A Spinning Join That Does Not Get Dizzy

Philip Frey, Romulo Gonçalves, Martin Kersten, **Jens Teubner**

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CWI

Distributed Databases Folklore

Assumptions:

- Workload is **known in advance**.
- Network is **slow**.

Prevalent Architecture:

- **Allocate data to nodes** (based on workload).
- **Ship queries and state**, minimize traffic.
  - *E.g.*, ship partial results or filtered data.

30 Years Later

Reality Today:

- Complex **ad-hoc workloads** (BI, eScience).
- Networks are **fast** ($\geq$ 10 Gb/s).

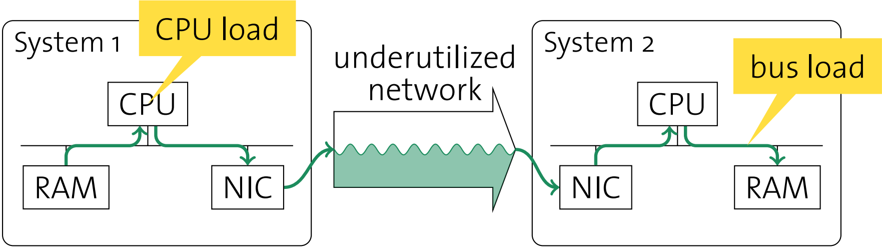
Thus: Re-Think Architecture

- Don't be afraid to **move data**.
- Leverage available **network speed**.

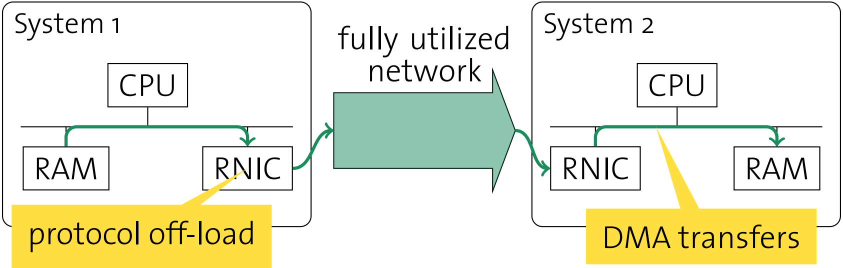
This Talk:

1. *Data Cyclotron* architecture (**transport layer**).
2. **Join execution** (*cyclo-join*).

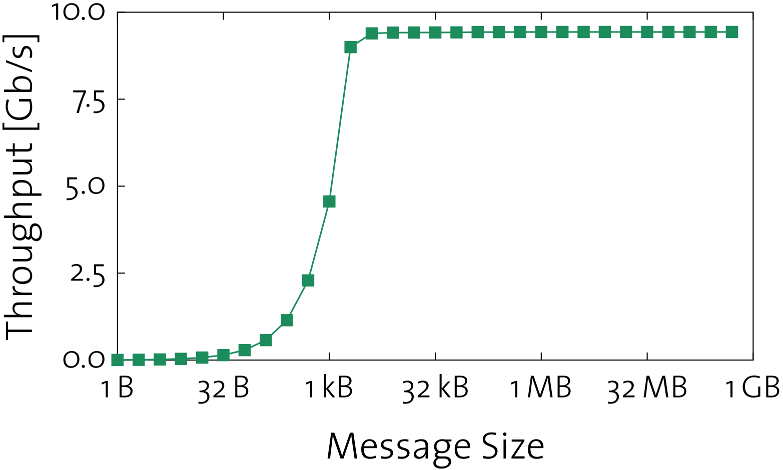
High-Speed Networks?



Thus: **Remote Direct Memory Access (RDMA)**

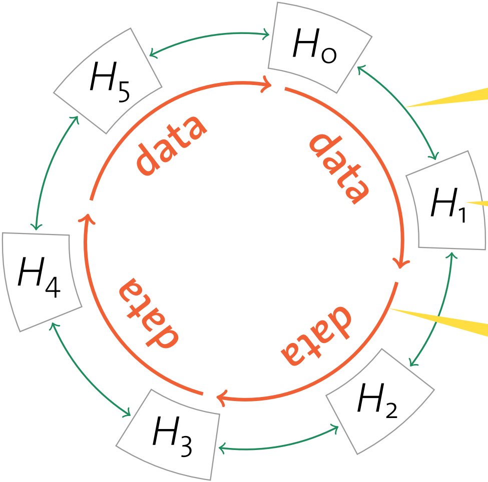


RDMA Throughput



→ Design algorithms **carefully**.

Data Cyclotron Architecture



RDMA
(10 Gb/s Ethernet)

in-memory processing

data spinning in ring

Join Problem

Task: Find tuples $r \in R$ and $s \in S$ such that predicate $p(r, s)$ holds.

Naïve implementation: Nested-Loops Join

```
foreach $r \in R$ do
    foreach $s \in S$ do
        if $p(r, s)$ then
            append $\langle r, s \rangle$ to result;
```

Better: Hash Join or **Sort-Merge Join**

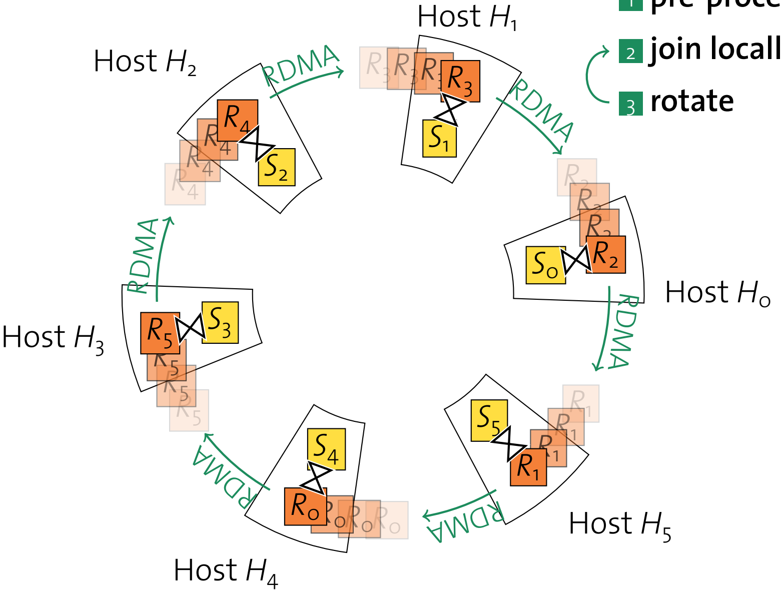
1. **Pre-process data** (create hash table / sort)
2. **Compute join** (scan and probe / merge sorted tables)

Trade-off: Pre-processing cost vs. join cost.

1 **pre-process**

2 **join locally**

3 **rotate**



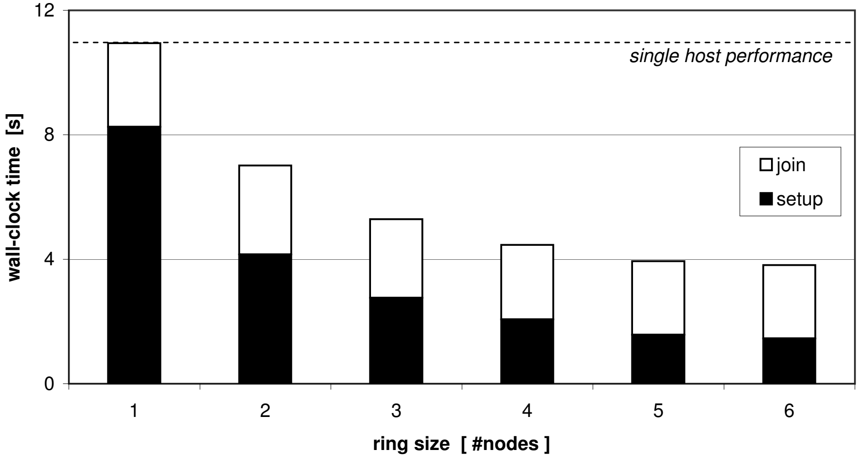
RDMA: join **and** rotate

Cyclo-Join Properties

- Distributed input → distributed output.
  → Multi-step joins.
- Pair with **any** local join algorithm.
- **Scale** with **distributed memory**.
  → Large joins for analytics and business intelligence.

Will it blend?

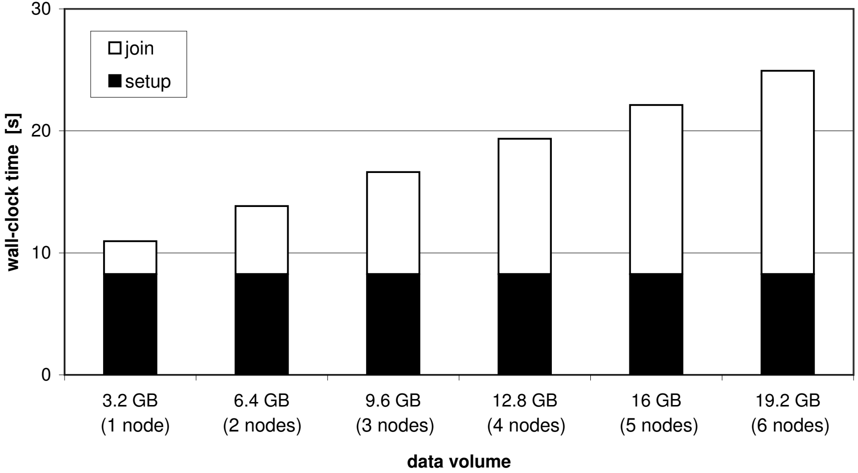
Evaluate 1.6 GB $\bowtie$ 1.6 GB[1] using 1, . . . , 6 hosts:



→ Distribute pre-processing work.

[1] hash join; 140 million rows per table; 12 bytes per tuple.

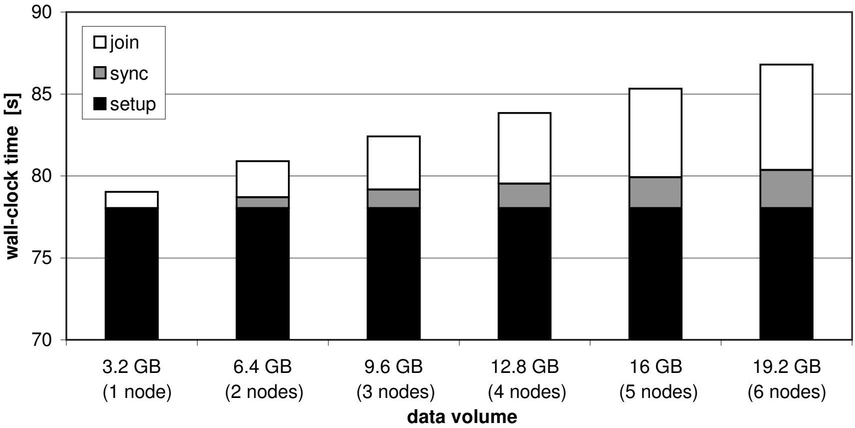
Scale up: 1.6 GB ⋈ 1.6 GB → 9.6 GB ⋈ 9.6 GB:



→ Leverage **distributed memory**

(could not have done this join on a single host).

Sort-Merge Join: Maximum Bandwidth Need



We now see a **communication overhead**.

→ **RDMA** avoids most of it, though.

Take-Home Message

High-speed networks ↯ classical assumptions.

Data Cyclotron:

- Pump data **continuously** through a **ring**.
- Design for **hardware acceleration** (**RDMA**).

Join Processing in *Data Cyclotron* (*cyclo-join*):

- **Rotate one relation** once, **distributed join**.
- Leverage **distributed memory**, **scale with data sizes**.

This work was supported by a Swiss National Science Foundation (SNSF) *Ambizione* grant.