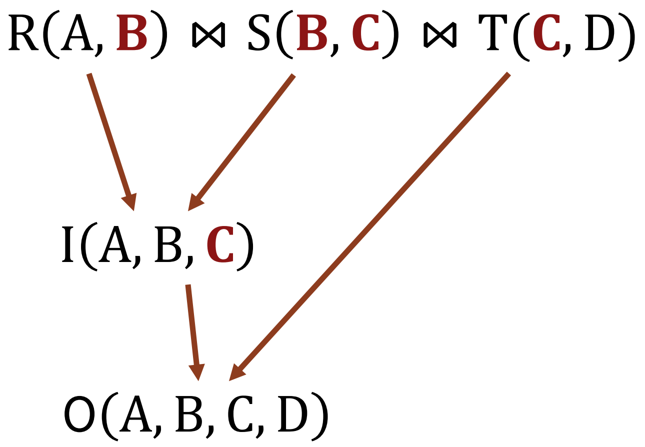
# Efficient Multiway Hash Join

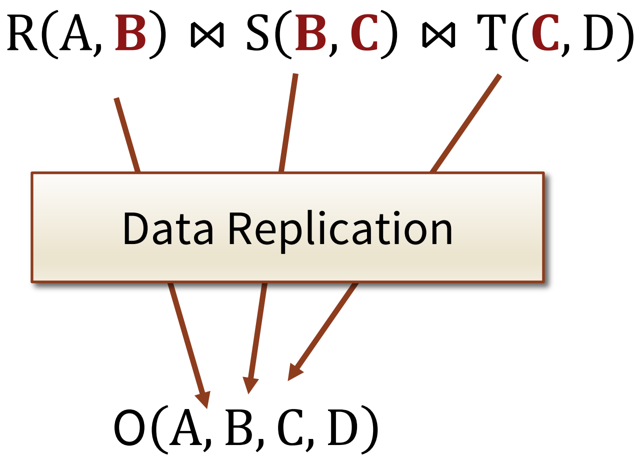
## Multiway Hash join on Reconfigurable Hardware

**Rekha Singhal , Yaqi Zhang ,** Jeffrey D. Ullman, Raghu Prabhakar and Kunle Olukotun

TATA CONSULTANCY SERVICES

# Parallel Join of Multiple Relations



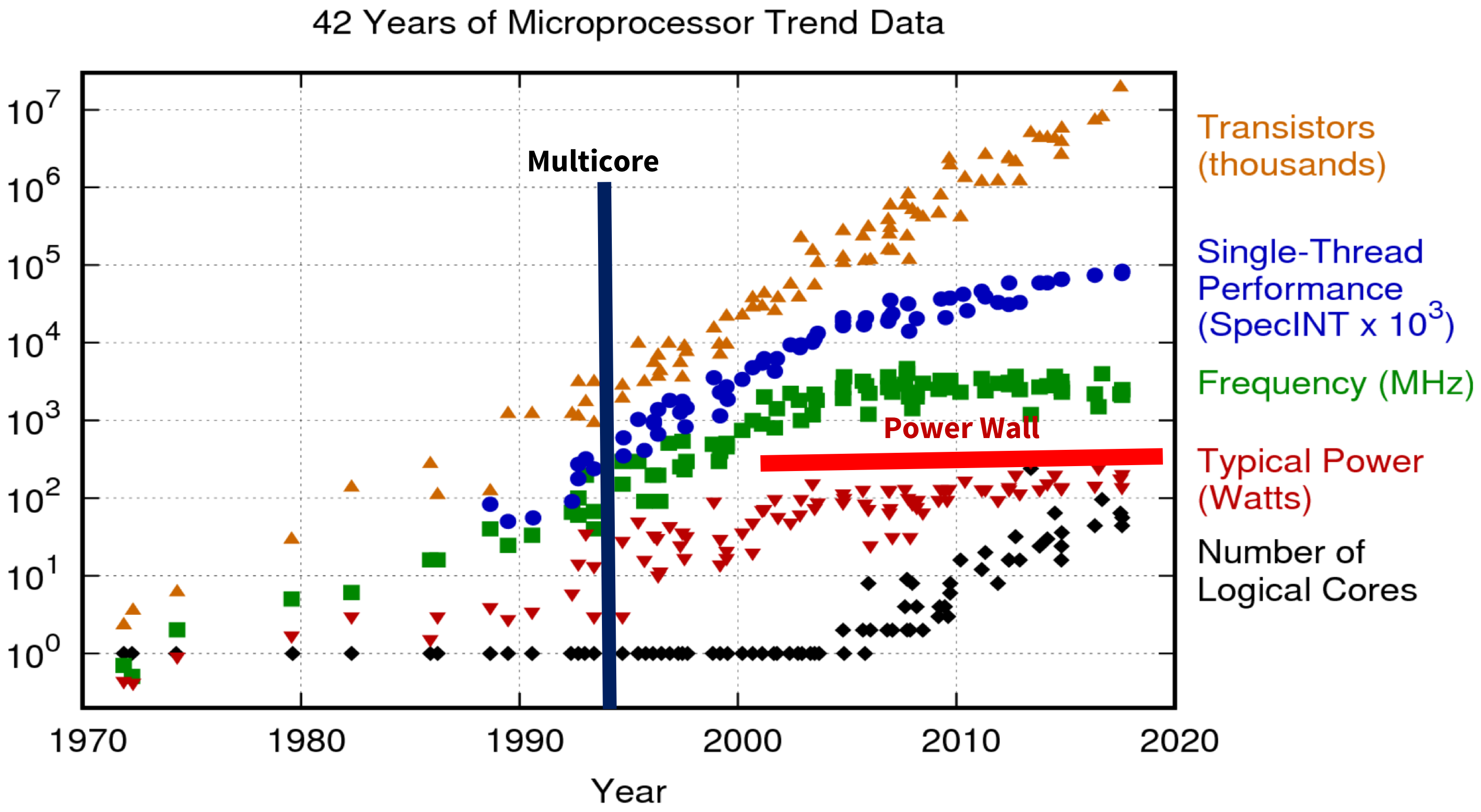

**Cascaded Binary Joins**

- Cost grows with size of ‘I’
- Performance bounded by memory bandwidth

**Multiway Joins**

- Cost grows with amount of replication to different processors
- Performance bounded by parallel compute and communication bandwidth

# Microprocessor Trends




Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

# Power and Performance

**Performance** **Energy Efficiency**

$$\text{Power} = \frac{\text{Op}}{\text{second}} \times \frac{\text{Joule}}{\text{Op}}$$

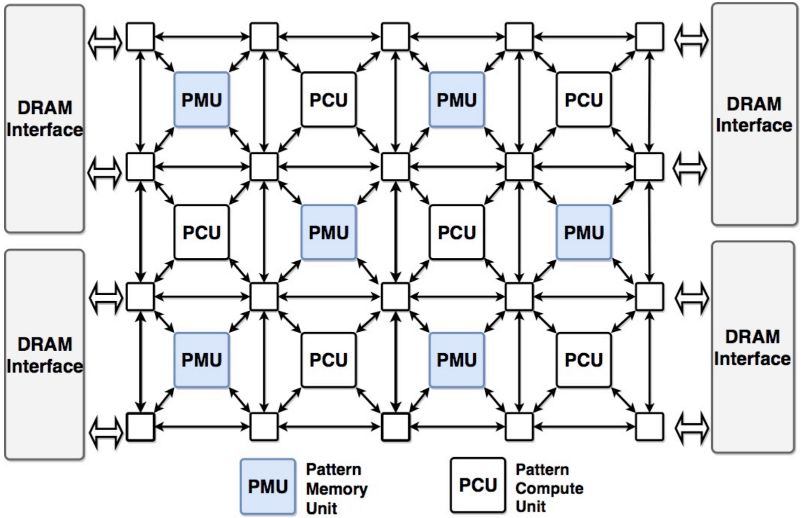
**Fixed**

↑ ↓

**Reconfigurable Hardware**

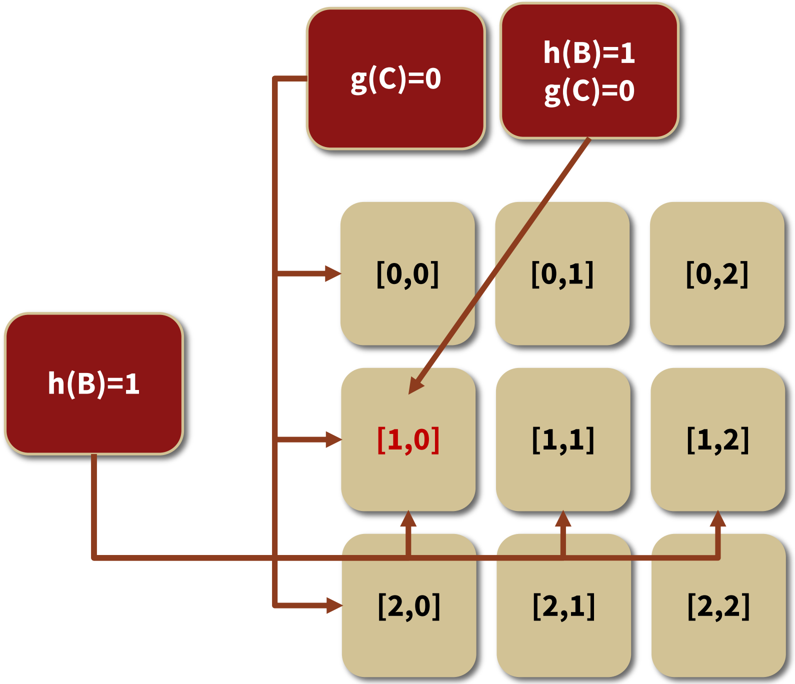
**e.g. FPGA, CGRA**

# Plasticine Architecture




- Coarse-grained reconfigurable architecture
- Application domains:
  - Linear algebra
  - Classic machine learning
  - Deep learning
  - Database
  - Networking
- 1GHz, 12.3 TFLOPS
- On-chip memory BW: 4TB/s
- On-chip network bisection BW: 1.5TB/s
- Power: 49W
- Area: 112.77 $mm^2$
- Technology: 28nm

# Multiway hash join in a cluster of processors

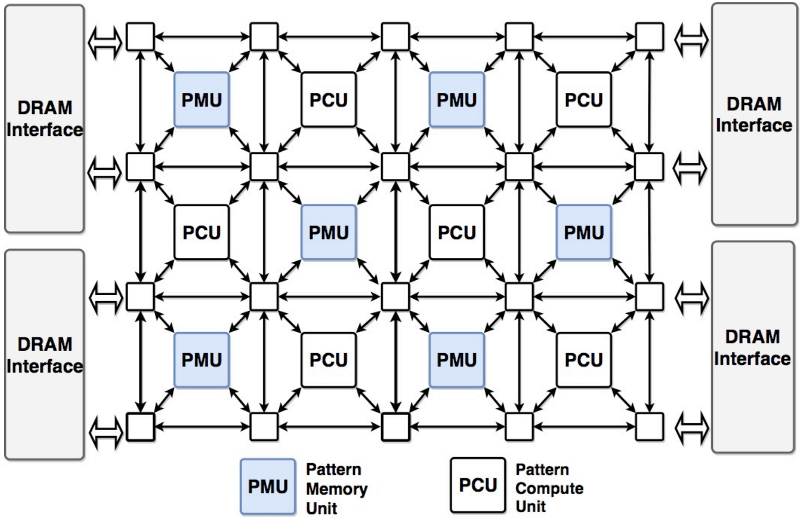



$\boxed{R(A, B)} \bowtie \boxed{S(B, C)} \bowtie \boxed{T(C, D)}$

- Hash on tuples with hash functions h(B) and g(C).
- Tuples of are distributed to processors according to (h(B), g(C))
- Parallelism on relation processing.
- **Cons:**
- Bounded by communication bandwidth

# Linear Join of 3 Relations on Plasticine Accelerator

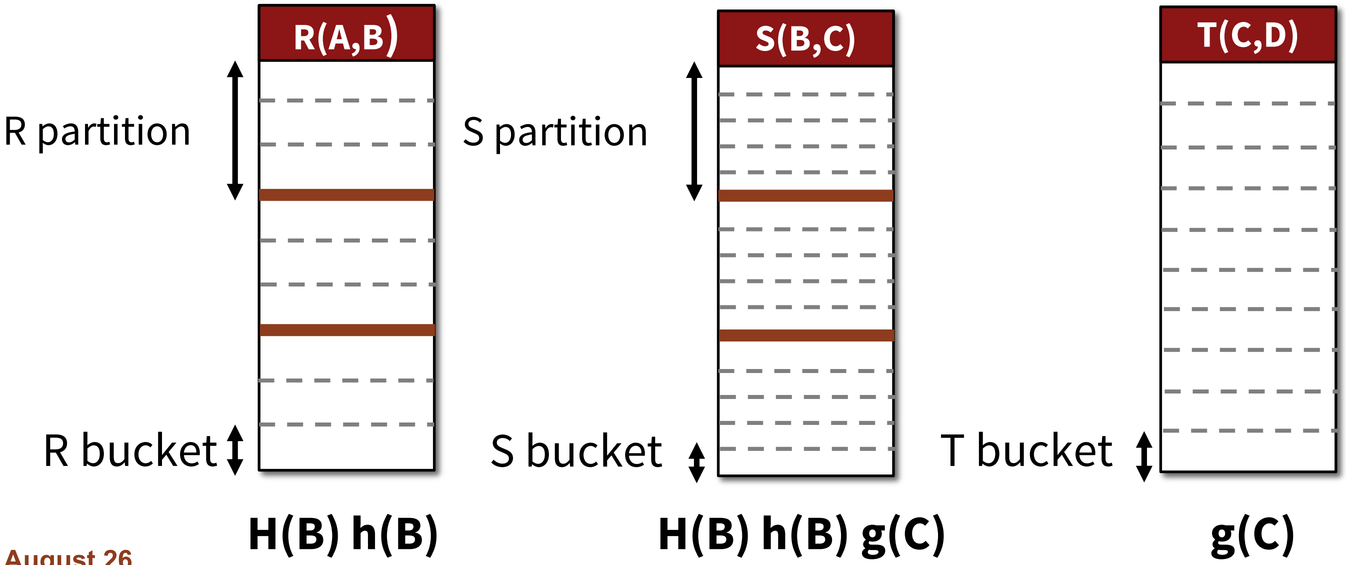
$$R(A, \mathbf{B}) \bowtie S(\mathbf{B}, \mathbf{C}) \bowtie T(\mathbf{C}, D)$$




- High compute throughput
- High on-chip network and memory bandwidth
- Explicit on/off-chip transfer
- Streaming communication

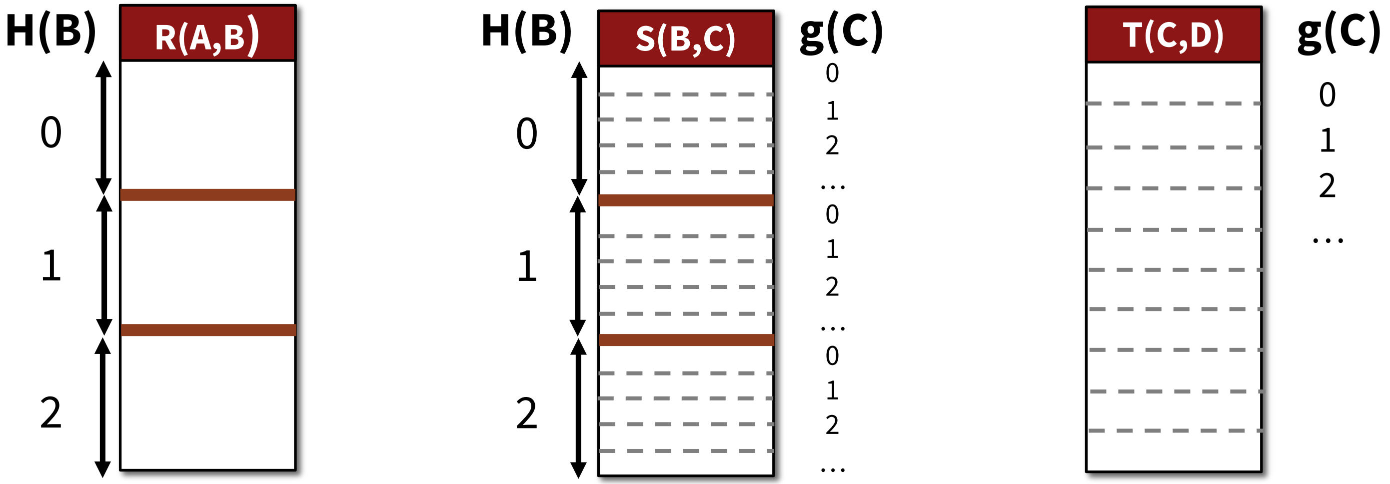
# Algorithm Highlight

- Outer-level hash function **H(B)** divides Table R and S into *partitions*
- Additional functions **h(B)** and **g(C)** hash *partitions* into *buckets* that gets joined on-chip in parallel.



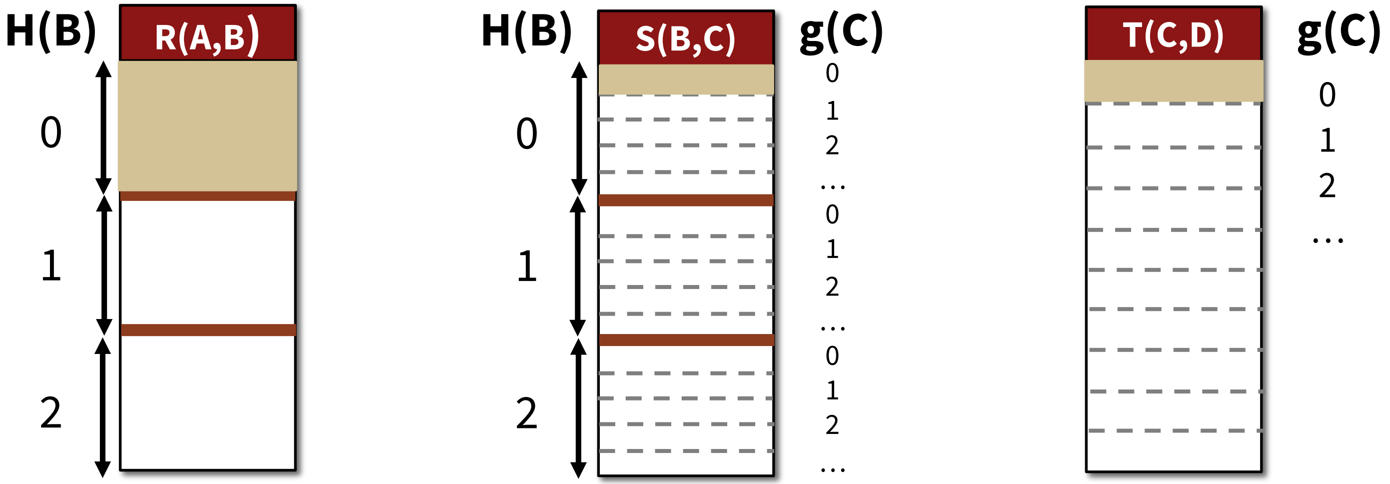
# Algorithm: Re-layout Data

- Step 1: re-layout data based on **H(B)** and **g(C)** values.
  - Pass 1: compute hashes and count # values in each partition/bucket
  - Pass 2: recompute hashes; compute off-chip addresses; re-layout records



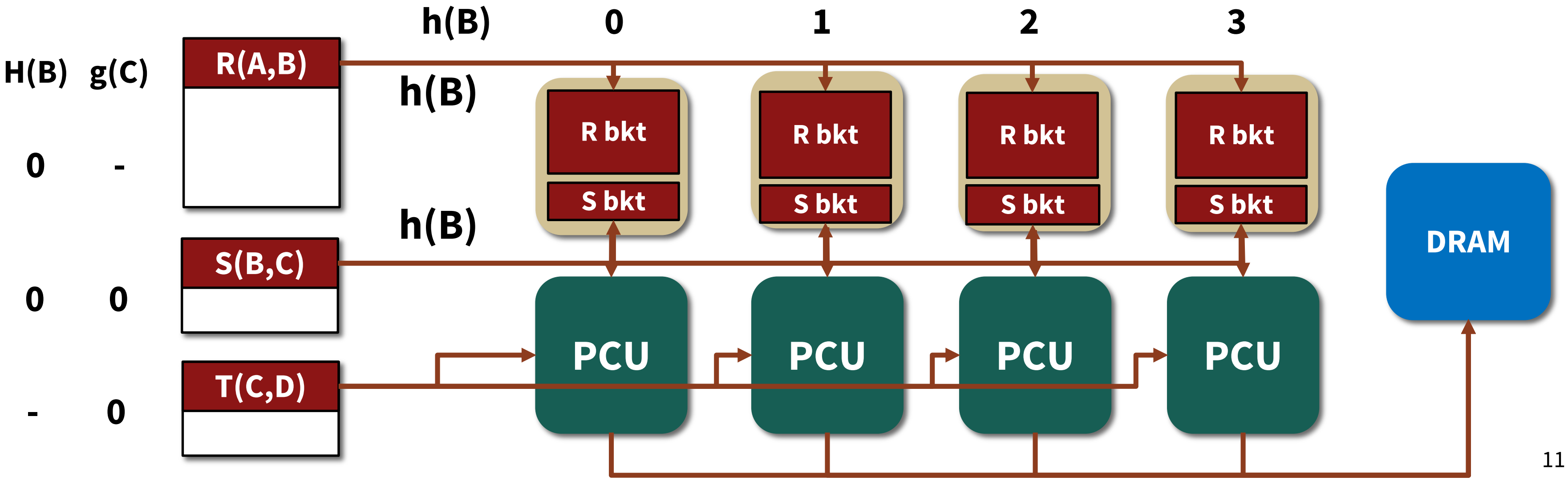
# Algorithm: Re-layout Data

- Step 2: Join partitions on-chip



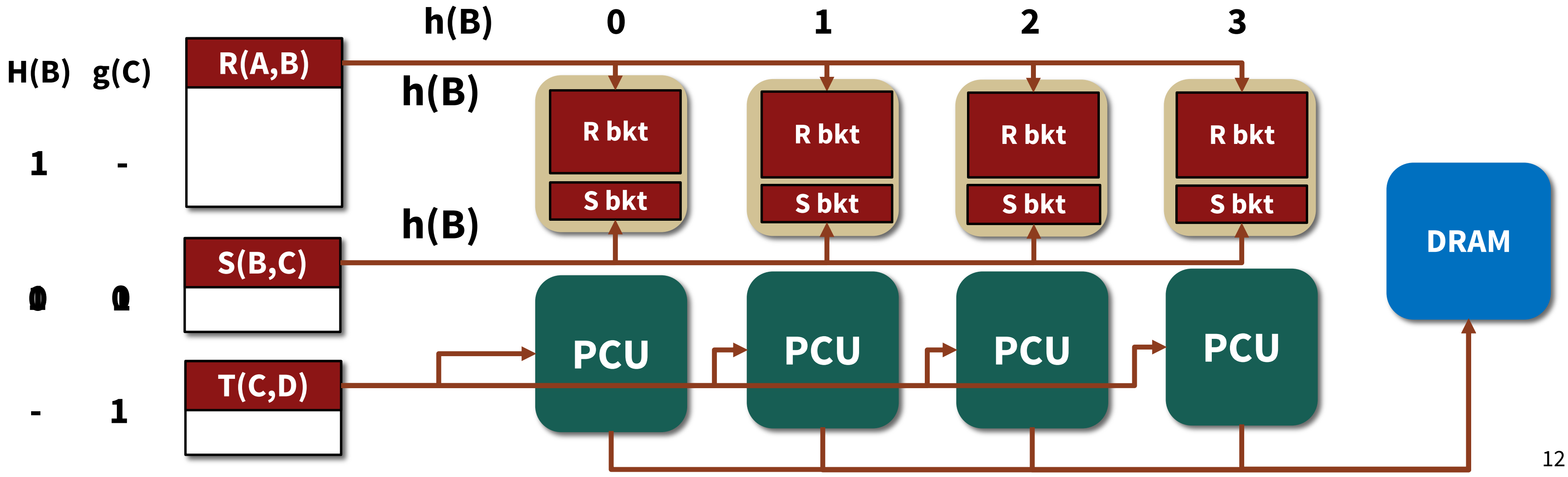
# Algorithm: 3-way Self Join

- Step 2: Join partitions on-chip
  1. Load a R partition; compute **h(B)**; send to corresponding PMUs
  2. Load a S bucket; compute **h(B)**; send to corresponding PMUs
  3. Stream in T records; broadcast to all PCUs; vectorized join between R,S,T records; stream results to DRAM.



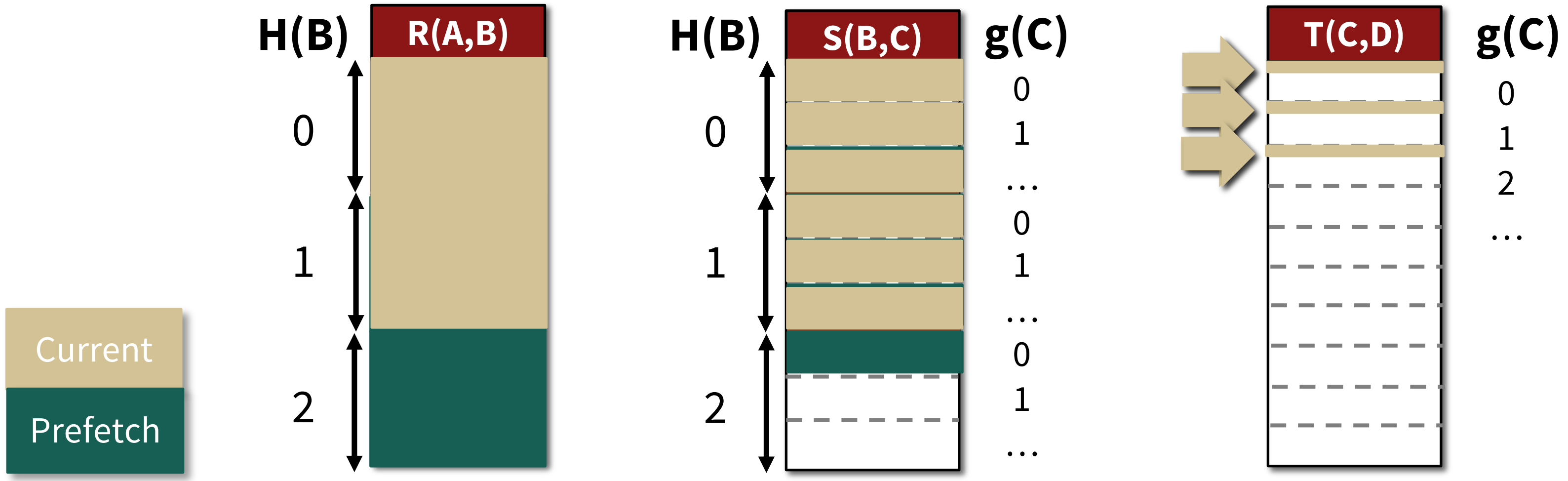
# Algorithm: 3-way Self Join

- Step 2: Join partitions on-chip
  1. Work on next S and T bucket with g(C) = 1. ***S and T grouped on g(C)!***
  2. Work on next R and S partition with H(B) = 1. ***R and S grouped on H(B)!***



# Algorithm Summary

- Prefetch next R partition while working on current partition.
- Prefetch next S bucket while working on current bucket.
- Overlap streamed T records with comparison.
- Maximize compute throughput + memory bandwidth.

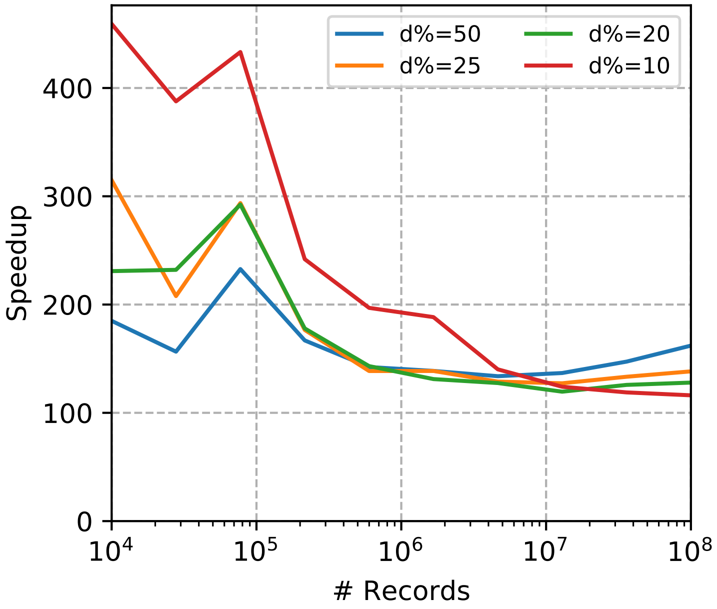


# Evaluation Environment

- System Configuration
  - Plasticine – 64 PMUs, 64 PCUs, DDR3 (49GB/s)
  - Postgres – Intel Xeon E7-8890 v3 at 2.5 GHz, multi-threaded, DDR4 (85GB/s)
- Workloads
  - Count of friends of friends of friends relations
  - TPC-H : get **lineitem** for each **part** requested in an **order**
- Data
  - Synthetic data with a uniform distribution
  - Parameterized on # distinct values in joining columns
- Evaluation
  - Cascaded binary join on CPU and Plasticine
  - Cascaded binary join vs. multiway join on Plasticine

# Cascaded Binary Join on CPU vs. Plasticine

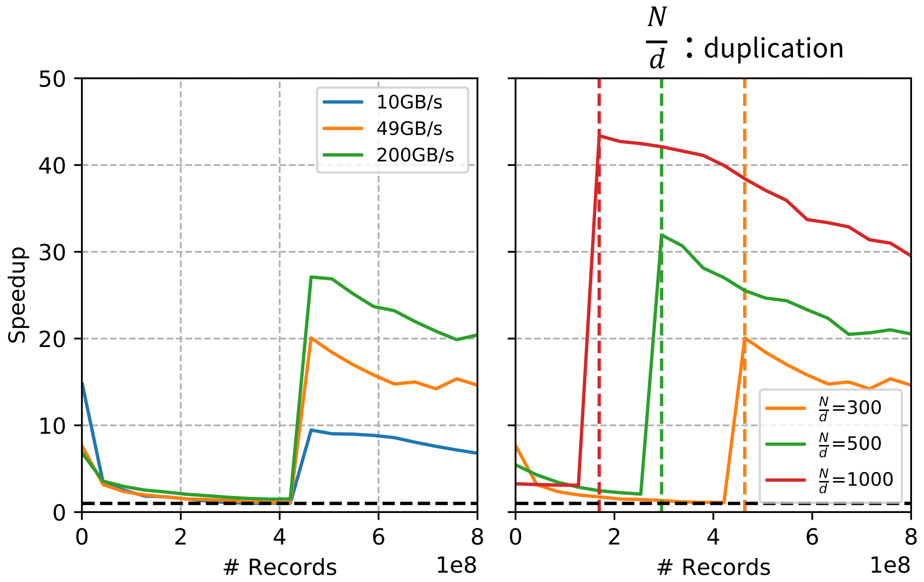
d%: % unique column values




- Postgres uses 1-5 cores for the provided dataset.
- Poor memory bandwidth usage with small data size on CPU.
- Higher speedup with more duplications/larger intermediate result.

**>100X speedup on Plasticine over Postgres**

# 3-way vs. Cascaded Binary Joins on Plasticine




- Significant speedup when intermediate result of binary joins does not fit in DRAM.
- Cascaded binary joins fit more buckets on-chip, which reduces computation complexity.

Up to 45X speedup over Cascaded binary join on Plasticine
Combined 4500X speedup over Postgres

# Conclusions and Future Work

- We have presented an algorithm for Multiway Hash Join on Plasticine-like CGRA for high performance and energy efficient

- Cascaded binary join shows at least 100x speedup over Postgres on Plasticine-like architectures
- 3-way Linear self join has up to 45X speedup over cascaded binary joins on Plastcine-like architectures.

**Future Work**

- Plan to do speedup analysis for cyclic joins
- Extend algorithm for heavy hitters – skew join
- Explore other approaches such as LeapfrogTrie join on Plasticine-like accelerator

# CPU Comparison

| N | d% | CPU (s) | Join 1 (s) | Join (s) | Accel (s) |
|---|---|---|---|---|---|
| 100000000 | 50 | 230.587056 | 0.00164063 | 0.00207031 | 1.42265756 |
| 100000000 | 25 | 251.087493 | 0.00245443 | 0.002806 | 1.81562664 |
| 100000000 | 20 | 262.652579 | 0.00294271 | 0.00327605 | 2.05312686 |
| 100000000 | 10 | 386.222669 | 0.00555209 | 0.00585681 | 3.32453486 |
| 35938136 | 50 | 75.3082163 | 0.00058961 | 0.00074403 | 0.51127708 |
| 35938136 | 25 | 86.994952 | 0.00088208 | 0.00100842 | 0.65250305 |
| 35938136 | 20 | 92.8356616 | 0.00105755 | 0.00117922 | 0.73796806 |
| 35938136 | 10 | 142.018154 | 0.00199531 | 0.00210484 | 1.19477673 |
| 12915496 | 50 | 25.1308157 | 0.0002119 | 0.00026739 | 0.18374426 |
| 12915496 | 25 | 29.8497026 | 0.000317 | 0.00036241 | 0.23449845 |
| 12915496 | 20 | 31.7024844 | 0.00038006 | 0.00042379 | 0.26521217 |
| 12915496 | 10 | 53.2836006 | 0.00071707 | 0.00075645 | 0.42938129 |
| 4641588 | 50 | 8.83704805 | 7.62E-05 | 9.61E-05 | 0.06603494 |
| 4641588 | 25 | 10.8591459 | 0.00011393 | 0.00013025 | 0.08427525 |
| 4641588 | 20 | 12.1534784 | 0.00013658 | 0.0001523 | 0.09531261 |
| 4641588 | 10 | 21.6276131 | 0.00025769 | 0.00027187 | 0.15431259 |
| 1668100 | 50 | 3.29217792 | 2.74E-05 | 3.45E-05 | 0.02373263 |
| 1668100 | 25 | 4.19987917 | 4.09E-05 | 4.68E-05 | 0.03028806 |
| 1668100 | 20 | 4.49304557 | 4.91E-05 | 5.47E-05 | 0.03425408 |
| 1668100 | 10 | 10.4484825 | 9.26E-05 | 9.77E-05 | 0.05546084 |

# CPU Comparison

| N | d% | CPU (s) | Join 1 (s) | Join (s) | Accel (s) |
|---|---|---|---|---|---|
| 599484 | 50 | 1.21376657 | 9.84E-06 | 1.24E-05 | 0.00852966 |
| 599484 | 25 | 1.50879955 | 1.47E-05 | 1.68E-05 | 0.01088524 |
| 599484 | 20 | 1.76229048 | 1.76E-05 | 1.97E-05 | 0.01231137 |
| 599484 | 10 | 3.92580438 | 3.33E-05 | 3.52E-05 | 0.01993426 |
| 215443 | 50 | 0.51171207 | 3.54E-06 | 4.46E-06 | 0.00306625 |
| 215443 | 25 | 0.6908071 | 5.29E-06 | 6.05E-06 | 0.00391269 |
| 215443 | 20 | 0.78749514 | 6.34E-06 | 7.08E-06 | 0.00442551 |
| 215443 | 10 | 1.73348927 | 1.20E-05 | 1.27E-05 | 0.00716592 |
| 77426 | 50 | 0.25660062 | 1.27E-06 | 1.60E-06 | 0.00110234 |
| 77426 | 25 | 0.4132576 | 1.90E-06 | 2.18E-06 | 0.00140695 |
| 77426 | 20 | 0.46525812 | 2.28E-06 | 2.55E-06 | 0.00159135 |
| 77426 | 10 | 1.11664009 | 4.31E-06 | 4.57E-06 | 0.00257759 |
| 27825 | 50 | 0.06212115 | 4.57E-07 | 5.78E-07 | 0.00039697 |
| 27825 | 25 | 0.10527158 | 6.85E-07 | 7.86E-07 | 0.00050655 |
| 27825 | 20 | 0.13294554 | 8.21E-07 | 9.22E-07 | 0.00057289 |
| 27825 | 10 | 0.36002111 | 1.55E-06 | 1.67E-06 | 0.00092869 |
| 10000 | 50 | 0.02654147 | 1.65E-07 | 2.09E-07 | 0.00014351 |
| 10000 | 25 | 0.05765629 | 2.48E-07 | 2.87E-07 | 0.00018313 |
| 10000 | 20 | 0.04781318 | 2.97E-07 | 3.38E-07 | 0.0002071 |
| 10000 | 10 | 0.15425682 | 5.60E-07 | 6.24E-07 | 0.00033594 |

# Binary and 3-way Join Runtime on Plasticine