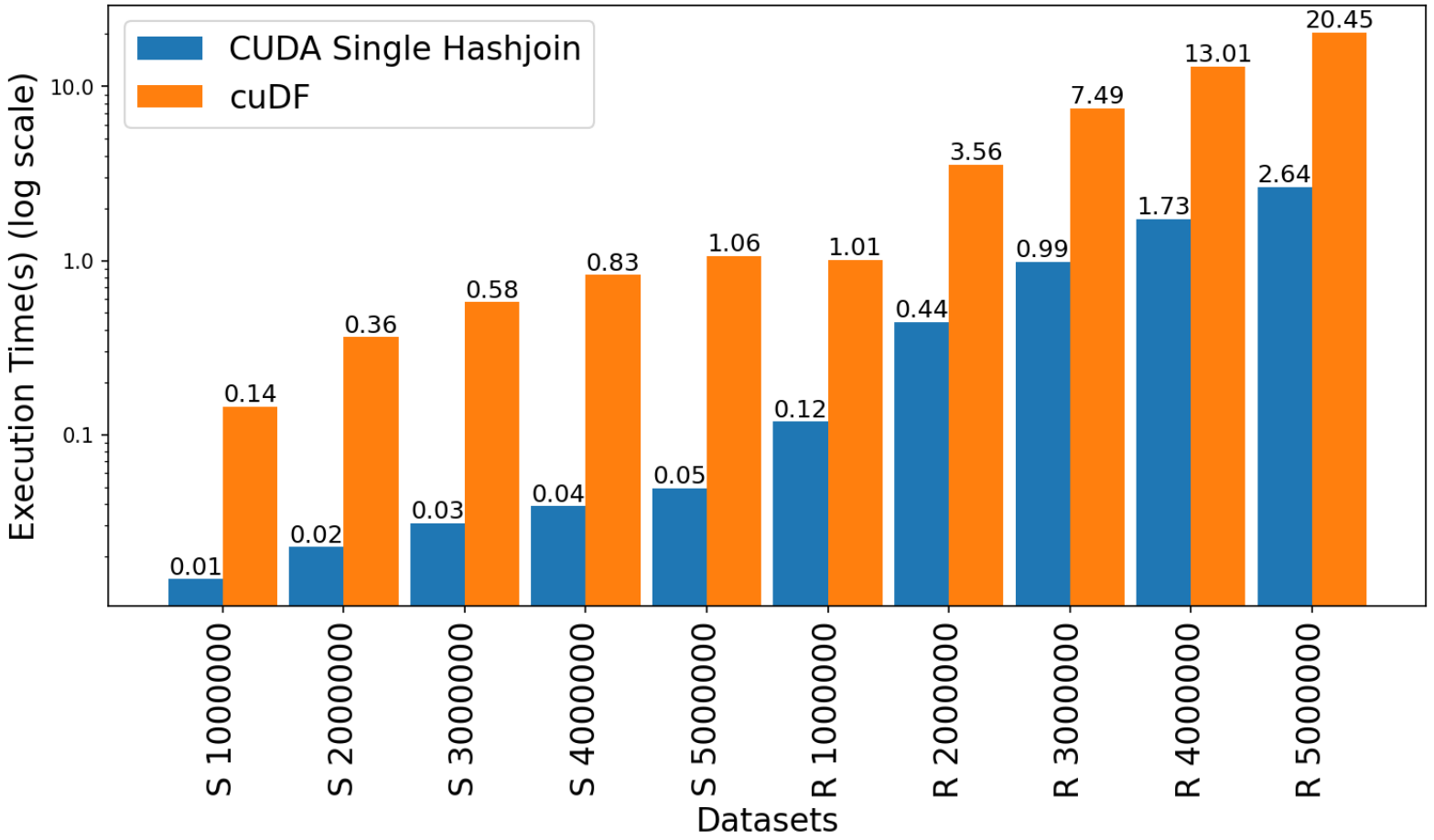
Hash Table Performance

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



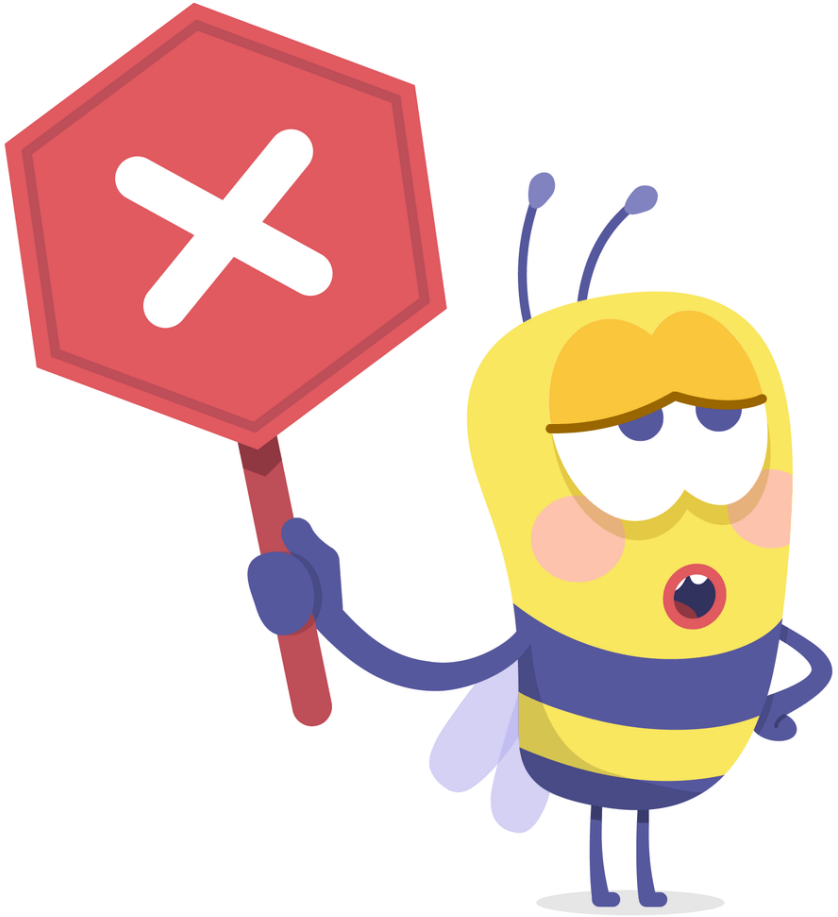
- Leadership Computing Facility, A. (2022). Argonne Leadership Computing Facility. Theta GPU Nodes. URL: https://www.alcf.anl.gov/support-center/theta-gpu-nodes

Transitive Closure: 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

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

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

Topological Data Analysis

TDA analyzes high-dimensional data sets using algebraic topology

Main tool: persistent homology, adapts homology to point cloud

Extracts topological features and visualizes in lower-dimensional space

Analyzes complex systems: 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

PH for High-Dimensional Data Analysis

Verify — PH of rs-fMRI can compactly represent topological features of FCNs

Demonstrate — FCN networks are statistically similar despite variations in TRs

Remove — Non-neural variability in multi-site case-control studies

- Ashley Suh, Mustafa Hajij, Bei Wang, Carlos Scheidegger, and Paul Rosen. Persistent homologyguided force-directedgraph 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.

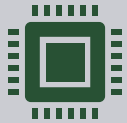
Table of Contents

Future Research Direction

- Develop a multi-node multi-GPU backend for Datalog
- Design GPU benchmarking techniques for iterative RA
- Explore PH-based techniques on multi-site temporal brain FCNs
- Develop declarative analytics technique for high-dimensional datasets

Table of Contents

Conclusion



Reviewed Datalog applications on heterogeneous systems

Explored topological data analysis techniques for brain network

Identified research direction for declarative analytics on high-dimensional data

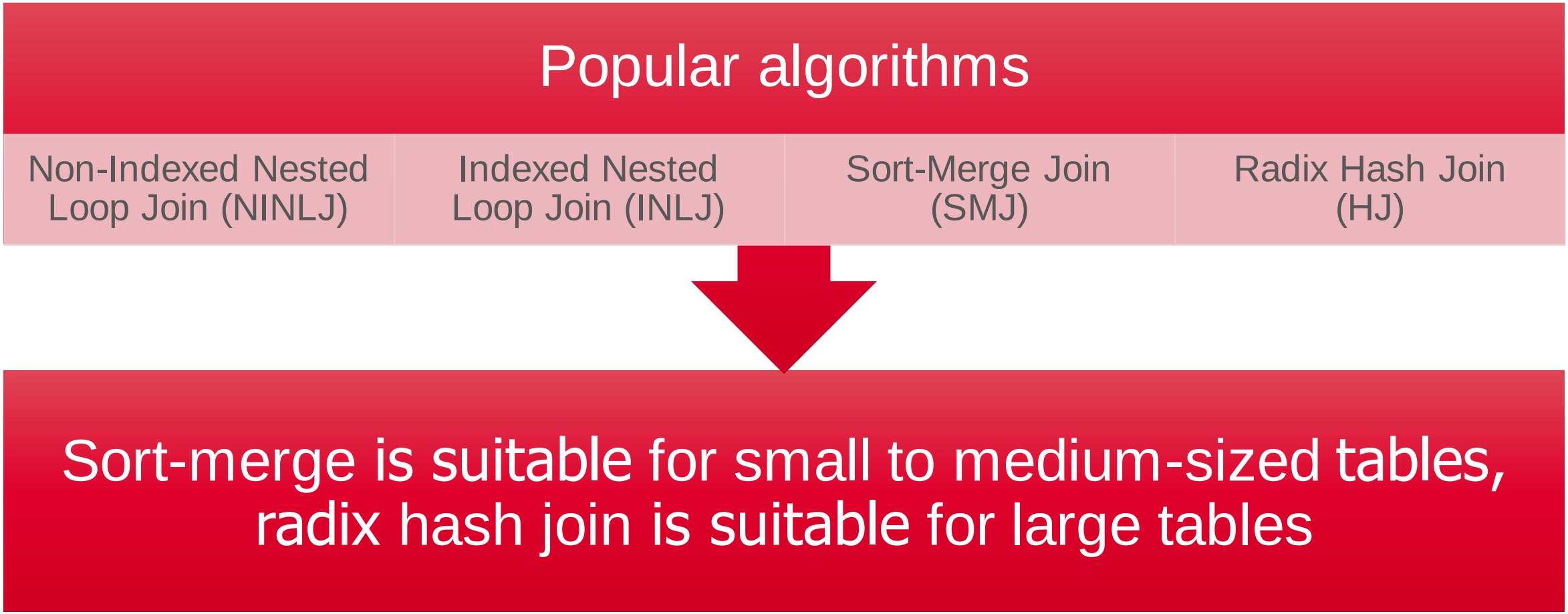
Publications

- Ahmedur Rahman Shovon, Landon Richard Dyken, Oded Green, Thomas Gilray, and Sidharth Kumar. Accelerating datalog applications with cudf. In 2022 IEEE/ACM 11th Workshop on Irregular Applications: Architectures and Algorithms (IA3), pages 41–45, 2022
- Ahmedur Rahman Shovon, Thomas Gilray, Kristopher Micinski, and Sidharth Kumar. Towards Iterative Relational Algebra on the GPU. In 2023 USENIX Annual Technical Conference (USENIX ATC 23), pp. 1009-1016.
- Yihao Sun, Ahmedur Rahman Shovon, Thomas Gilray, Kristopher Micinski, Sidharth Kumar. Modern Datalog on the GPU, In review.
- Sewell, A., Fan, K., Shovon, A. R., Dyken, L., Kumar, S., & Petruzza, S. (2024, January). Bruck Algorithm Performance Analysis for Multi-GPU All-to-All Communication. In Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region (pp. 127-133).
- Sidharth Kumar, Ahmedur Rahman Shovon, and Gopikrishna Deshpande. The invariance of persistent homology of brain networks to data acquisition-related non-neural variability in resting state fmri. Human Brain Mapping, 44(13), 4637–4651
- Darshan Shimoga Chandrashekar, Santhosh Kumar Karthikeyan, Praveen Kumar Korla, Henalben Patel, Ahmedur Rahman Shovon, Mohammad Athar, George J Netto, Zhaohui S Qin, Sidharth Kumar, Upender Manne, et al. Ualcan: An update to the integrated cancer data analysis platform. Neoplasia, 25:18–27, 2022.
- Kashyap Balakavi, Rushitha Janga, Ahmedur Rahman Shovon, Don Dempsey, Elliot Lefkowitz, Sidarth Kumar. Scalable, interactive and hierarchical visualization of virus taxonomic data. In 2023 IEEE Workshop on Visual Analytics in Healthcare (VAHC), in conjunction with IEEE VIS 2023.


Thank You
ashov@uic.edu
UIC

Appendix

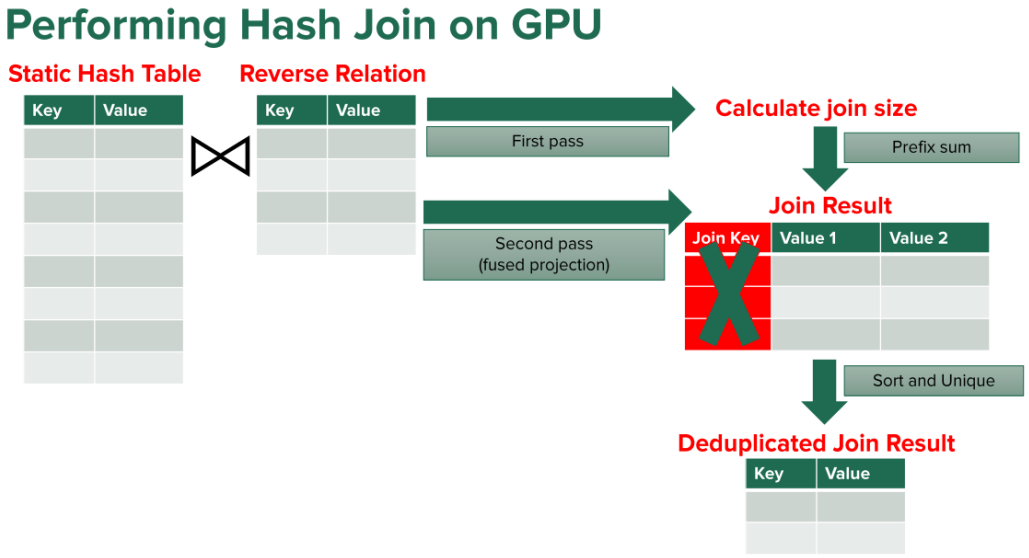
Parallel Join: Algorithms



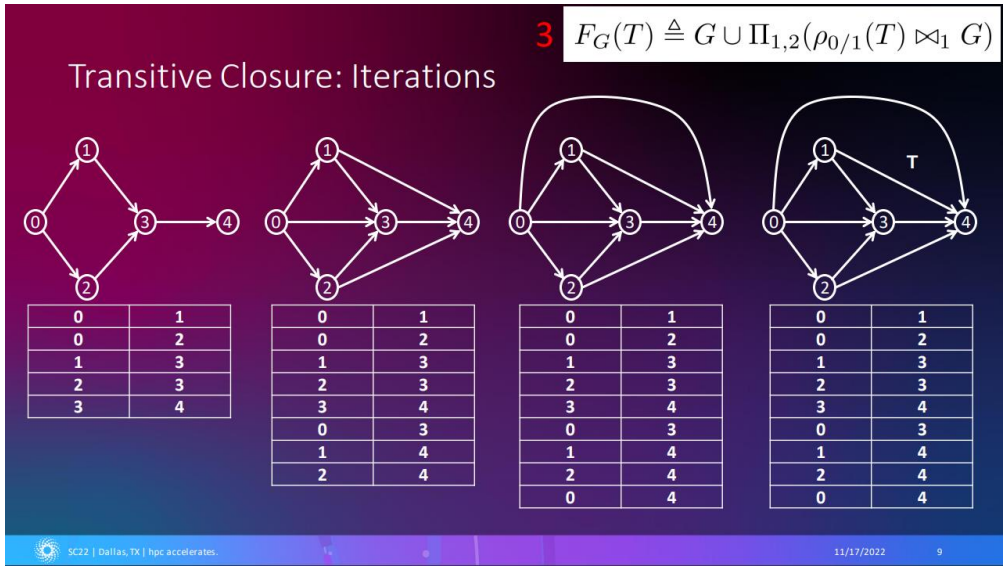
- Chengxin Guo, Hong Chen, Feng Zhang, and Cuiping Li. Parallel hybrid join algorithm on gpu. 2019IEEE 21st International Conference on High Performance Computing and Communications; IEEE17th International Conference on Smart City; IEEE 5th International Conference on Data Science andSystems (HPCC/SmartCity/DSS), 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.

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.

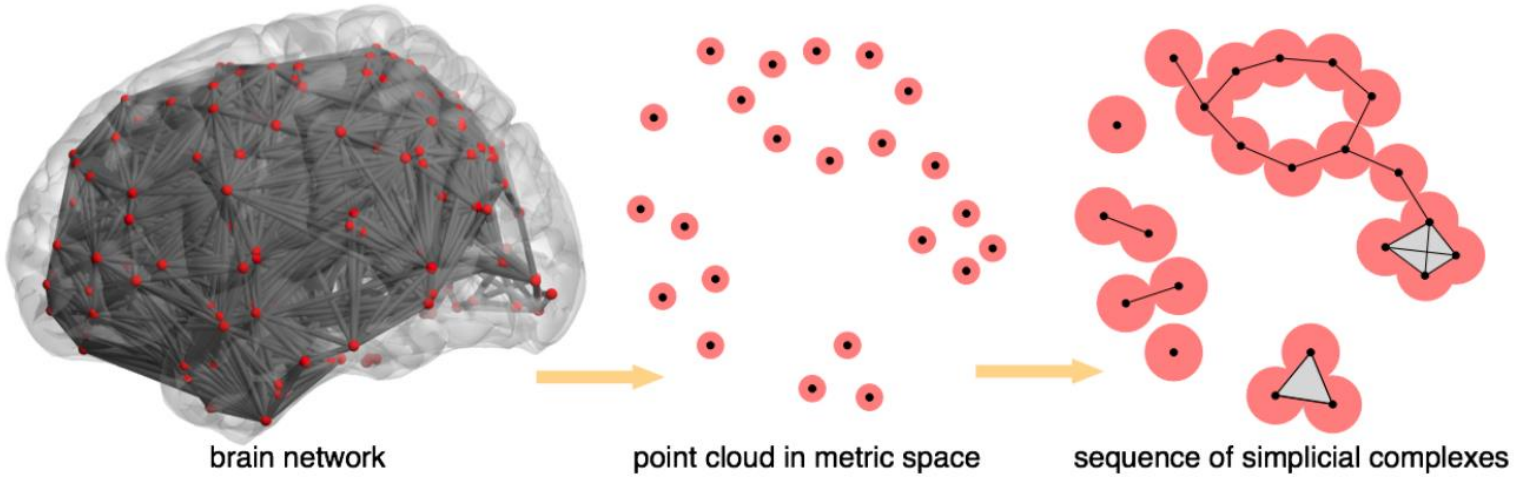


Preserve Brain Topology

Topology of brain networks should be preserved irrespective of sampling density

Metrics capturing topology may be statistically similar across sampling periods, reducing non-neural variability

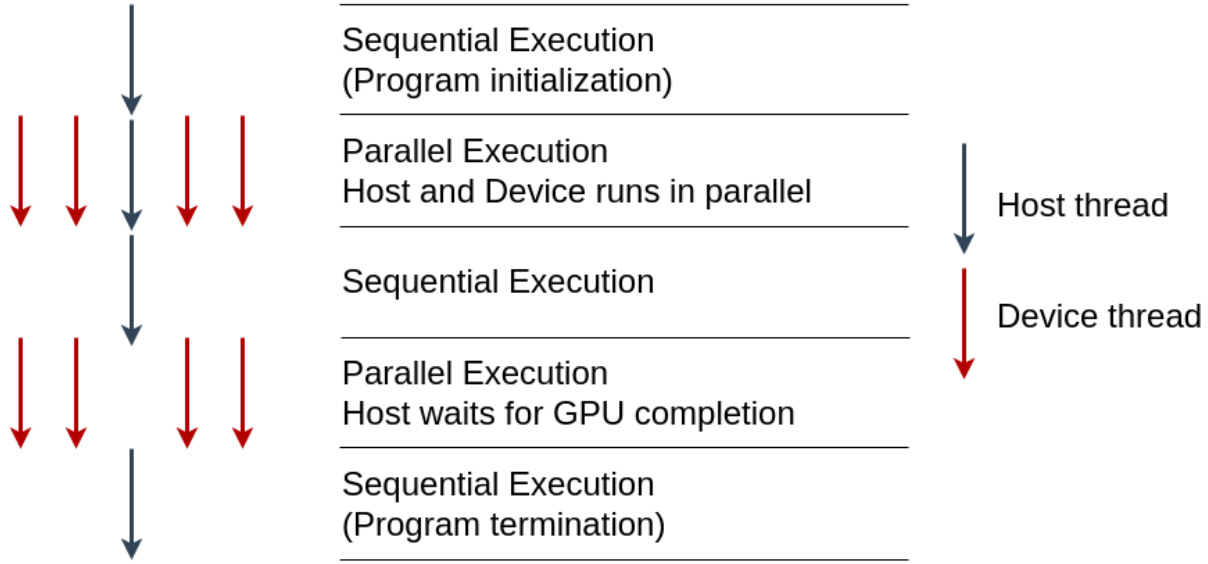
PH can be robust to changes in data acquisition parameters like sampling period



- A. Patania, F. Vaccarino, and G. Petri, "Topological analysis of data," EPJ Data Science, vol. 6, pp. 1–6, 2017.
- Bourakna, A.E.Y., Chung, M.K., Ombao, H. 2022 Topological data analysis for multivariate time series data arXiv:2204.13799

CUDA Programming Model

- Globally Sequential Locally Parallel programming pattern
- Invokes parallel kernels that execute across a set of threads
- CUDA spawns the threads from a hierarchy of grid (3D) and block (3D)
- Each thread executes an instance of the kernel
- Each thread has thread ID, program counter, registers, and per-thread private memory
- Supports C/C++, Fortan, Python



- Jason. Sanders. CUDA by example : an introduction to general-purpose GPU programming. Addison-Wesley, Upper Saddle River, NJ, 2011.
- Jianbin Fang, Chun Huang, Tao Tang, and Zheng Wang. Parallel programming models for heterogeneous many-cores: a comprehensive survey. CCF Transactions on High Performance Computing, 2(4):382–400, 2020.

GPGPU Challenges

Complex memory hierarchies

Different architectures from multicore CPUs

Convert sequential programming to parallel

- Gerassimos Barlas. Chapter 6 - gpu programming. In Gerassimos Barlas, editor, Multicore and GPU Programming, pages 391–526. Morgan Kaufmann, Boston, 2015.
- J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone and J. C. Phillips, "GPU Computing," in Proceedings of the IEEE, vol. 96, no. 5, pp. 879-899, May 2008, doi: 10.1109/JPROC.2008.917757.

Join on GPUs: Control Flow Analysis (CFA)

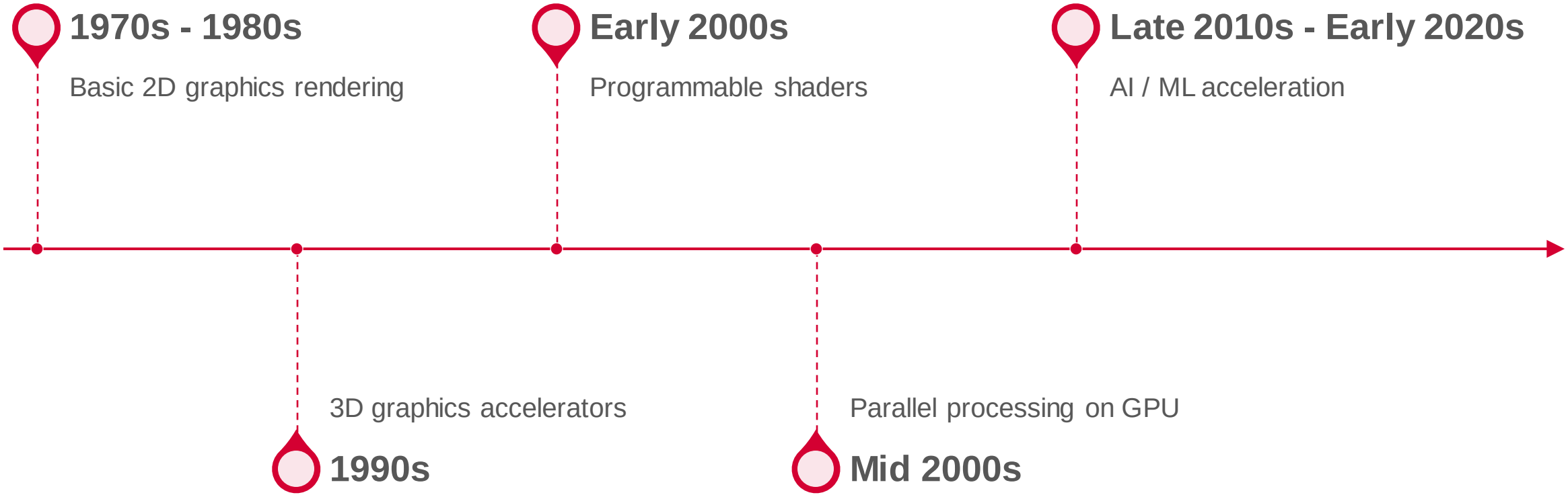
Parallel functional CFA encoded in Datalog utilizes RA as the foundation on GPU

Applied 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.

Advancements in GPU



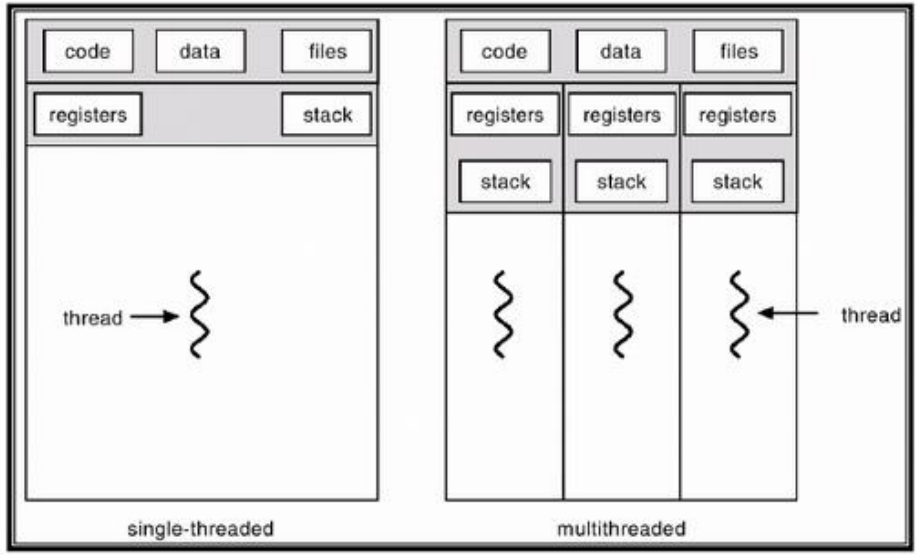
CPU vs GPU (A Sample Machine)

CPU

- 13th Gen Intel® Core™ i9-13900H
- 14 cores (6 P-cores + 8 E-cores)
- Total 20 threads
- Suitable for serial workloads
- Access to system memory (RAM)

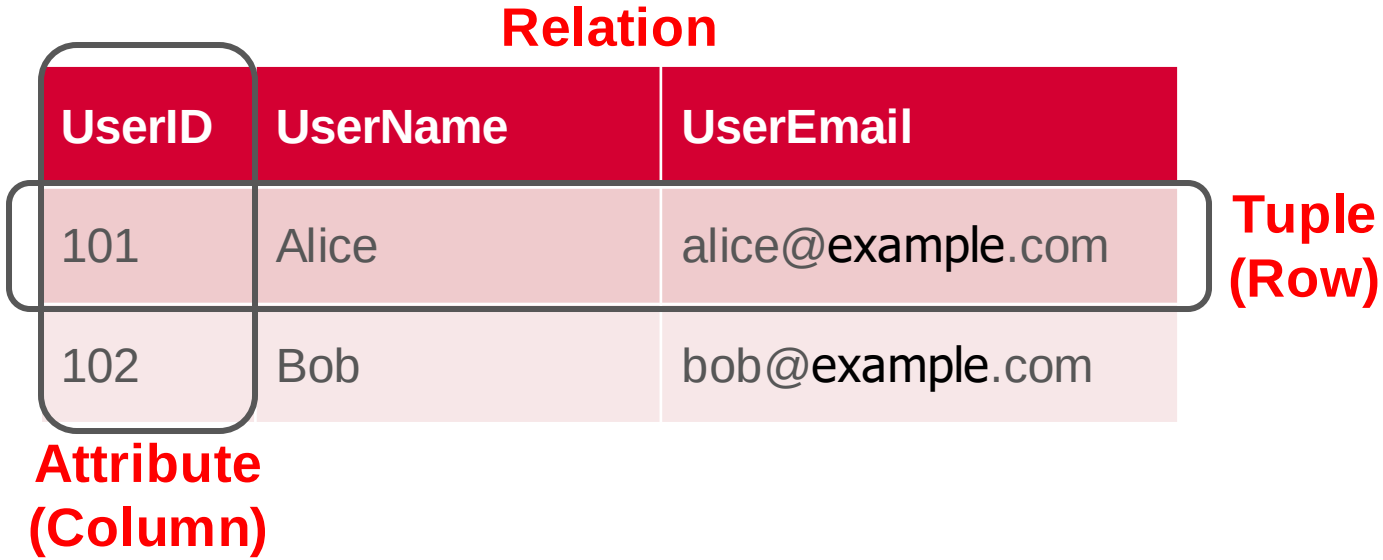
GPU

- NVIDIA GeForce RTX 4070, 8 GB GDDR6
- 5888 CUDA cores
- Total 94,208 threads
- Suitable for parallel workloads
- Access to dedicated VRAM



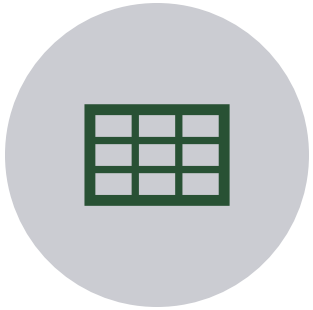
Recap to Relational Data

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



- 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

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 (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
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

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
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

$$User \bowtie Order$$

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

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

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

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.

Dataset: Stanford large network dataset collection

Graph	Type	Nodes	Edges
p2p-Gnutella09	Directed	8,114	26,013
p2p-Gnutella04	Directed	10,876	39,994
Skitter	Undirected	1,696,415	11,095,298
roadNet-CA	Undirected	1,965,206	2,766,607
roadNet-TX	Undirected	1,379,917	1,921,660
roadNet-PA	Undirected	1,088,092	1,541,898
SF.cedge	Undirected	1,74,955	2,23,001
cal.cedge	Undirected	21,048	21,693
TG.cedge	Undirected	18,263	23,874
OL.cedge	Undirected	6,105	7,035

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

Join on CPUs

Processing large amounts of data is a challenging task

Radix hash joins and sort-merge joins have been widely discussed and compared in previous studies

Modern HPC architectures have led to advancements in distributed join algorithms

Most of the multi-core implementations use Message Passing Interface (MPI)

Fundamental part of fixed-point iterative algorithm such as Transitive Closure (TC)

Join on GPUs

GPUs have high memory bandwidth and hierarchy for faster data access

GPUs offer parallel architecture for faster join operation

Data divided into smaller subsets for parallel processing

GPU-based join operation improves performance and scalability

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, and Sudhakar Yalamanchili. Red fox: An execution environment for relational query processing on gpus. In Proceedings 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 processing unit (GPGPU). PhD thesis, Georgia Institute of Technology, 2015.

Join on GPUs: Relational Learning Framework

Expedites rule coverage on GPUs for healthcare records data

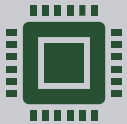
Outperforms 75% of applications over multi-core CPU systems

Duplicate tuples not efficiently managed and GPU memory overflows

- Carlos Alberto Mart´ınez-Angeles, Haicheng Wu, In ̂es 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

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.

Modern Implementations of Datalog

- LogicBlox
- Soufflé
- RadLog
- PRAM
- SLOG

LogicBlox

Gradual advances in logic programming: Advancement over the years

LogiQL language: Datalog-like, null-free

LogicBlox system: Mature, enterprise-integrated

Massive financial benefits: $300B in revenue for business analytics

- Molham Aref, Balder ten Cate, Todd J. Green, Benny Kimelfeld, Dan Olteanu, Emir Pasalic, Todd L. Veldhuizen, and Geoffrey Washburn. Design and implementation of the logicblox system. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD '15, page 1371–1382, New York, NY, USA, 2015. Association for Computing Machinery

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

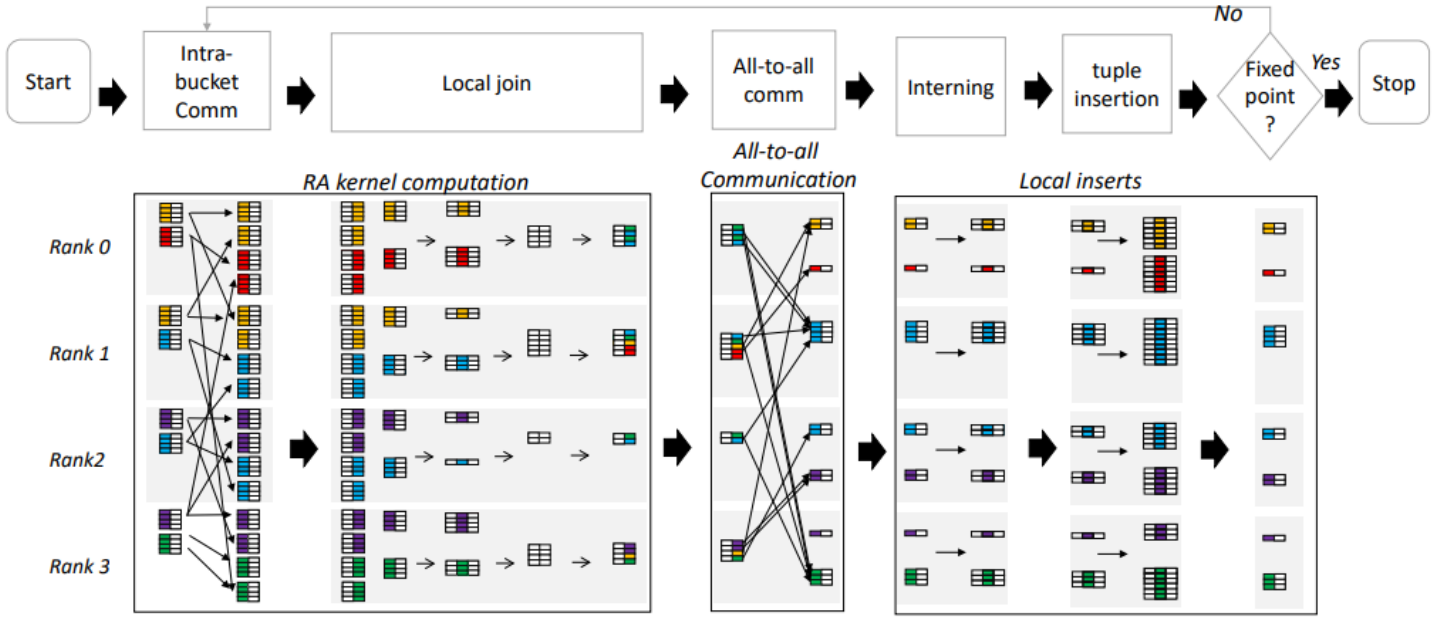
Supports limited number of threads for task-level parallelism

Cannot provide data parallelism

- Herbert Jordan, Bernhard Scholz, and Pavle Subotić. Soufflé: On synthesis of program analyzers. In International Conference on Computer Aided Verification, pages 422–430. Springer, 2016.
- 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.

SLOG

- High-performance implementation providing a compiler, REPL, and runtime
- Uses PRAM as relational algebra backend
- Outperforms Soufflé and RadLog for higher core counts
- Does not yet utilize GPU devices



- Thomas Gilray, Arash Sahebolamri, Sidharth Kumar, and Kristopher Micinski. Higher-order, dataparallel structured deduction. arXiv preprint arXiv:2211.11573, 2022.
- Thomas Gilray, Sidharth Kumar, and Kristopher Micinski. Compiling data-parallel datalog. InProceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction, pages23–35, 2021.

Join on CPUs: Fault Tolerant Hash Joins

Nasibullin and Novikov developed a distributed fault-tolerant hash-join

Utilizes data replication and alive signals for fault detection and recovery

Can handle a single point of failure with a keeper-worker model

Does not outperform distributed hash join algorithms in a failure-free situation

Can be criticized for data skewness and lack of accelerator use

- Arsen Nasibullin and Boris Novikov. Fault tolerant distributed hash-join in relational databases. In Fifth Conference on Software Engineering and Information Management (SEIM-2020), pages 11–17, 2020.

Datalog on GPUs

Outperforms non-GPU systems for join and transitive closure computation

Use GPU kernels to eliminate unnecessary tuples, join, and project

Achieved up to 40x performance enhancement over single CPU system

Limitation in relation size and applicability to distributed platforms

- Carlos Alberto Mart´ınez-Angeles, In ̂es Dutra, V´ıtor Santos Costa, and Jorge Buenabad-Ch´avez. Adatalog engine for gpus. In Declarative Programming and Knowledge Management: DeclarativeProgramming Days, KDPD 2013, Unifying INAP, WFLP, and WLP, Kiel, Germany, September 11-13,2013, Revised Selected Papers 0, pages 152–168. Springer, 2014.
- Grigoris Antoniou, Sotiris Batsakis, Raghava Mutharaju, Jeff Z. Pan, Guilin Qi, Ilias Tachmazidis,Jacopo Urbani, and Zhangquan Zhou. A survey of large-scale reasoning on the web of data. TheKnowledge Engineering Review, 33, 2018.

Persistent Homology (PH)

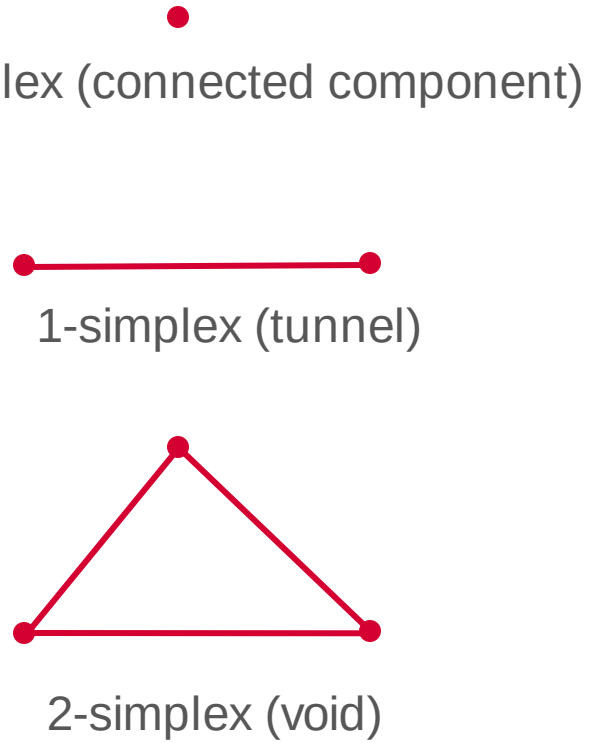
1
1
0

Investigates connections between different parts of complex networks using simplicial complexes

Simplicial complexes represent topological spaces as collections of simplices (points, line segments, triangles, etc.)

Detects topological features and measures the significance of relationships across multiple thresholds

Topological features refer to the 0-, 1-, 2-dimensional homology groups of a metric space describing connected components, tunnels, and voids



- 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

Barcodes and Persistent Diagram

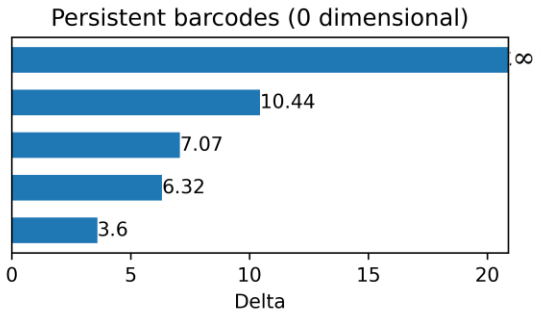
1
1
1
A

PH can track when features are created and destroyed with varying scales

Quantifies birth and death of features in the graph according to their significance

Represented in the form of barcodes, a fingerprint for a graph

Persistent diagram: points in 2D plane obtained from barcodes, represent topological feature persistence with x as birth time and y as death time



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

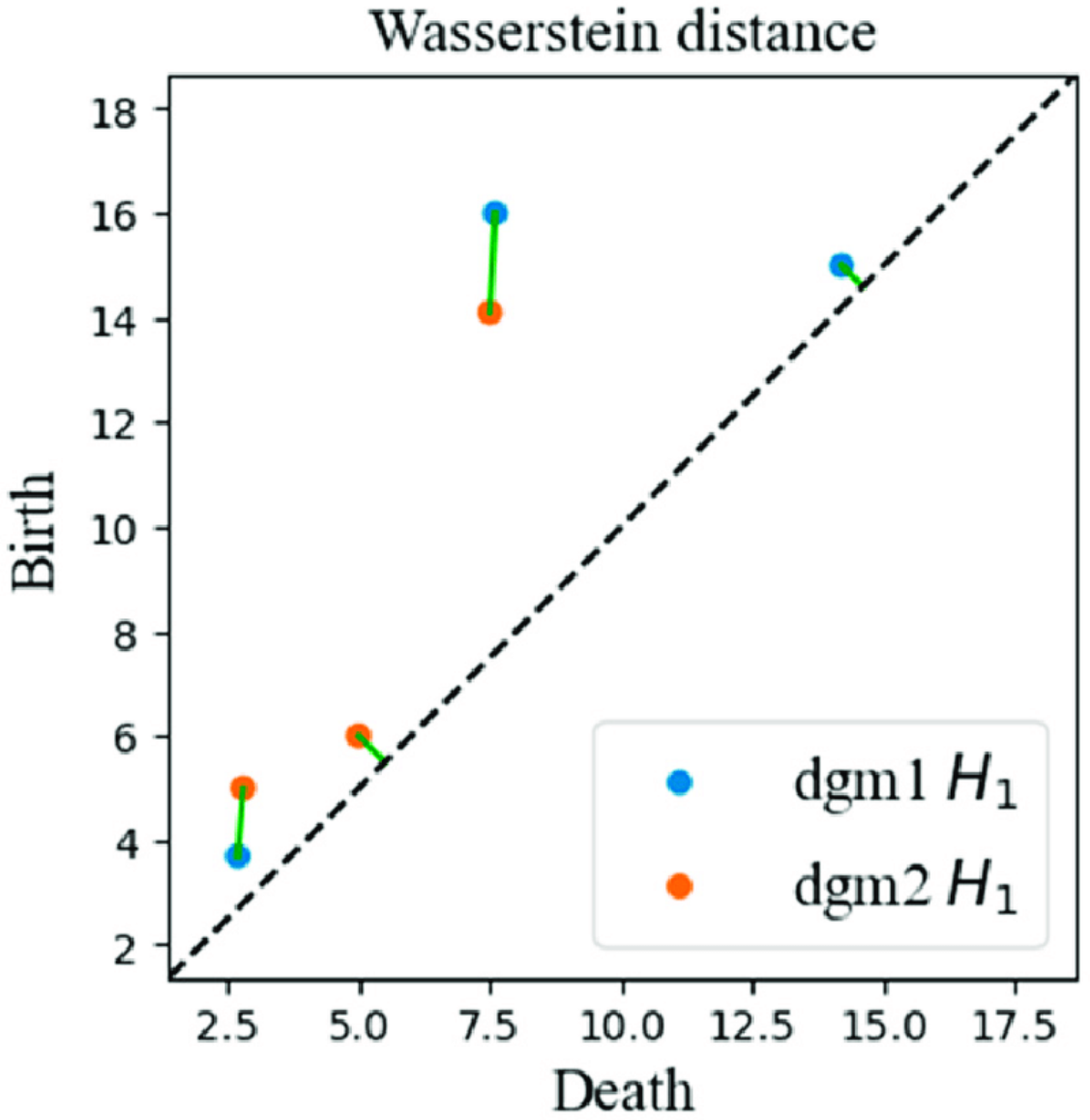
- Aktas, M.E., Akbas, E. & Fatmaoui, A.E. Persistence homology of networks: methods and applications. Appl Netw Sci 4, 61 (2019). https://doi.org/10.1007/s41109-019-0179-3

Wasserstein distance (WD

Wasserstein distance (Earth Mover's Distance) measures distance between two probability distributions

Minimum amount of work required to transform u into v, work = distribution weight x distance

Minimum value achieved by a perfect matching between the features of two Persistent Diagrams



- S. Vallender, "Calculation of the wasserstein distance between probability distributions on the line," Theory of Probability & Its Applications, vol. 18, no. 4, pp. 784–786, 1974.
- H. Edelsbrunner, "Persistent homology: theory and practice," 2013.
- Qingzhang, Shi. (2021). Time Series Clustering with Topological and Geometric Mixed Distance. Mathematics. 9. 10.3390/math9091046.

Experimental data and network construction

Structural T1-weighted and rs-fMRI data were obtained from the publicly available Enhanced Nathan Kline Institute Rockland Sample database (NKI-RS)

The MRI data from 3T Siemens Magnetom Tim Trio scanner

The rs-fMRI data were acquired from each subject using 3 acquisition protocols with 3 temporal sampling rates (645ms, 1400ms, 2500ms)

Total number of subjects: 316 with mean time series from 113 brain regions

One weighted network per fMRI scan

- K. B. Nooner, S. Colcombe, R. Tobe, M. Mennes, M. Benedict, A. Moreno, L. Panek, S. Brown, S. Zavitz, Q. Li et al., "The nki-rockland sample: a model for accelerating the pace of discovery science in psychiatry," Frontiers in neuroscience, vol. 6, p. 152, 2012.

Embed FCNs in metric space

To map the association between two nodes u and v in the brain network, their Pearson's correlation coefficients corr(u, v) are used

The objective is to transform this association into a distance metric where stronger correlations between nodes correspond to shorter distances

Mapping to all 316 subjects across 3 temporal frequencies:

$$d(u, v) = 1 - corr(u, v)$$

Included both positive and negative correlations

Generates symmetric adjacency matrix with size 113 x 113 using Pearson's correlation

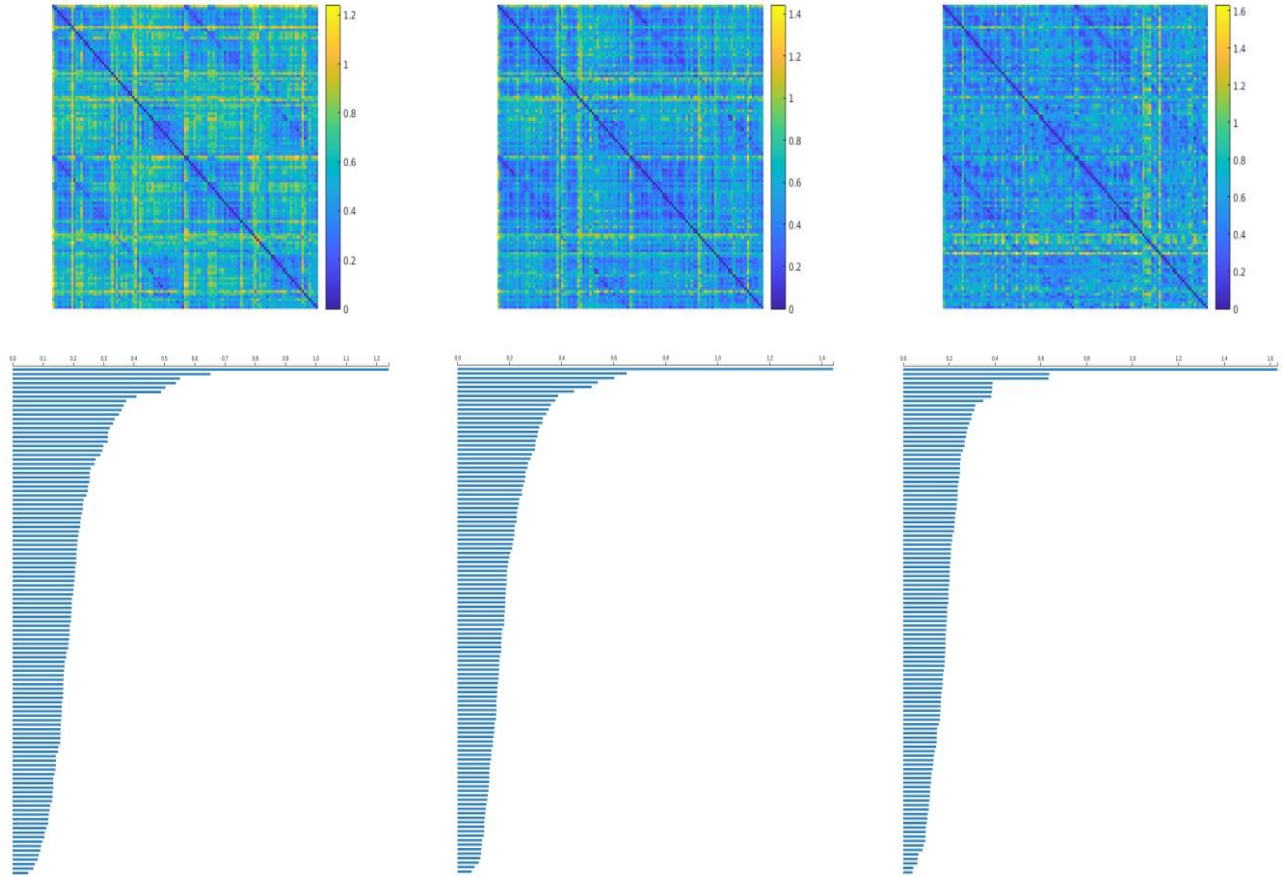
Total adjacency matrices: 948 (316 x 3)

- B. Cassidy, C. Rae, and V. Solo, "Brain activity: Conditional dissimilarity and persistent homology," in 2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI), 2015, pp. 1356–1359.
- S. M. Smith, K. L. Miller, G. Salimi-Khorshidi, M. Webster, C. F. Beckmann, T. E. Nichols, J. D. Ramsey, and M. W. Woolrich, "Network modelling methods for fmri," Neuroimage, vol. 54, no. 2, pp. 875–891, 2011.

Apply PH to metric space

Apply PH on the metric space to extract topological features

Compute 0-dimensional barcodes for all FCNs (3 x 316)



FCN (top-row) and extracted topological feature (bottom-row) for subject 31 for 645ms (left), 1400ms (center), and 2500ms (right)

- The GUDHI Project, GUDHI User and Reference Manual. GUDHI Editorial Board, 2015. [Online]. Available: http://gudhi.gforge.inria.fr/doc/latest/
- M. Kerber, D. Morozov, and A. Nigmetov, "Geometry helps to compare persistence diagrams," Journal of Experimental Algorithmics (JEA), vol. 22, pp. 1–20, 2017.

Statistical analysis on PDs

Convert extracted barcodes to persistent diagram (PD)

Quantify the difference between two PDs using Wasserstein distance

Wasserstein metrics provides the foundation of the statistical inference framework

Performed 3 kinds of statistical analysis:

- Across cohort study
- Within cohort study
- Comparision with traditional FCN analysis

- S. Vallender, “Calculation of the w asserstein distance between probability distributions on the line,” Theory of Probability & Its Applications, vol. 18, no. 4, pp. 784–786, 1974.
- H. Edelsbrunner, “Persistent homology: theory and practice,” 2013.

Across Cohort Study

Perform comparison of PDs across cohorts using Wasserstein distance

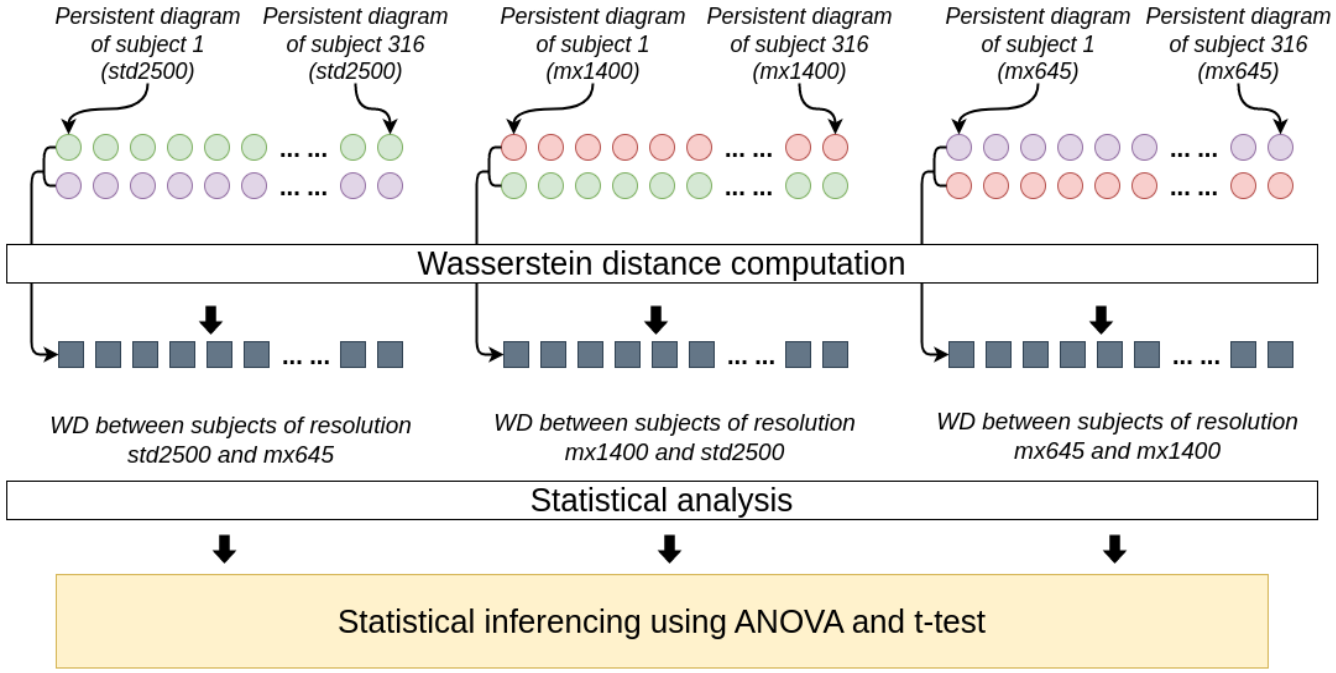
For each subject compute 3 WD

- 2500ms and 645ms
- 1400ms and 2500ms
- 645ms and 1400ms

Form 3 distributions each with 316 datapoints

Apply Analysis of Variance (ANOVA) and t-test on the distributions

High p-value (>0.05) will indicate the null hypothesis is true



Pipeline to compare WD across cohorts

- R. T. Shinohara, J. Oh, G. Nair, P. A. Calabresi, C. Davatzikos, J. Doshi, R. G.Henry, G. Kim, K. A. Linn, N. Papinutto et al., "Volumetric analysis from a harmonized multisite brain mri study of a single subject with multiple sclerosis," American Journal of Neuroradiology, vol. 38, no. 8, pp. 1501–1509, 2017.

Within Cohort Study

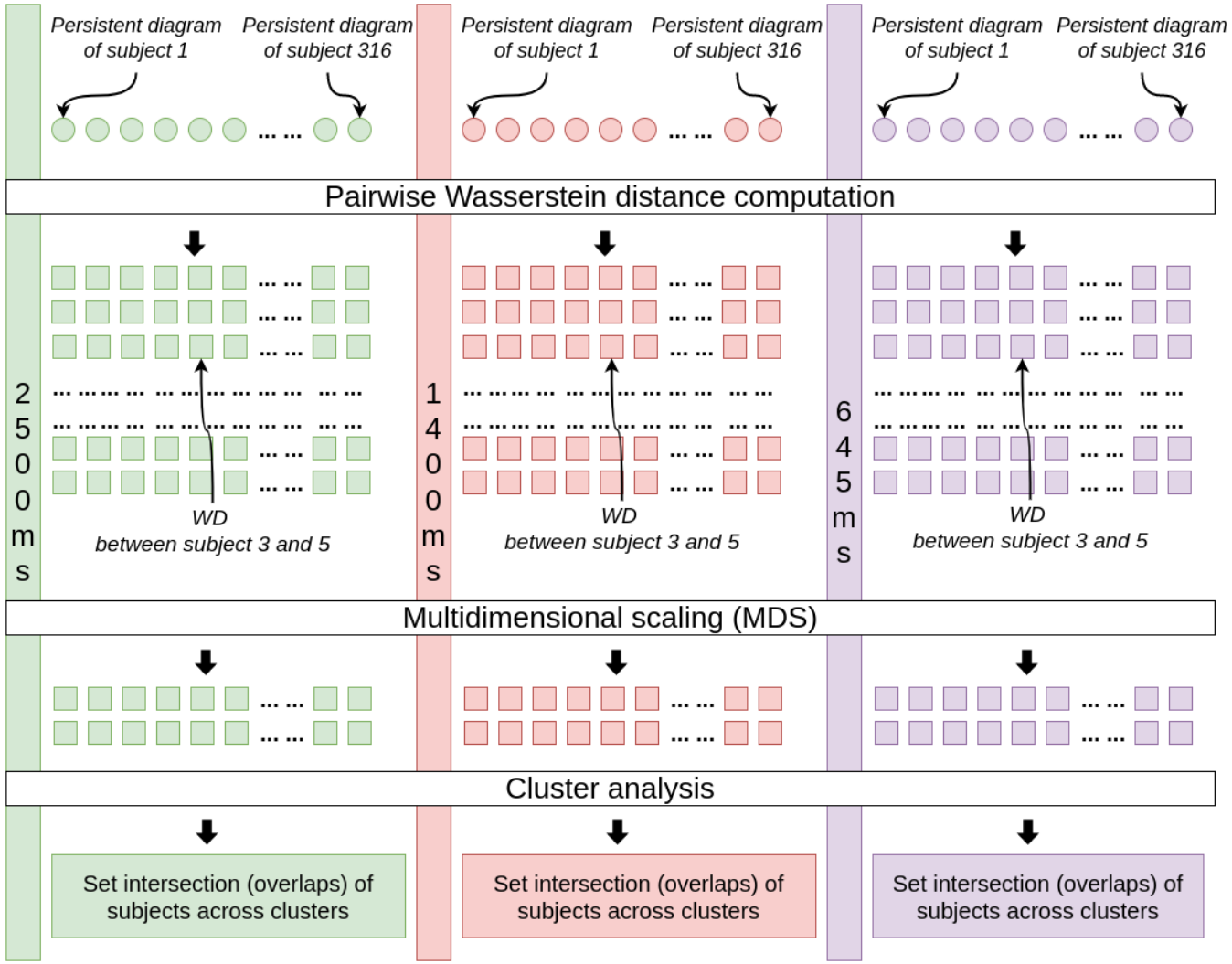
Perform comparison of pairwise WD between PDs of subjects within each cohort

Generate 3 adjancency matrix of size 316 x 316

Reduce the dimension using 2-components MDS (matrix size 316 x 2)

Apply K-Means clustering with Silhoutte coefficient

Computes set interaction of the subjects across clusters



Pipeline to compare WD within a given cohort

- Chung, M. K., Huang, S. G., Carroll, I. C., Calhoun, V. D., & Goldsmith, H. H. (2022). Dynamic Topological Data Analysis for Brain Networks via Wasserstein Graph Clustering. arXiv preprint arXiv:2201.00087.

Comparison with traditional FCN analysis

Use raw FCN instead of using PH features

Perform ANOVA test across all subjects for each node of the raw FCN

Compute 1 p-value per entry of the FCN

As FCN size is 113 x 113, generates 113 x 113 p-values

Threshold the p-values with significance level of $\alpha = 0.05$

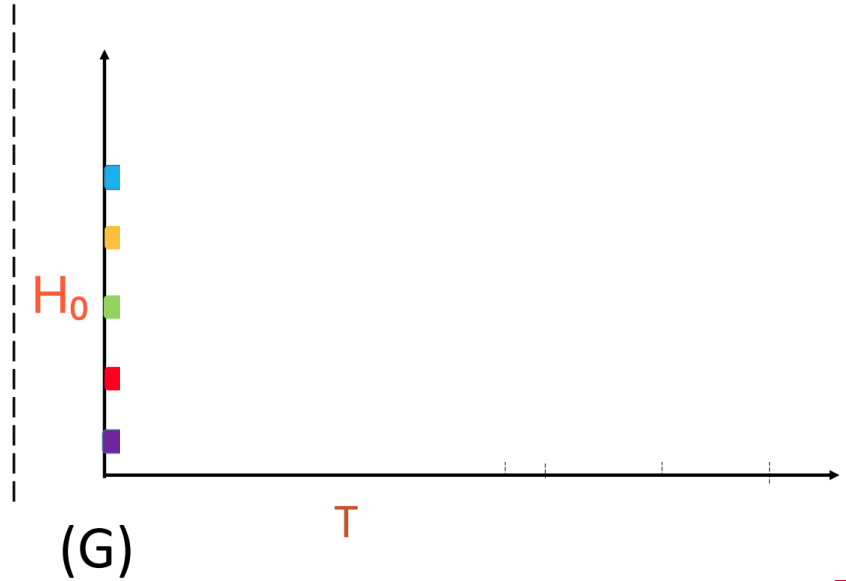
- Assign 1 if p-value is less than the threshold, 0 otherwise

If majority of the value are 1, then it will indicate that FCN acquired from different sampling periods are different

- Feng, W., Liu, G., Zeng, K., Zeng, M., & Liu, Y. (2022). A review of methods for classification and recognition of ASD using fMRI data. Journal of neuroscience methods, 368, 109456.

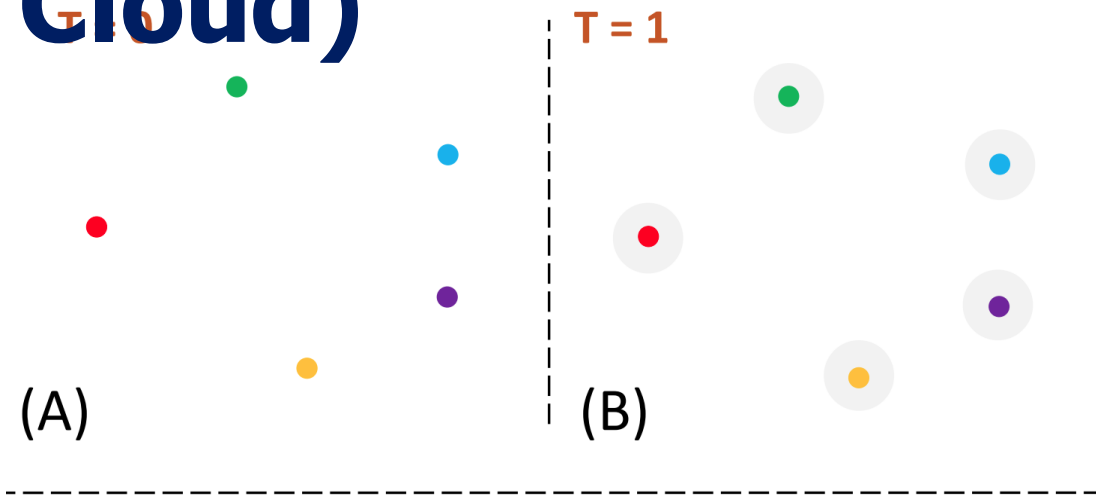
Computing Persistent Homology (Point Cloud)

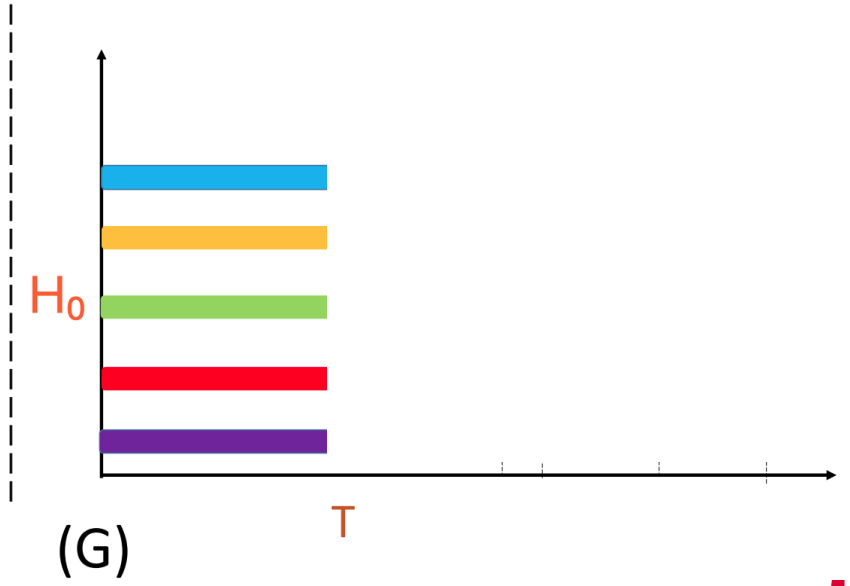
(A)



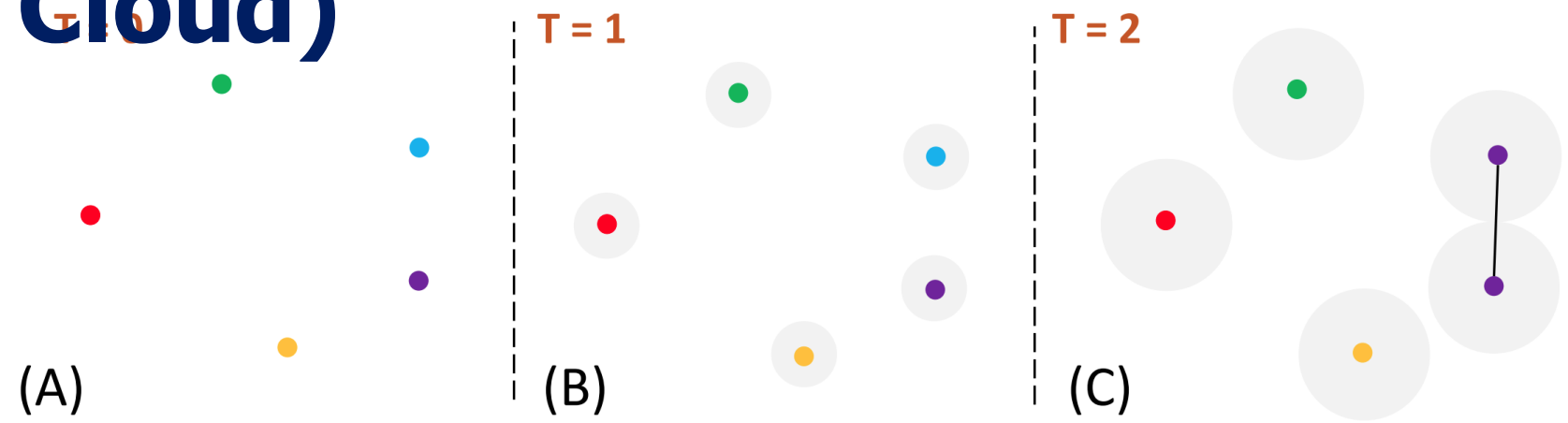
(G)

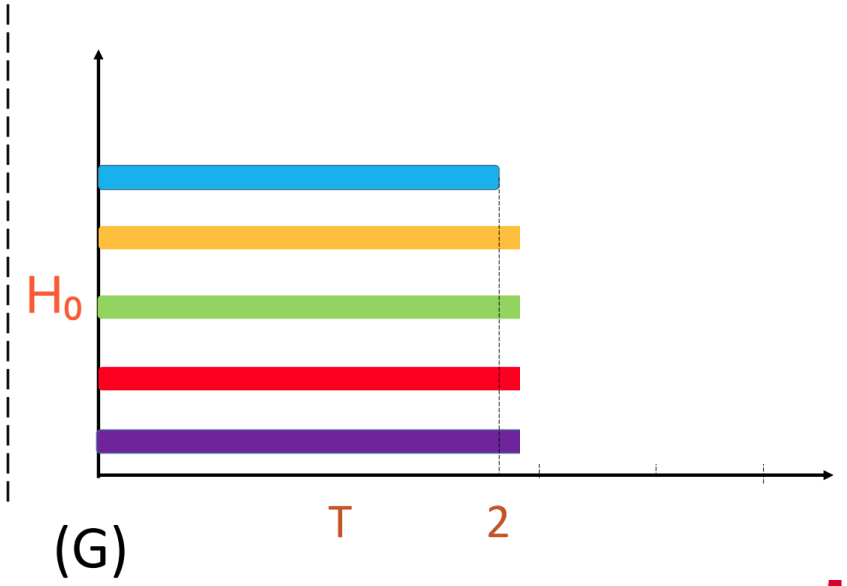
Computing Persistent Homology (Point Cloud)



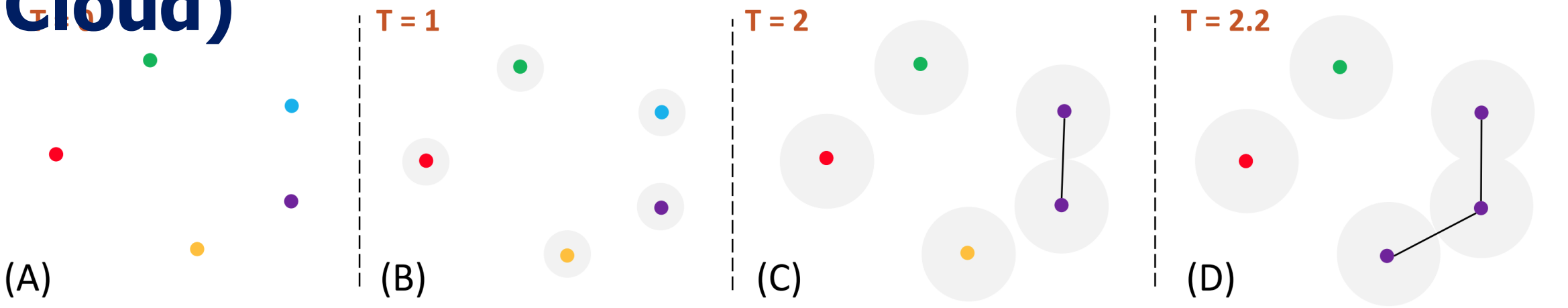


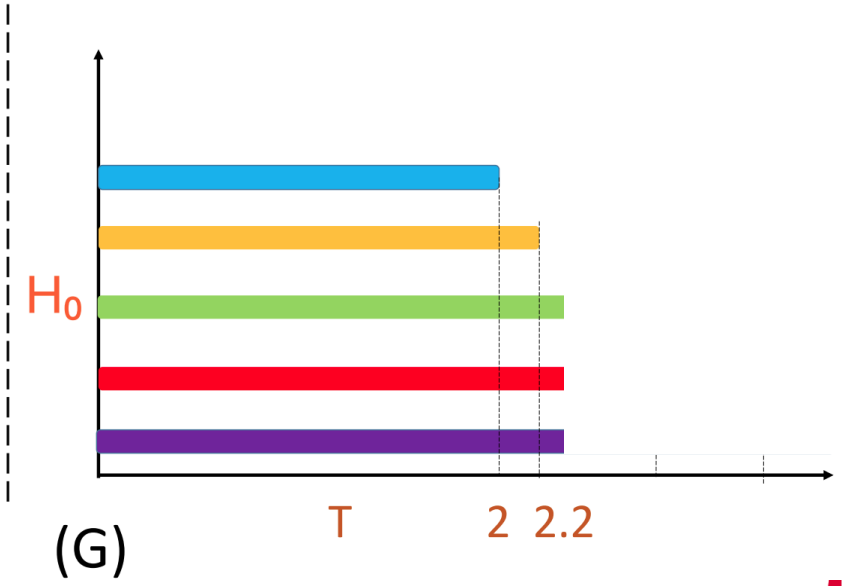
Computing Persistent Homology (Point Cloud)



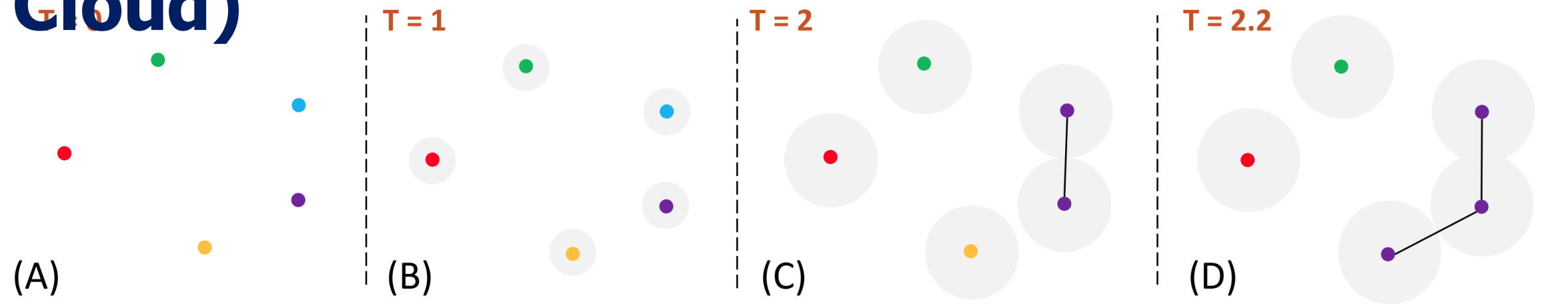


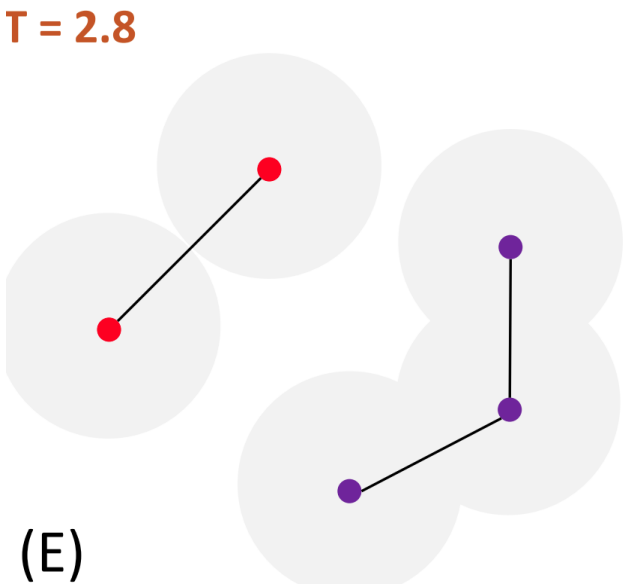
Computing Persistent Homology (Point Cloud)

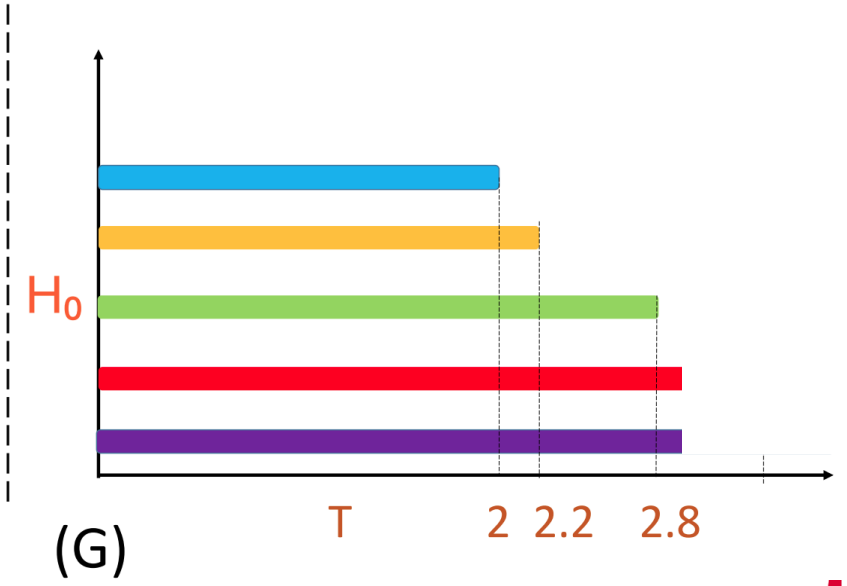




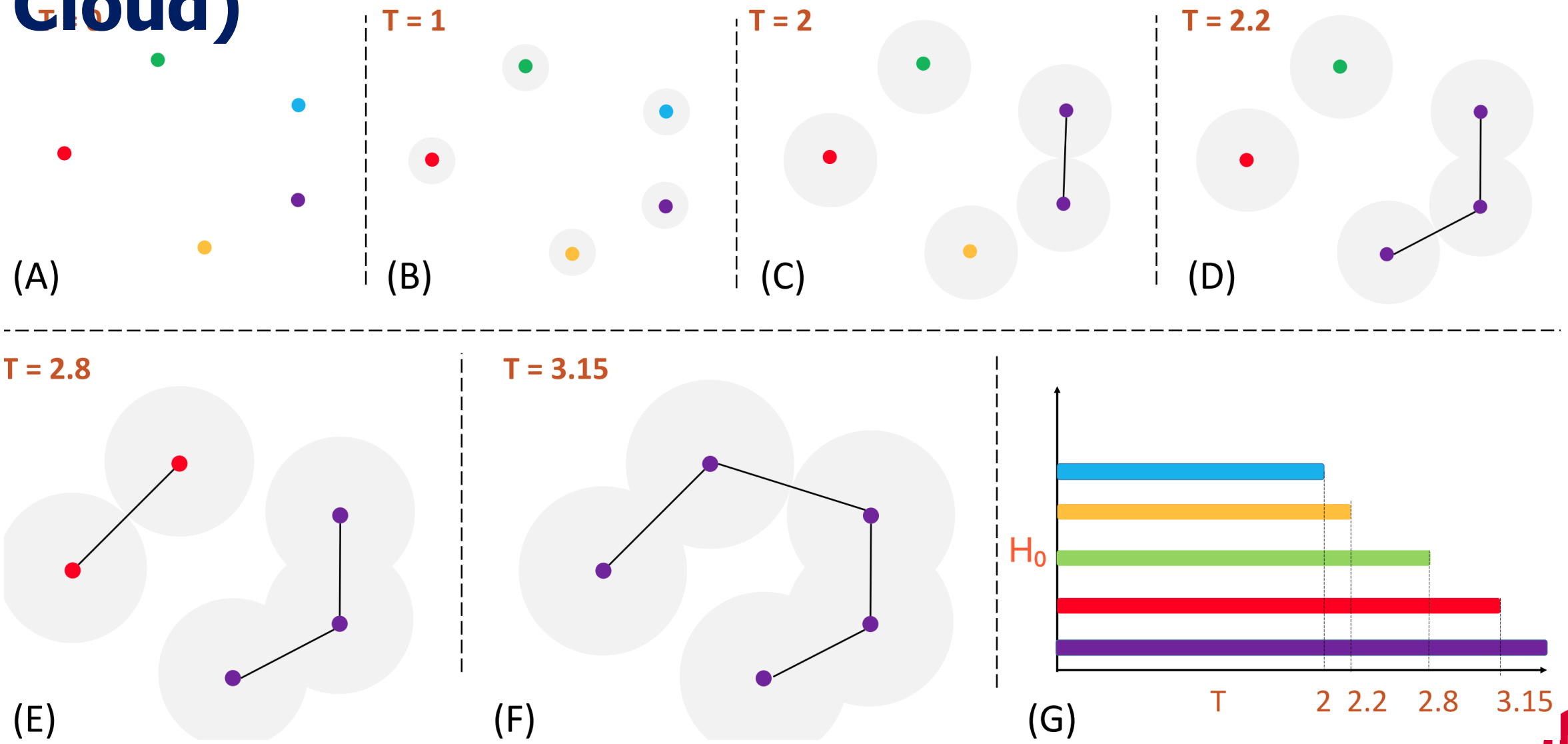
Computing Persistent Homology (Point Cloud)







Computing Persistent Homology (Point Cloud)

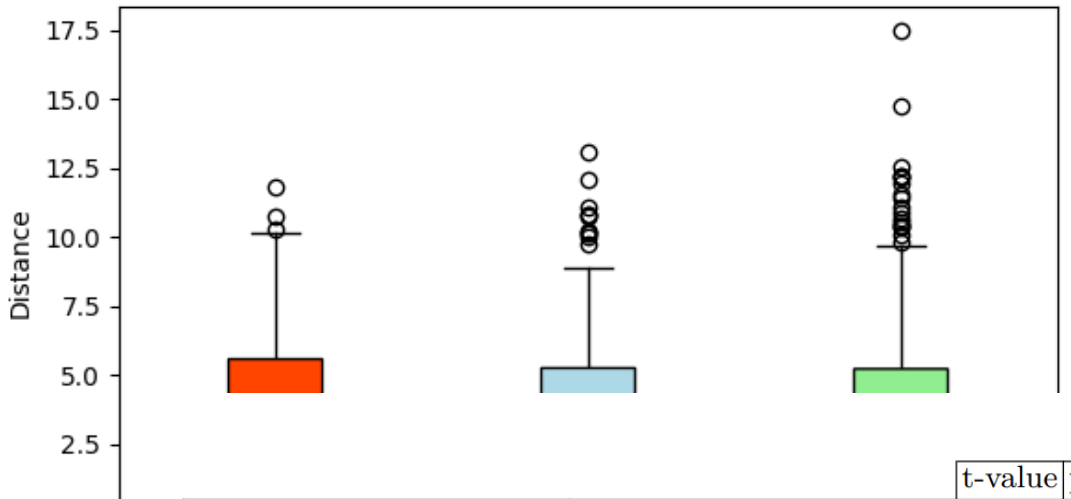


Across Cohort Study Results

Boxplots are aligned indicating similarity of the three distributions

ANOVA test yields p-value of 0.29

T-values and p-values are greater than the significance level of α = 0.05



		t-value	p-value
WD(TR=645, TR=1400)	WD(TR=1400, TR=2500)	**0.059**	**0.088**
WD(TR=1400, TR=2500)	WD(TR=2500, TR=645)	**0.460**	**0.519**
WD(TR=2500, TR=645)	WD(TR=645, TR=1400)	**0.286**	**0.387**

Boxplot and t-test results for across study, significance level of α = 0.05

Functional Connectivity Networks Analysis

FCNs use Pearson's correlation to measure time series association

Multi-site rs-fMRI studies recruit faster, larger samples

Sensitive to data acquisition parameters (sampling rates)

Non-neural variability negates advantages of 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.
- 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.

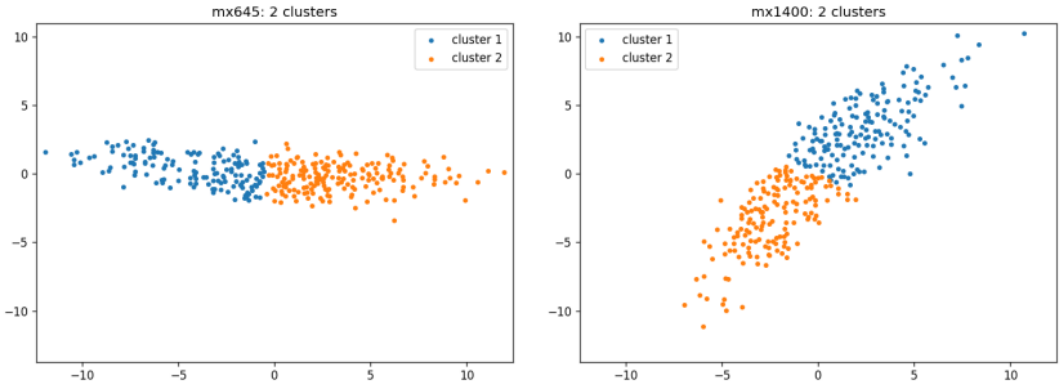
Within Cohort Study Results

Apply K-Means clustering with Silhoutte score on 2-components MDS, 3 matrices (316 X 2)

198 out of 316 subjects reside in same cluster group in all 3 TRs

251 out of 316 subjects reside in same pairwise cluster intersection

Compared against random FCNs which which gave p-value of 0



(a) Clustering result on MDS data for TR=645ms.

(b) Clustering result on MDS data for TR=1400ms.

(c) Clustering result on MDS data for MDS plots of three data cohorts

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.

Exascale Supercomputers

Aurora (ANL)

Frontier (ORNL)

El Capitan (LLNL)

- Evans, T. M., Siegel, A., Draeger, E. W., Deslippe, J., Francois, M. M., Germann, T. C., ... & Martin, D. F. (2022). A survey of software implementations used by application codes in the Exascale Computing Project. The International Journal of High Performance Computing Applications, 36(1), 5-12.
- D. Kothe, S. Lee and I. Qualters, "Exascale Computing in the United States," in Computing in Science & Engineering, vol. 21, no. 1, pp. 17-29, 1 Jan.-Feb. 2019, doi: 10.1109/MCSE.2018.2875366.

Comparison with traditional FCN analysis results

Majority pixels are black corresponding to a p-value < significance level of α = 0.05

7731 connections show FCNs are different for 3 TRs while 5037 connections are not

Traditional FCN analysis fails to identify connections from same subjects with different TRs

Results after ANOVA test on raw FCNs of 316 subjects, white pixels indicate no matching of the same subject across different TRs

Table of Contents