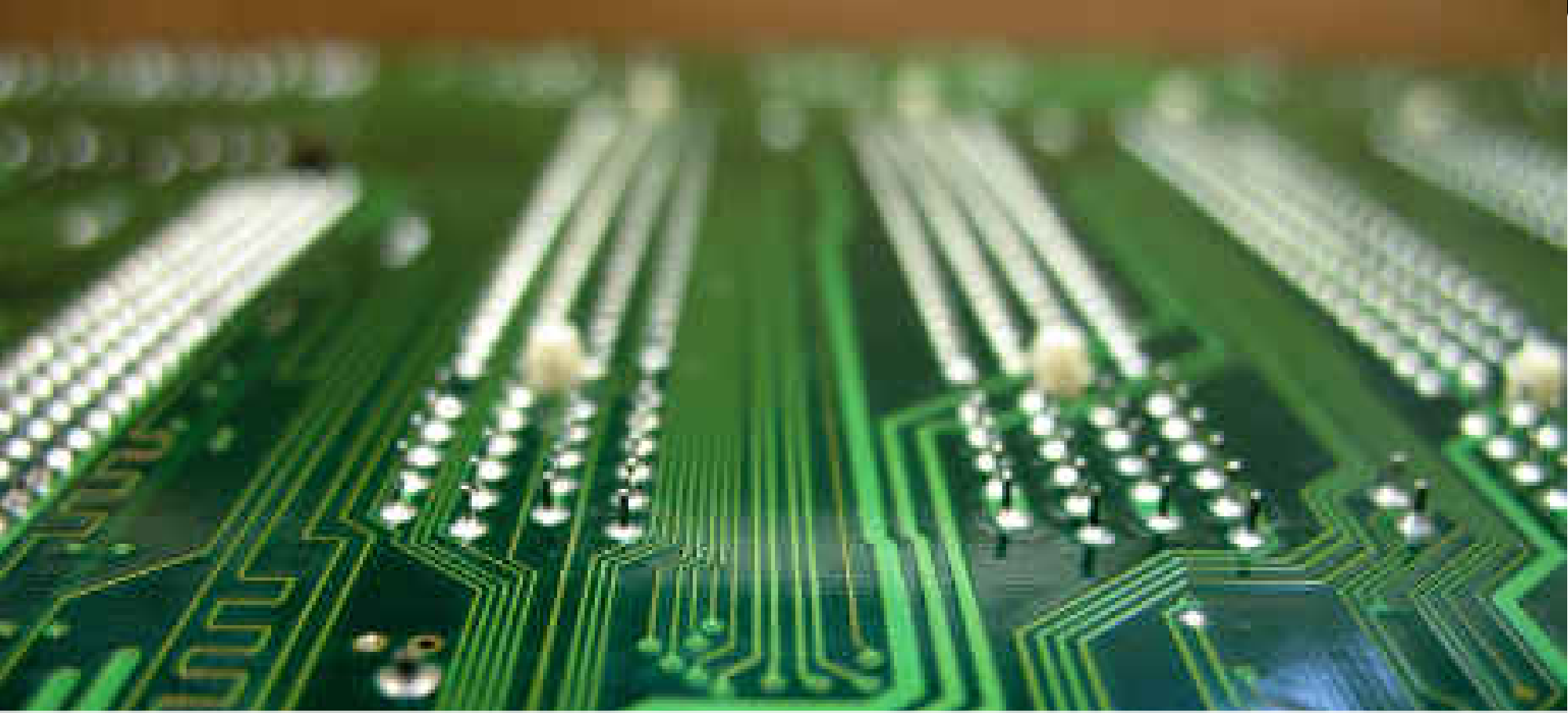
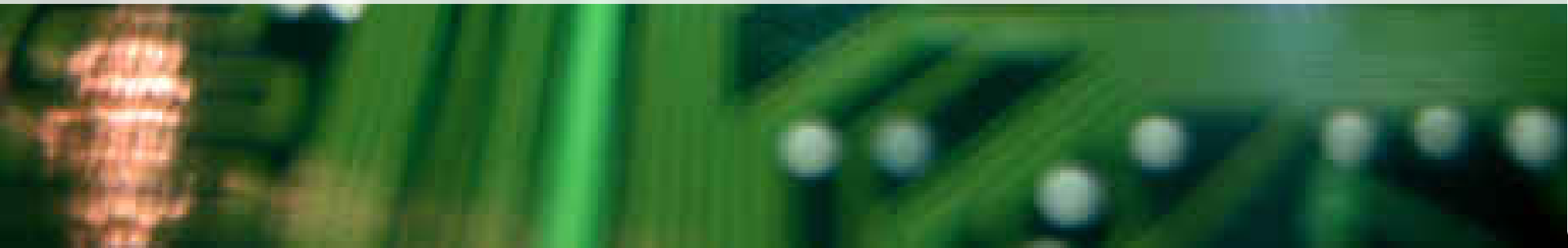
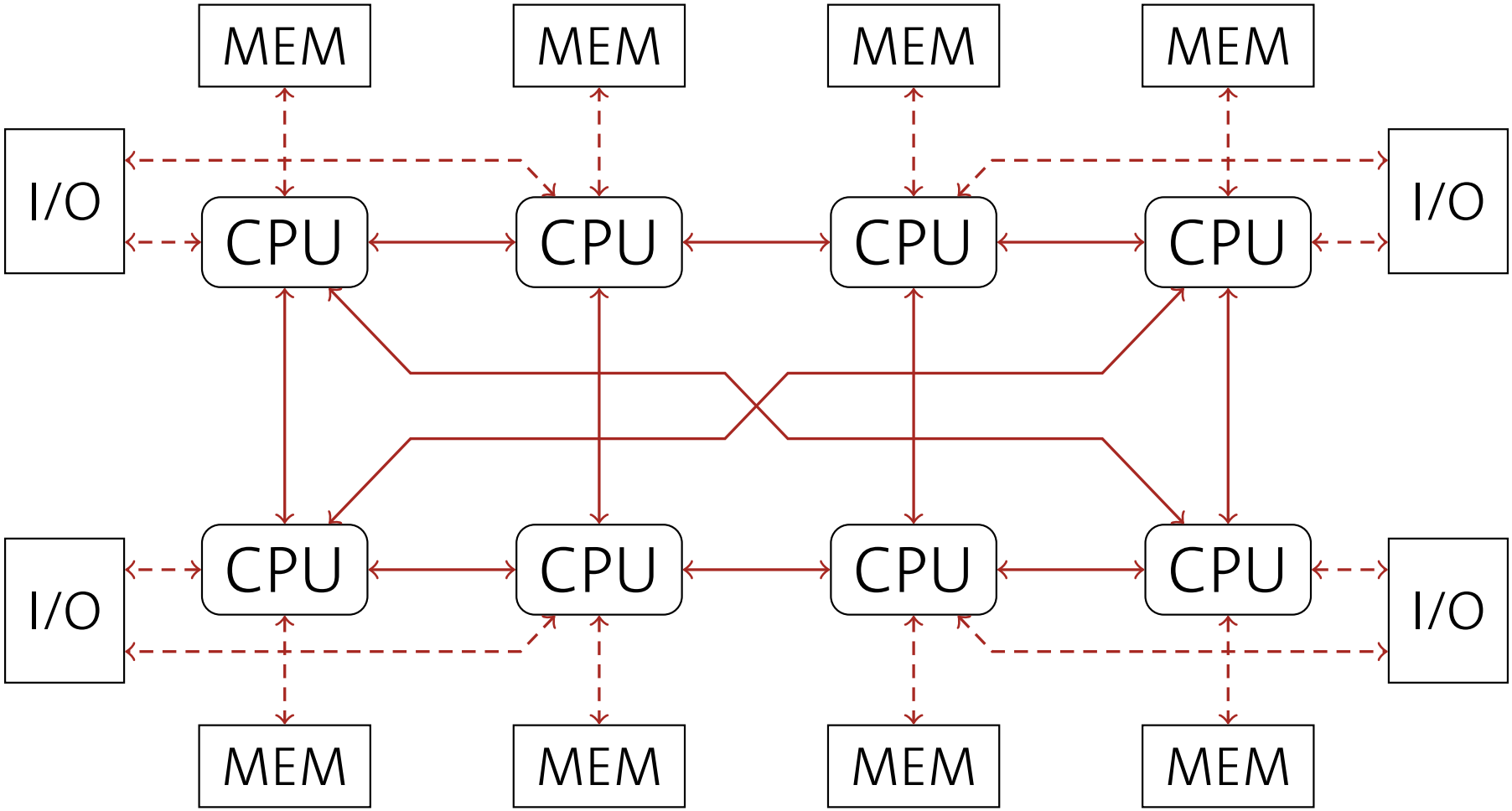


Database Algorithms $\leftrightarrow$ Networks

Jens Teubner · ETH Zurich, Systems Group



Modern computers are **networks**!



(8-way Intel Nehalem-EX)

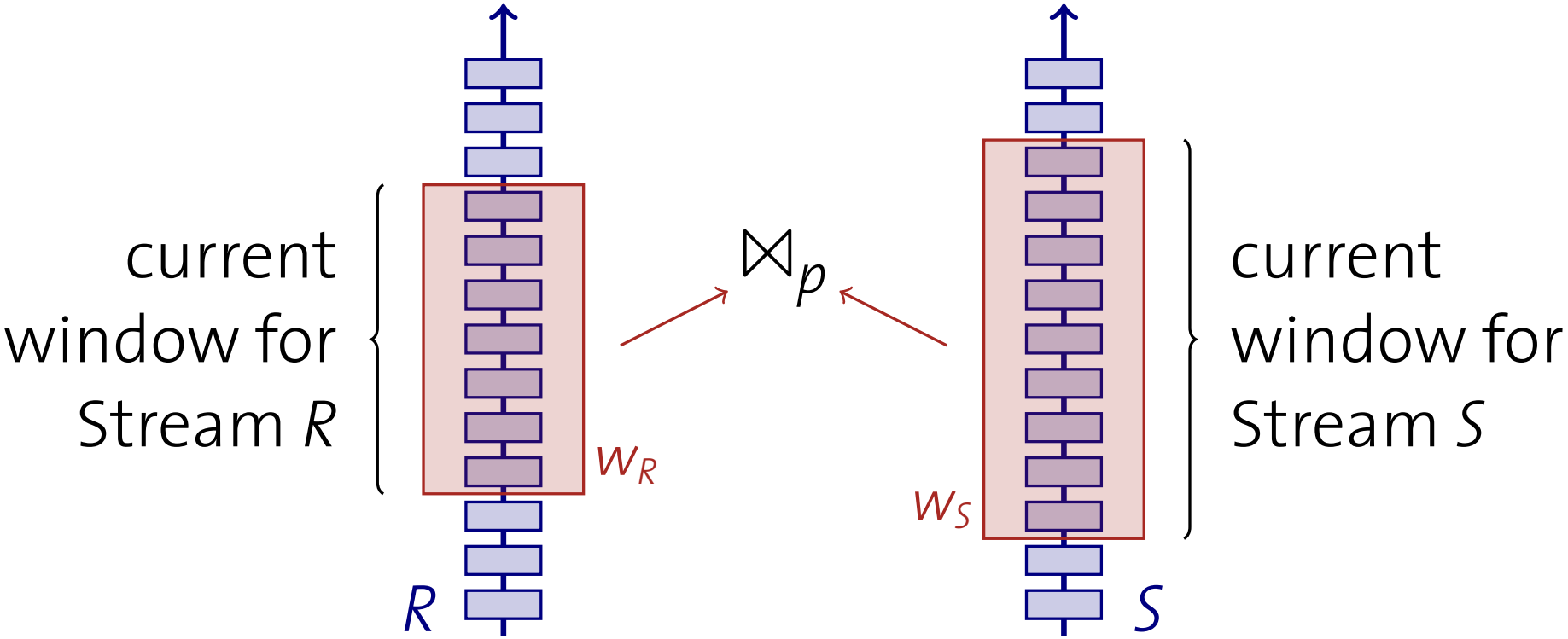
Networks are **all over the place**:

- **data center** networks
- **system-level** networks (QPI, PCIe, …)
- **on-chip** interconnect (*e.g.*, in FPGAs)

We need **topology-aware algorithms**.

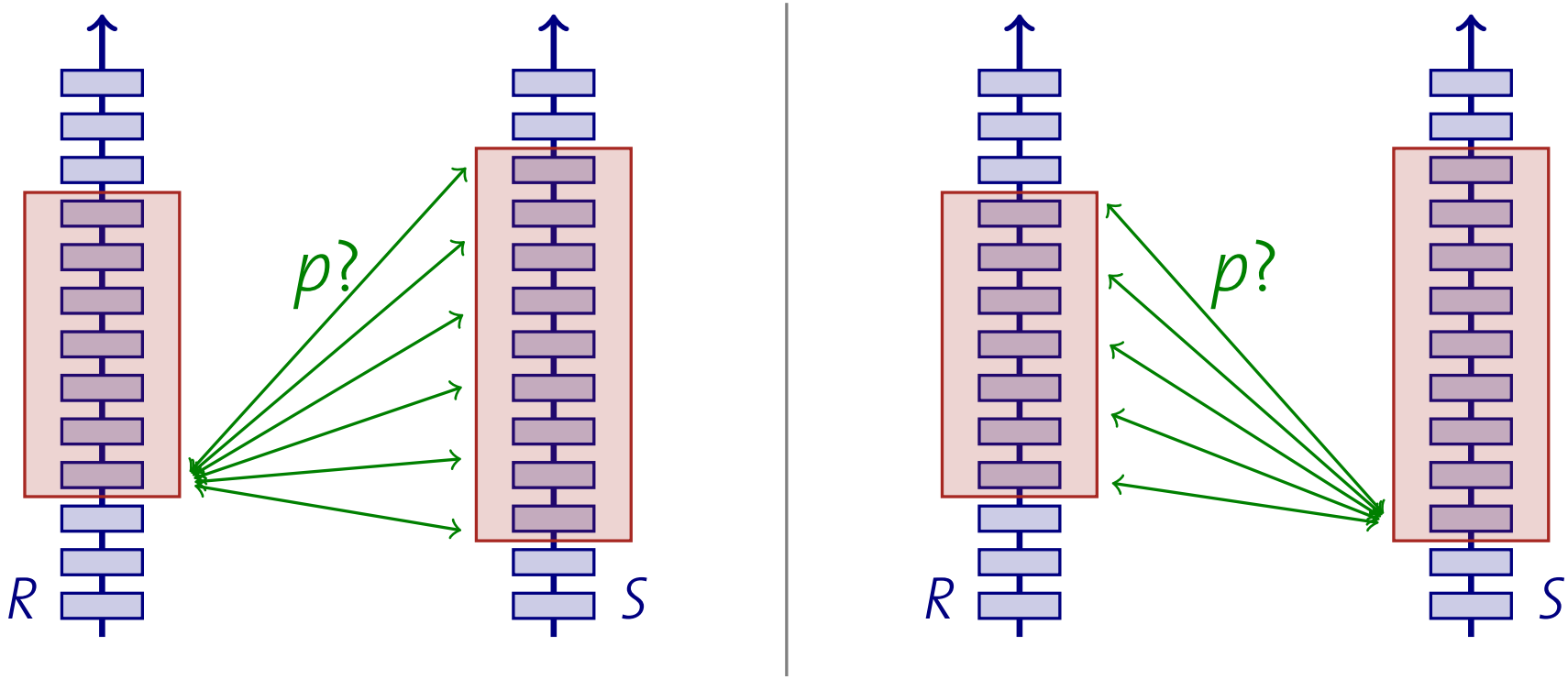
1. communication pattern
2. distance
3. synchronization

Joins Over Data Streams:



Task: Find all $\langle r, s \rangle$ in w_R, w_S that satisfy $p(r, s)$.

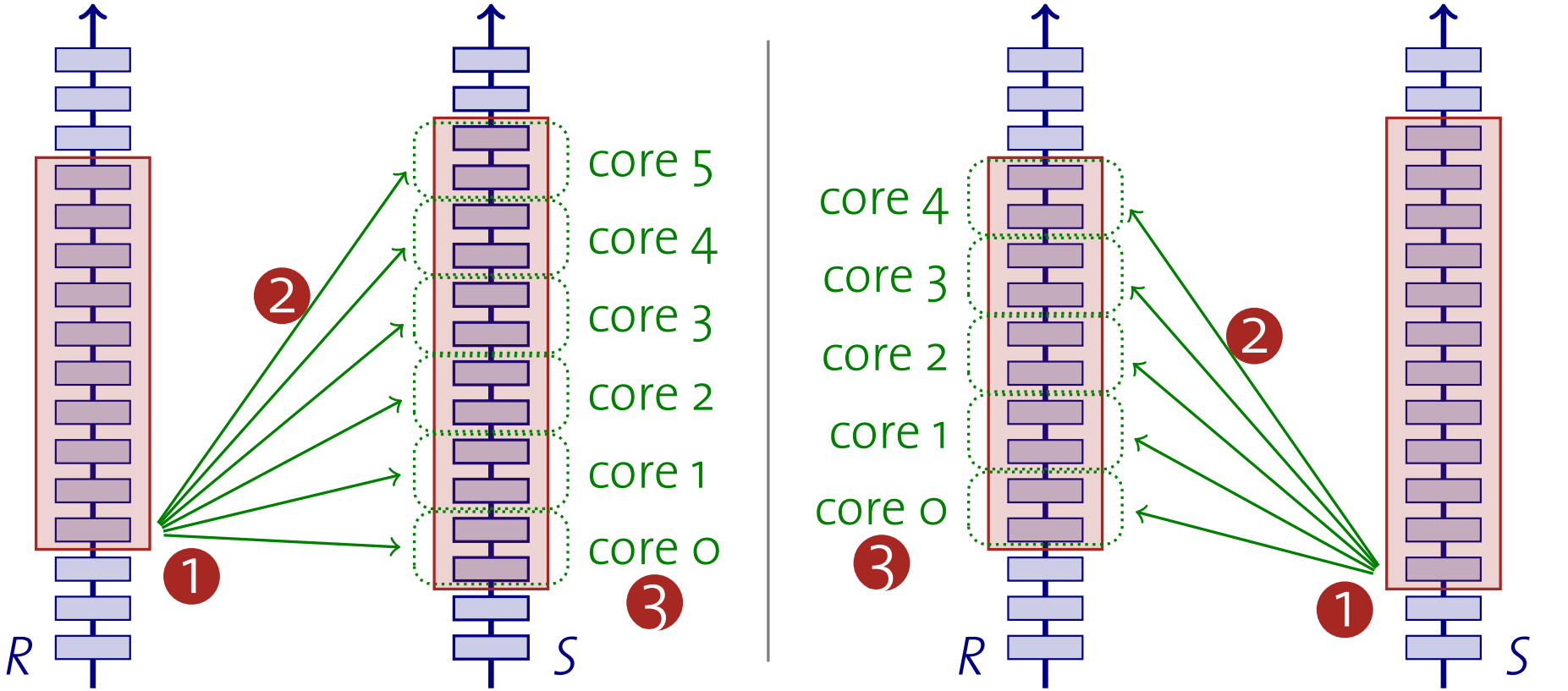
Implementation: [Kang *et al.*, ICDE 2003]



1. scan window, **2. insert** new tuple, **3. invalidate** old

Parallel Execution?

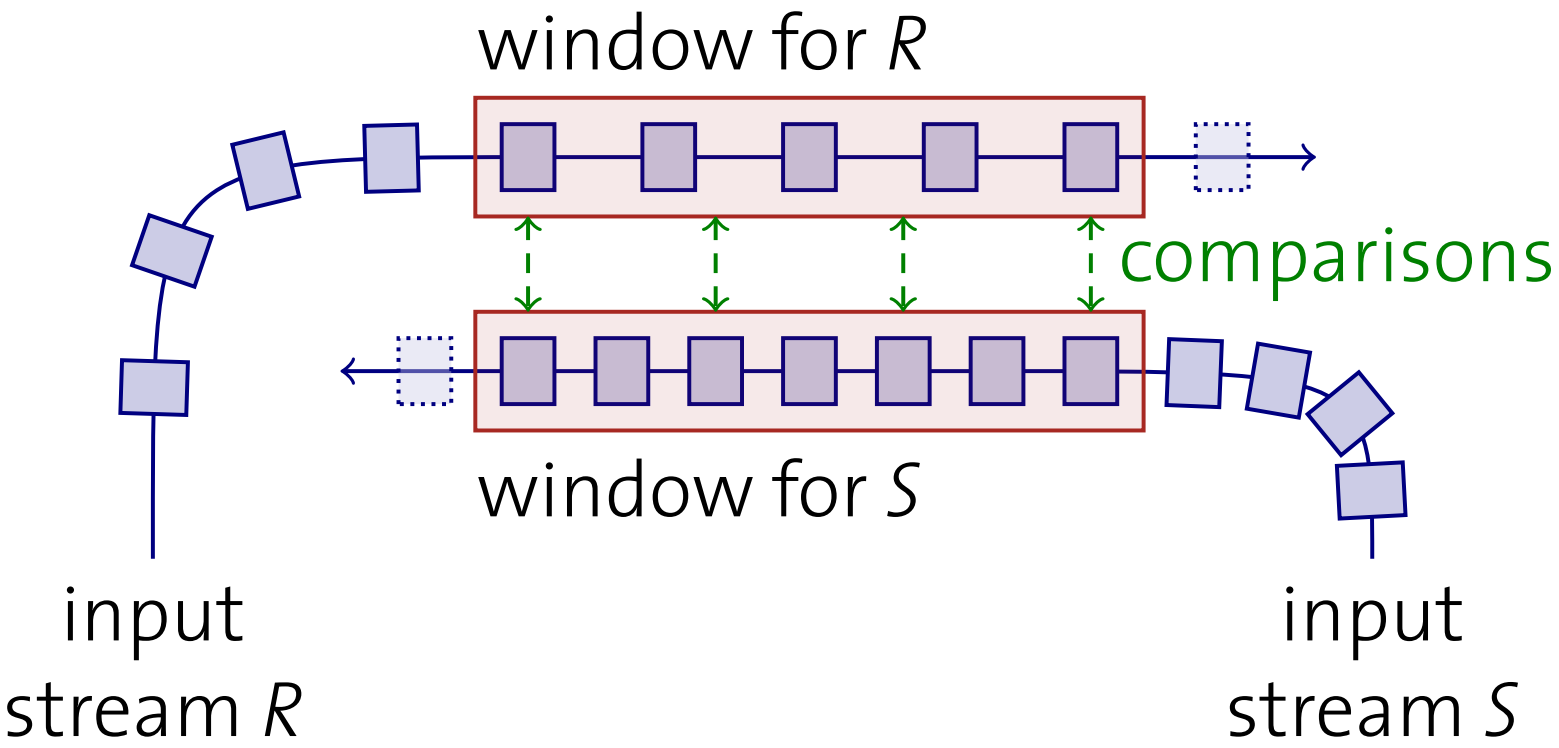
CellJoin: [Gedik *et al.*, VLDBJ 2009]



1. **bandwidth** bottlenecks
2. **long-distance** communication
3. **centralized** coordination and memory

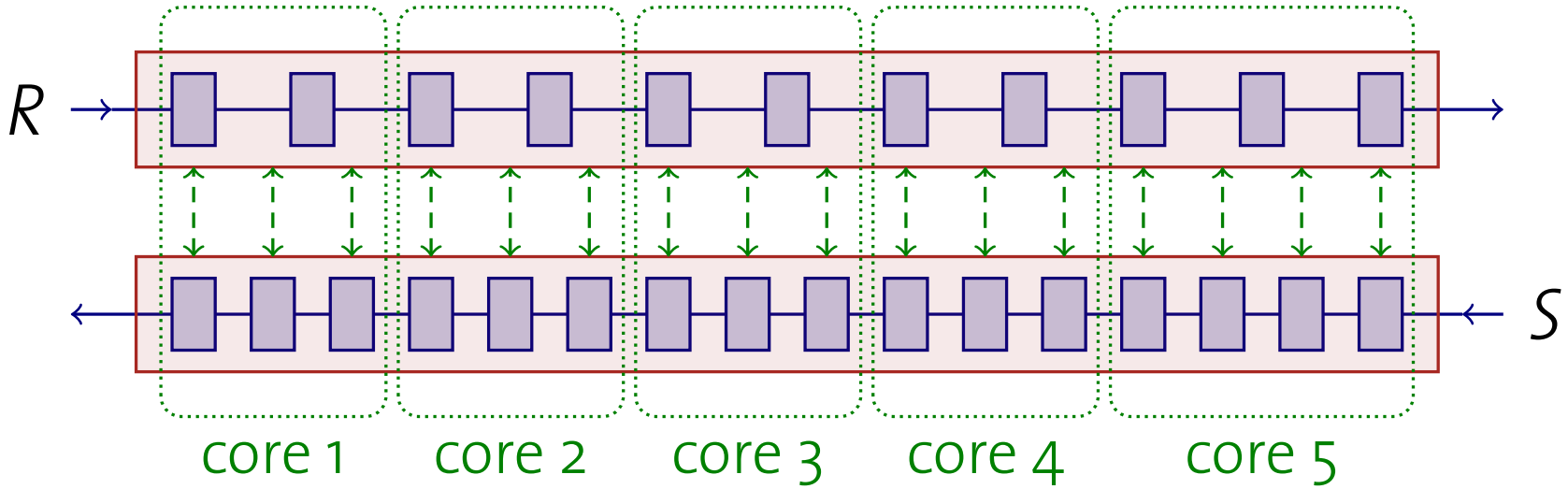
Handshake Join

Handshake Join:



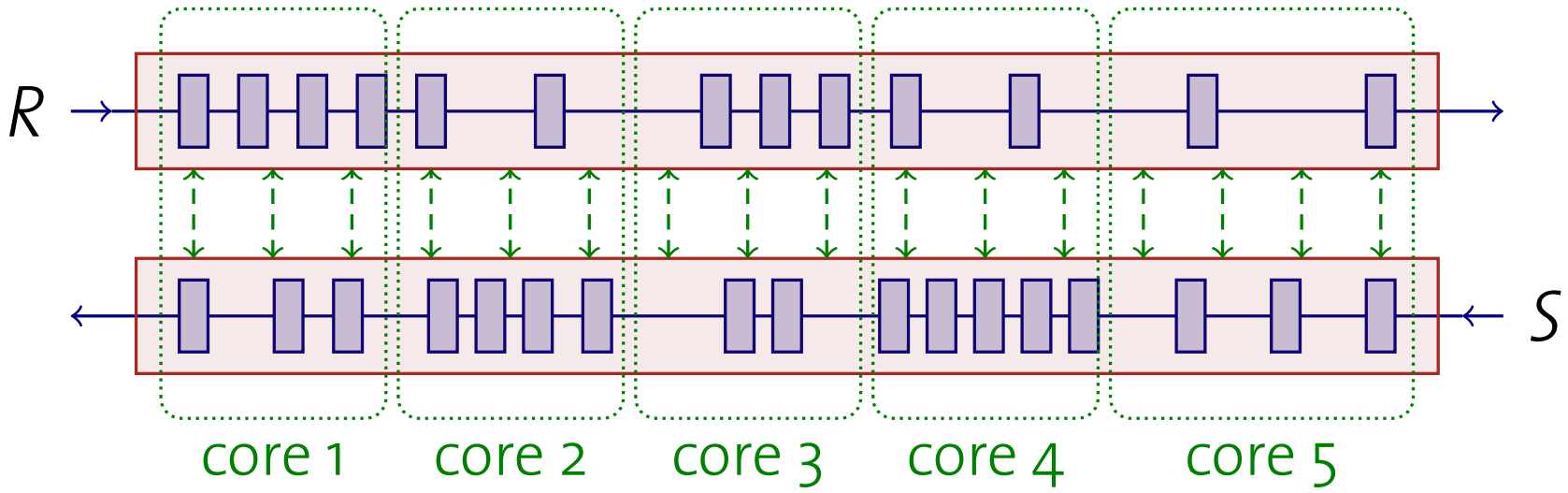
Streams flow by in **opposite directions**
Compare tuples when they **meet**

Data flow representation → **parallelization**:



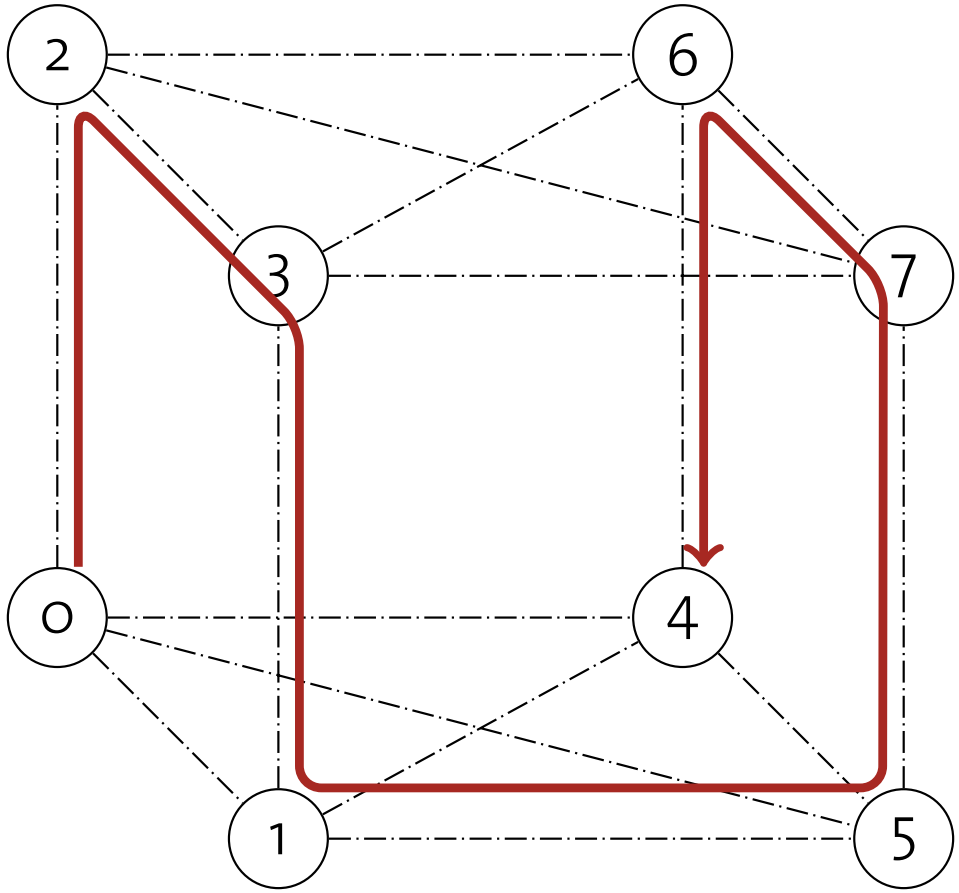
- **No bandwidth bottleneck** ❶ ✔
- Communication/synchronization stays **local** ❷ ✔

Coordination can now be done **autonomously**

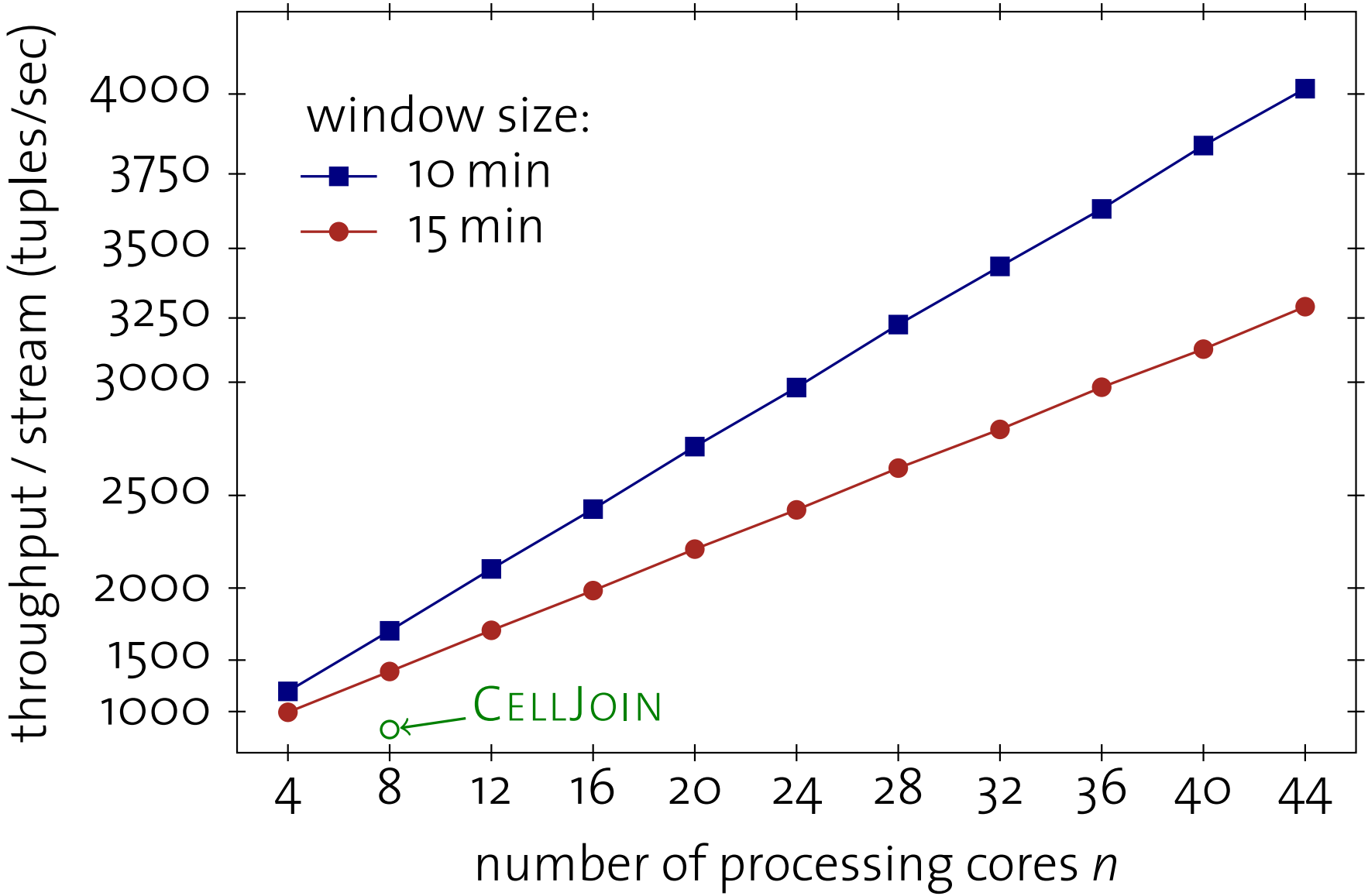


- no more centralized coordination ❸ ✔
- Autonomous **load balancing**
- **Lock-free message queues** between neighbors

Example: AMD “Magny Cours” (48 cores)



Experiments (AMD Magny Cours, 2.2 GHz)



Results validated by the SIGMOD 2011 repeatability committee.

Topology-awareness:

- communic. pattern
- distance
- synchronization

***E.g.*,**

- look at **data flow**
- **asynchronous** oper.

`http://www.systems.ethz.ch/projects/avalanche/`