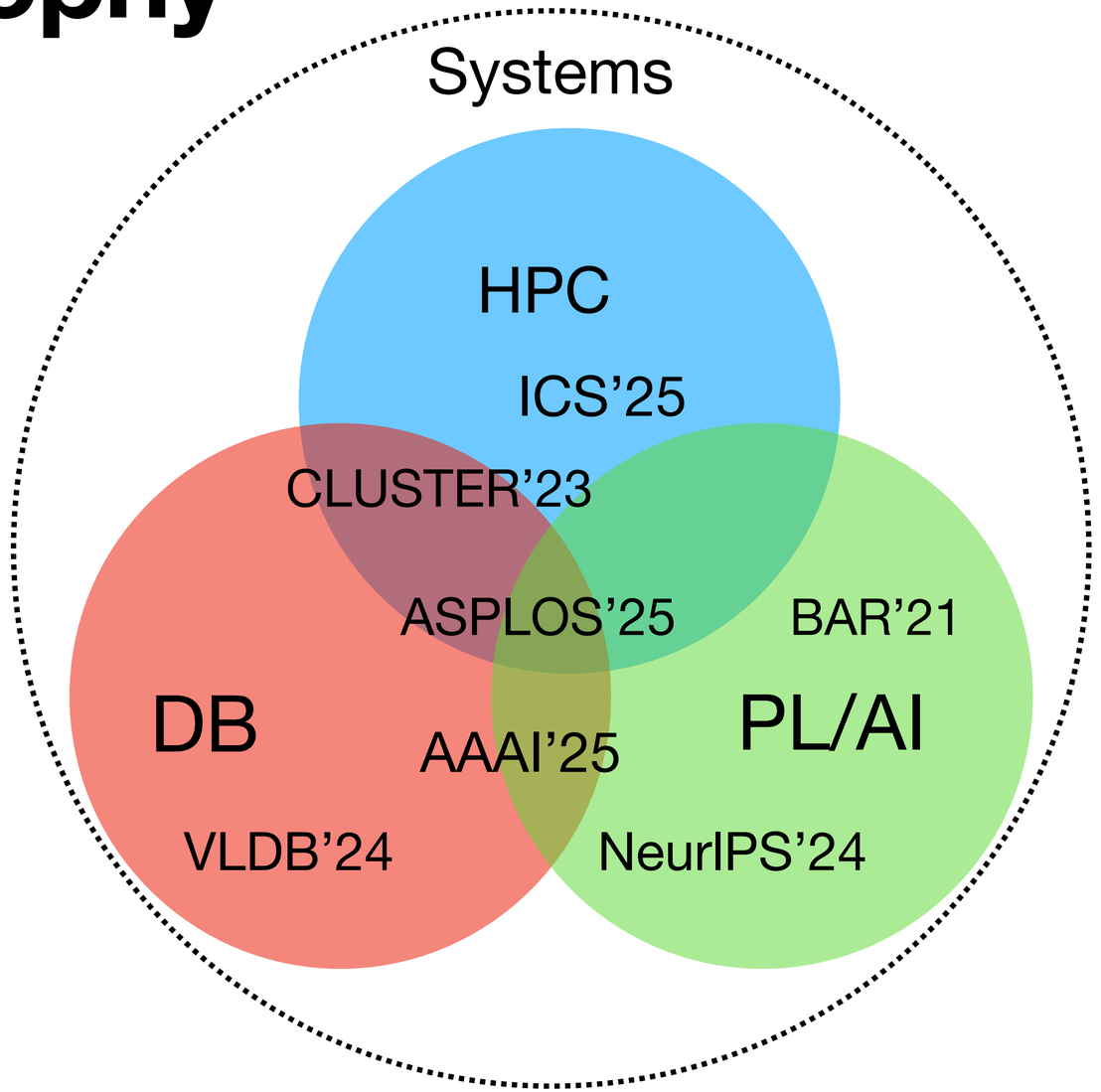
Journey to GPU Datalog

Yihao Sun Syracuse University 2026.3.11

Research Area & Philosophy

- Focus on building high performance Datalog systems.
- Key is hardware and software co-design.
- High-quality research with reproducibility.



Datalog : Syntax

Database query language only has 2 syntax forms:

$$Fact(x_1, \dots).$$

First order predicate … holds (Extensional Database)

$$Head(\dots) \leftarrow Body_1(\dots), Body_n(\dots)$$

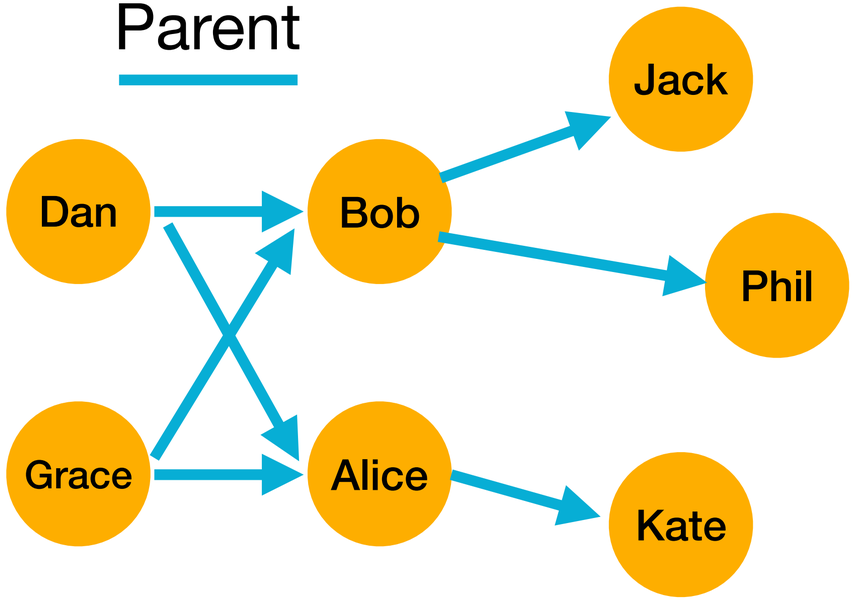
Horn clause

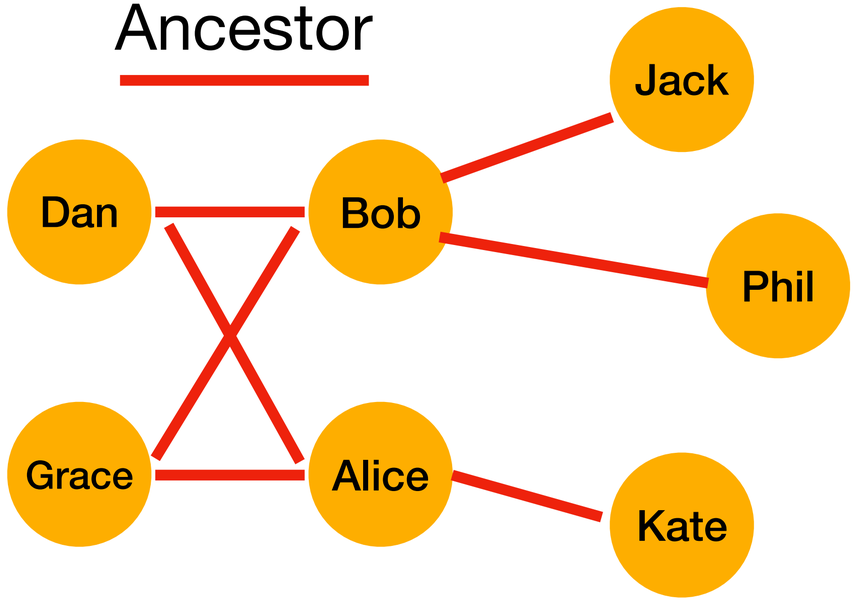
If all body clauses hold ,head clause is True
(Intentional Database)

Fixpoint semantics: Finding Ancestor

$$Ancestor(x, y) \leftarrow Parent(x, y)$$

$$Ancestor(x, z) \leftarrow Ancestor(x, y), Parent(y, z) .$$





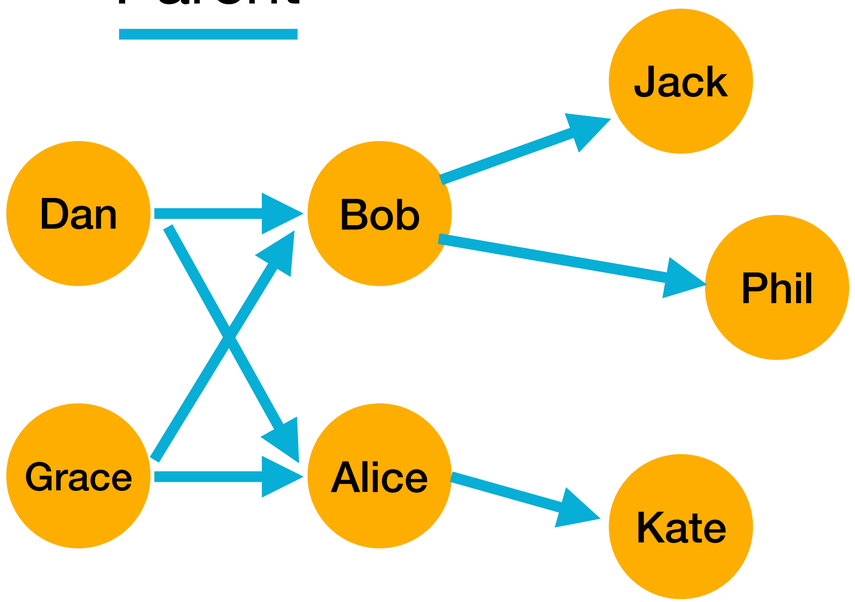
Fixpoint semantics: Finding Ancestor

$$Ancestor(x, y) \leftarrow Parent(x, y)$$

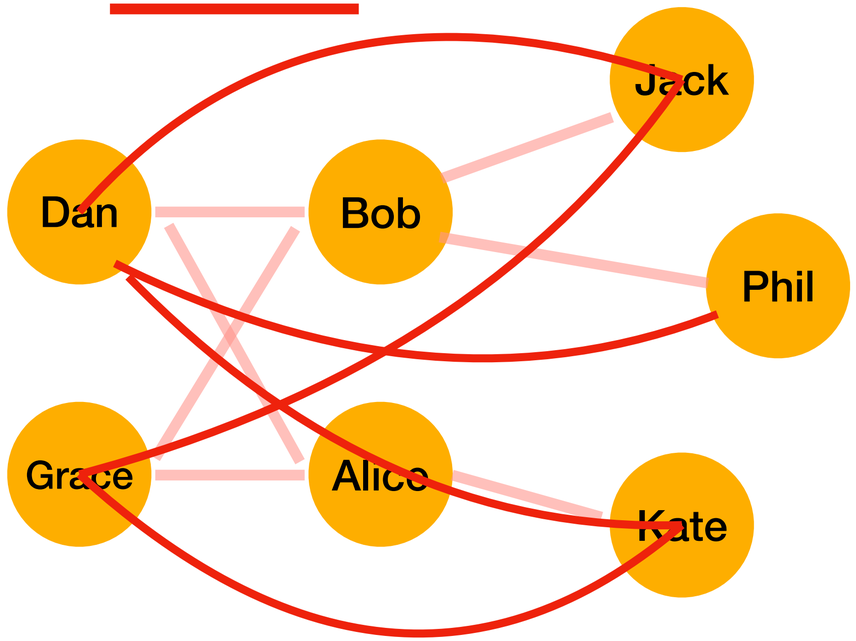
$$Ancestor(x, z) \leftarrow Ancestor(x, y), Parent(y, z) .$$

If we apply rules in 3rd iteration again no more tuples generated.

Parent



Ancestor



Datalog

$$Ancestor(x, y) \leftarrow Parent(x, y)$$

Recursion

$$Ancestor(x, z) \leftarrow Ancestor(x, y), Parent(y, z)\ .$$

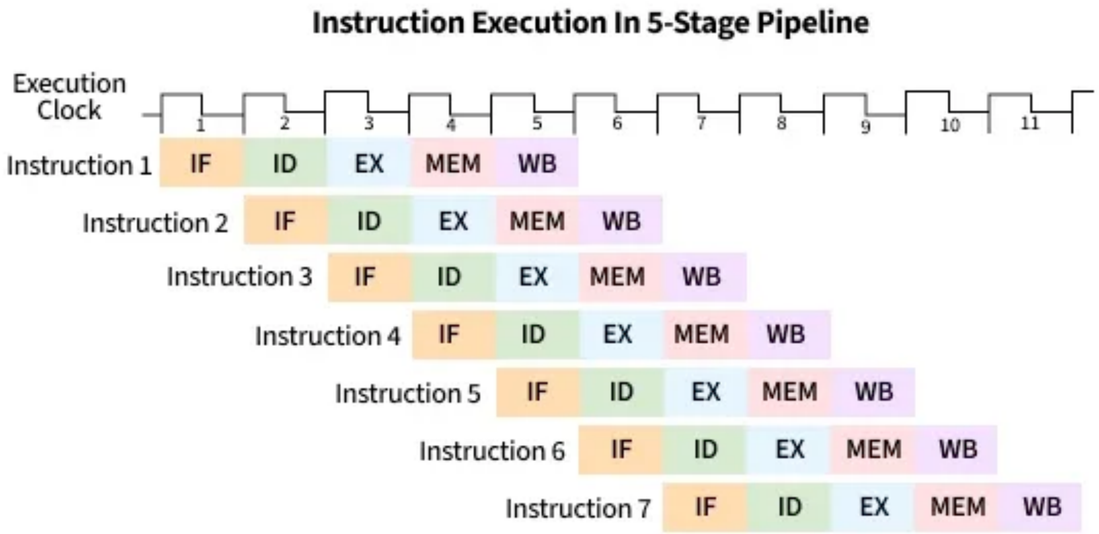
Translate to $\mathscr{RA}^{+}$

Fixpoint Operator!

$$Ancestor = \Pi_{x,y} Parent$$

$$Ancestor = \mathcal{O}(\Pi_{x,z}(Parent \bowtie_{y} Ancestor) \cup Ancestor)$$

CPU: Pipeline stall



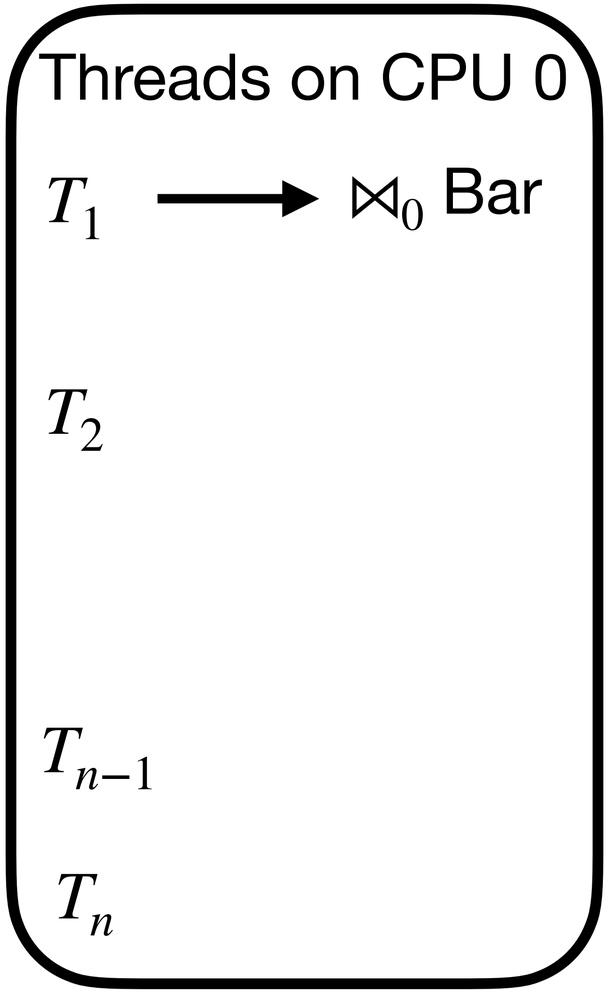
If a match return true, next instruction is body; If **not** match, ….

For (a, b) in relation **Foo**:
  if a in relation **Bar** 's column 1:
    for (a, c) in **Bar**:
      if c in relation baz :…:
        …..

c=1, c=100, c=1000 in relation baz, need access memory. DDR5 need 600 cycle to access!

CPU Threads and Latency

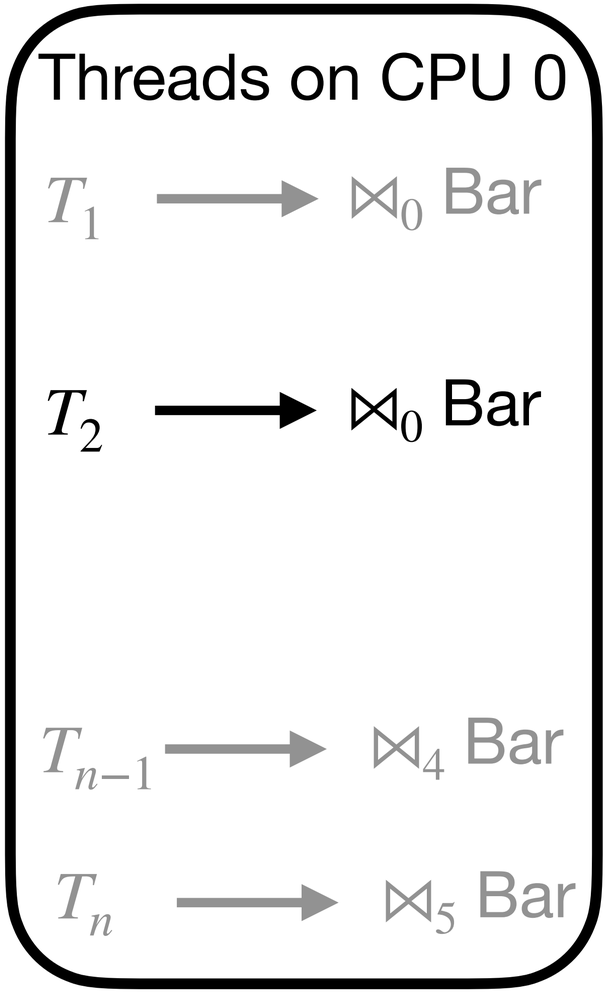
P	X
0	1
0	2
1	3
1	4
1	5
2	5
3	6
4	7
4	8
5	8



Stall caused by memory access

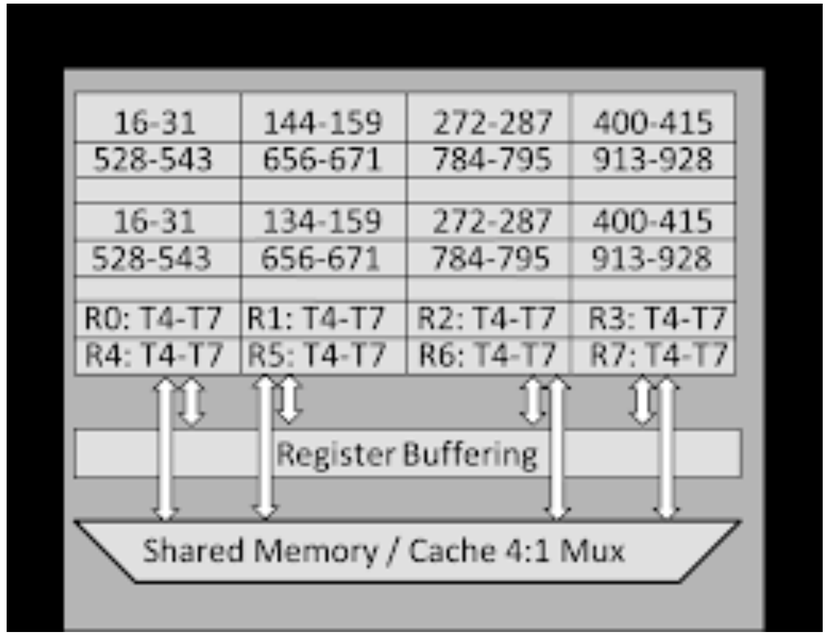
CPU Threads and Latency

P	X
0	1
0	2
1	3
1	4
1	5
2	5
3	6
4	7
4	8
5	8



Context switch to T2, but this can cause overhead.

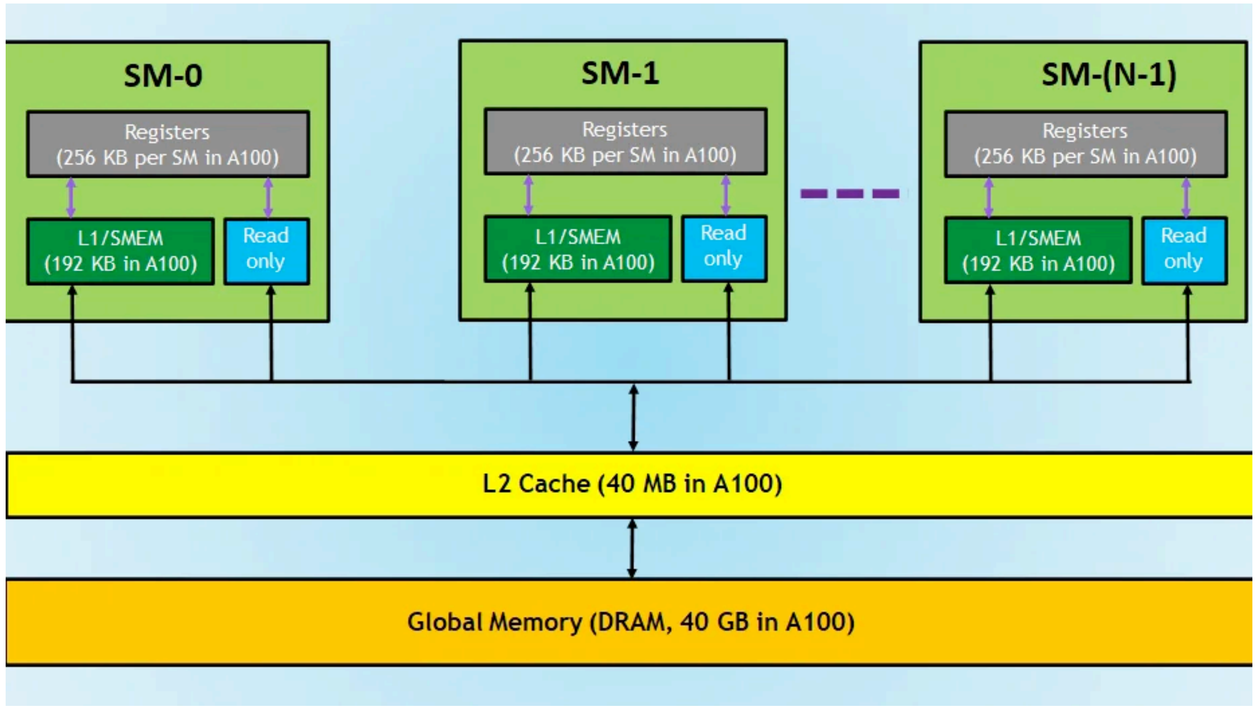
We can use more register to keep this cheap



The ideal CPU we want….

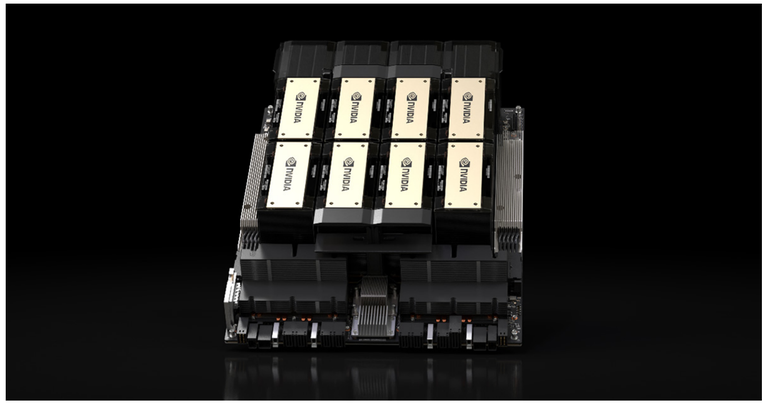
- Don’t have branch predication
- Can run multiple threads on the same CPU
- More registers for context switch
- More cache for memory speed
- Multiple threads can request memory at same time (more memory controller)

General Purpose GPU

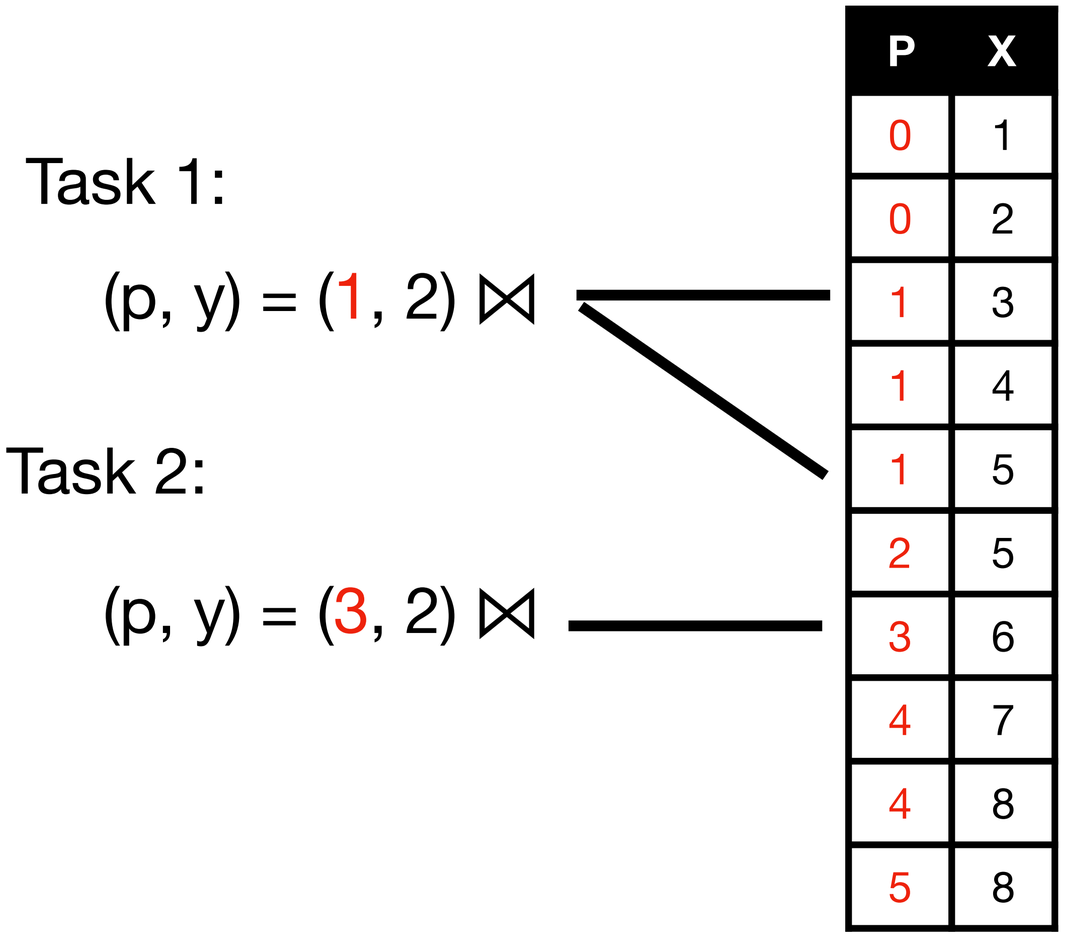


Migrating Datalog to GPU

- Most Datalog engines are CPU based.
- High bandwidth memory (H100 ~4TB/s) vs. 576GB/s(EPYC 9665)

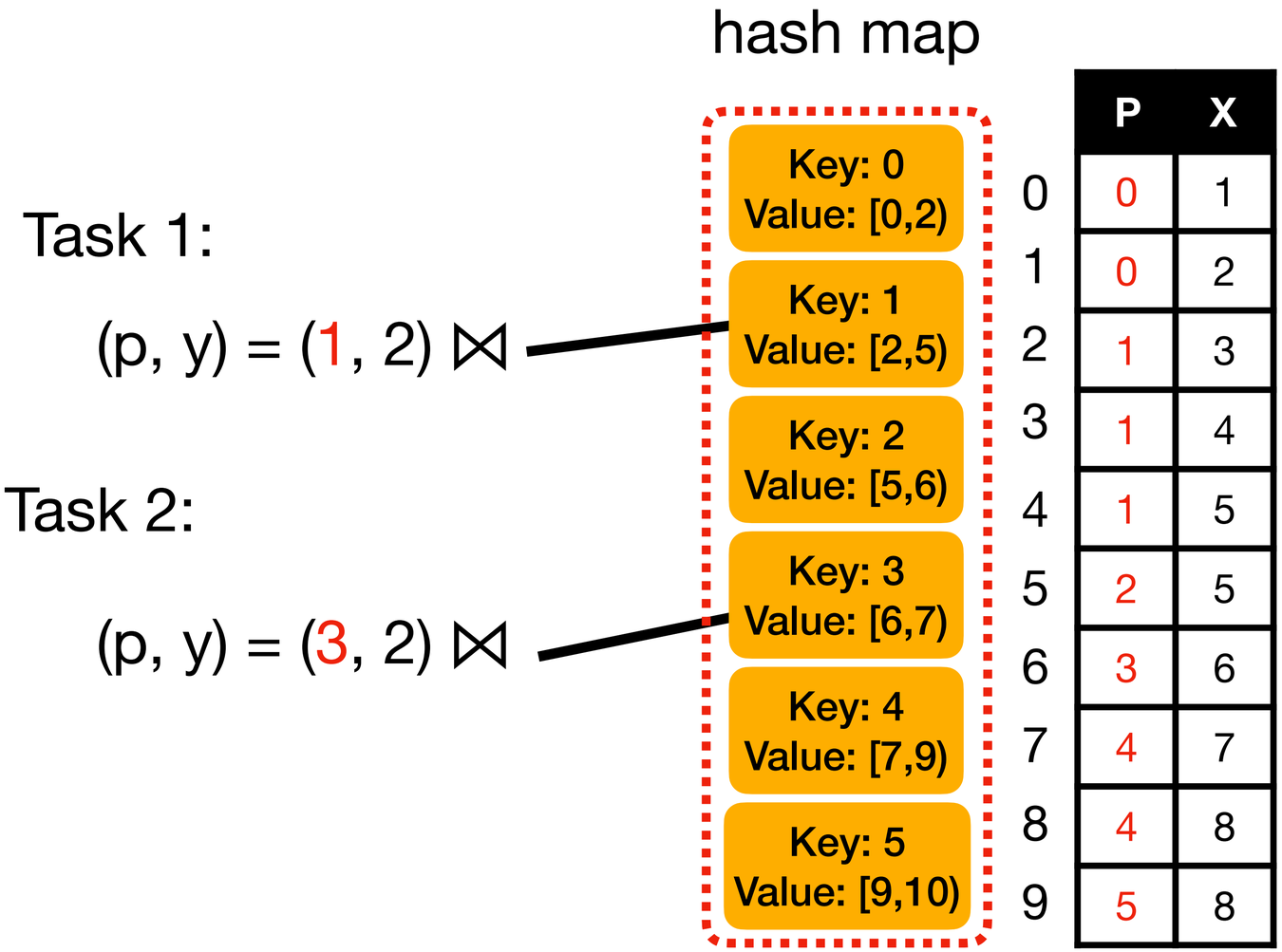


Challenge in Join : Index Data Structure



- Sort the right relation
- Join become range query
- GPUs excel at sorting, but the operation is memory-bound!
- However, GPUs perform poorly with binary search.

Challenge in Join : Index Data Structure



- Use a hash map for range query
- Hash table on GPU
- Unique key to range mapping
- GPUs are highly efficient with open-addressing hash maps

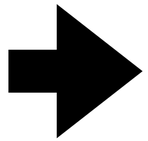
Permutation

	P	X
0	0	1
1	0	2
2	1	3
3	4	4
4	5	4

	P	X
5	2	5
6	3	6
7	4	7
8	4	8
9	5	8

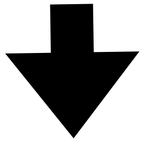
No in-place merge!

Allocate buffer
size = 2 * 10

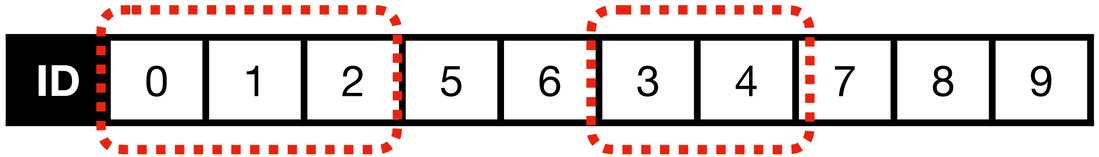


P	X
0	1
0	2
1	3
2	5
3	6
4	4
4	7
4	8
5	4
5	8

Allocate buffer
size = 10



Smaller buffer size, less memory spike

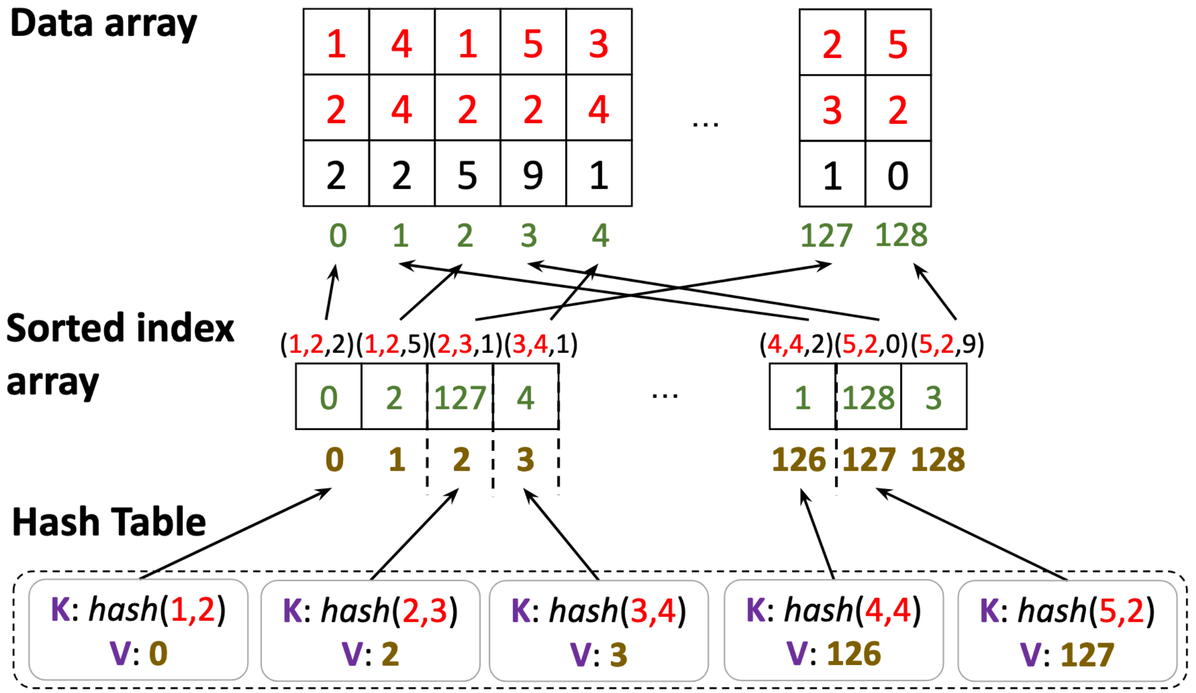


LSM like pointer chasing, need compaction after N iteration, but still 10 extra memory!

HISA: SIMT friendly Relation Storage

- Fast range query.
- Fast index merge.
- Open-addressing Hash with linear probing + Sorted compact data Array (HISA)

"Optimizing Datalog for the GPU"
ASPLOS 2025



GDlog vs. the SOTA

Dataset	SG size	Time (s)			
		GDlog	GDlog-HIP	Soufflé	cuDF
fe_body	408M	**5.05**	19.57	74.26	OOM
loc-Brightkite	92.3M	**3.42**	14.00	48.18	OOM
fe_sphere	205M	**2.36**	8.48	48.12	OOM
SF.cedges	382M	**5.54**	20.57	68.88	OOM
CA-HepTH	74M	**2.79**	5.92	20.12	21.24
ego-Facebook	15M	**1.23**	2.81	17.01	19.07

Same Generation (SG): GDlog vs. Soufflé (using 32 threads) and cuDF

- GDlog always fastest
- 10x more performance improvement against Soufflé
- Nvidia has performance better than AMD in similar theoretical performance.

Program Analysis Benchmark

Polonius: Rust's experimental borrow check in Datalog

Dataset comes from popular large rust project

Dataset	iter	VFlog	Souffle	Nemo	RDFox
clap-rs	1007	6.12	122	855.54	273
scallop-parser	179	0.49	18.57	72.25	40.1
scallop-runtim	297	0.45	7.28	35.9	17.2
cozodb-parse	713	1.76	13.85	63.54	19.5
cozodb-runtim	484	1.08	9.5	18.34	6.34
materialize-rei	2084	5.71	39.58	198.89	39.3
materialize-sir	2452	6.43	25.27	168.56	23.6
cairo-diagonis	185	0.31	7.64	15.17	7.03
cairo-hint	961	0.74	5.21	13.99	4.38
wgpu-glsl	1245	6.95	108.78	546.23	176
wgpu-msl	4528	13.35	67.04	601.4	59.4

Dense computation, accelerated very well.100x+ speed up

Very sequential, but GPU did still very well, 7x speed up!

Recap

- Background of Datalog
- Why we want migrating Datalog to GPU
- Data structure for GPU Datalog
- 10x speed up comparing to CPU based Datalog engine

Yihao Sun ysun67@syr.edu