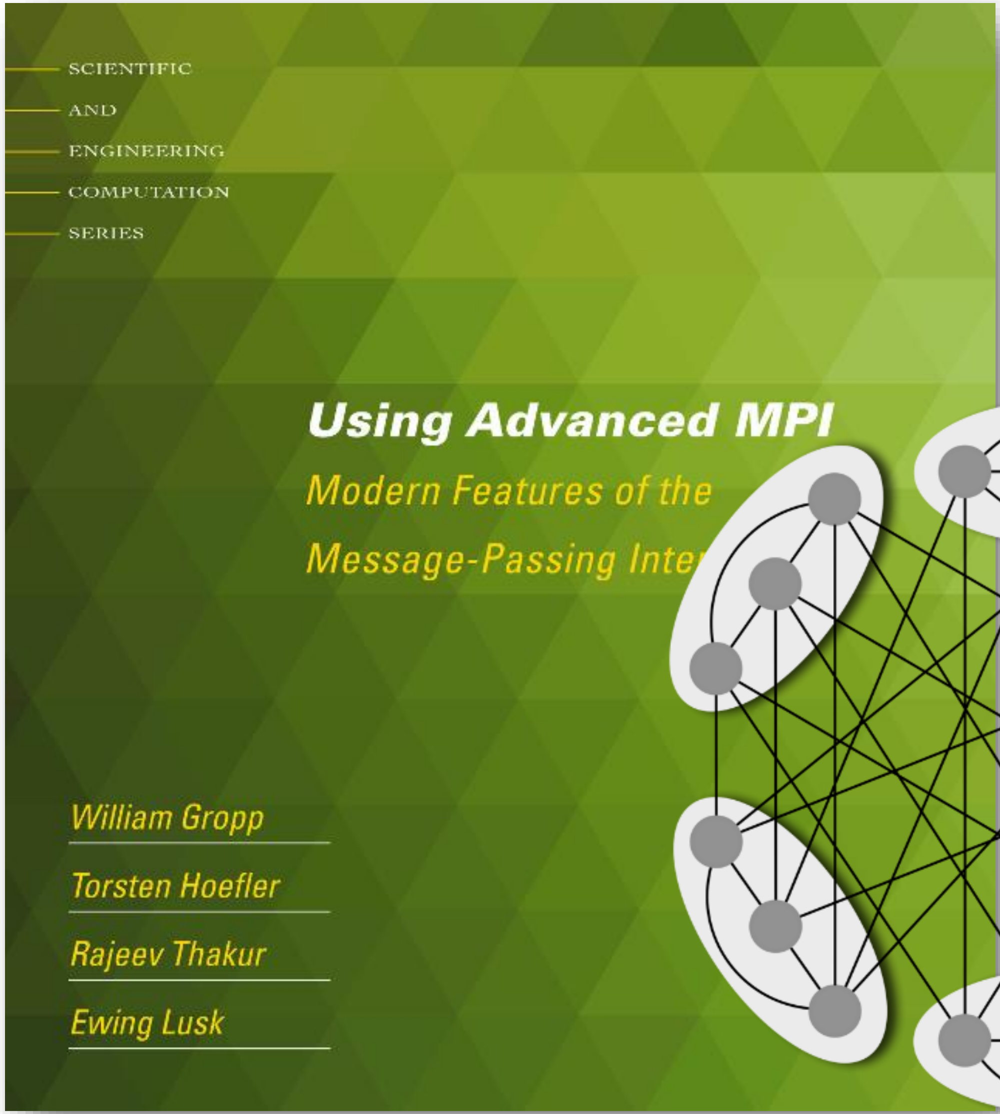
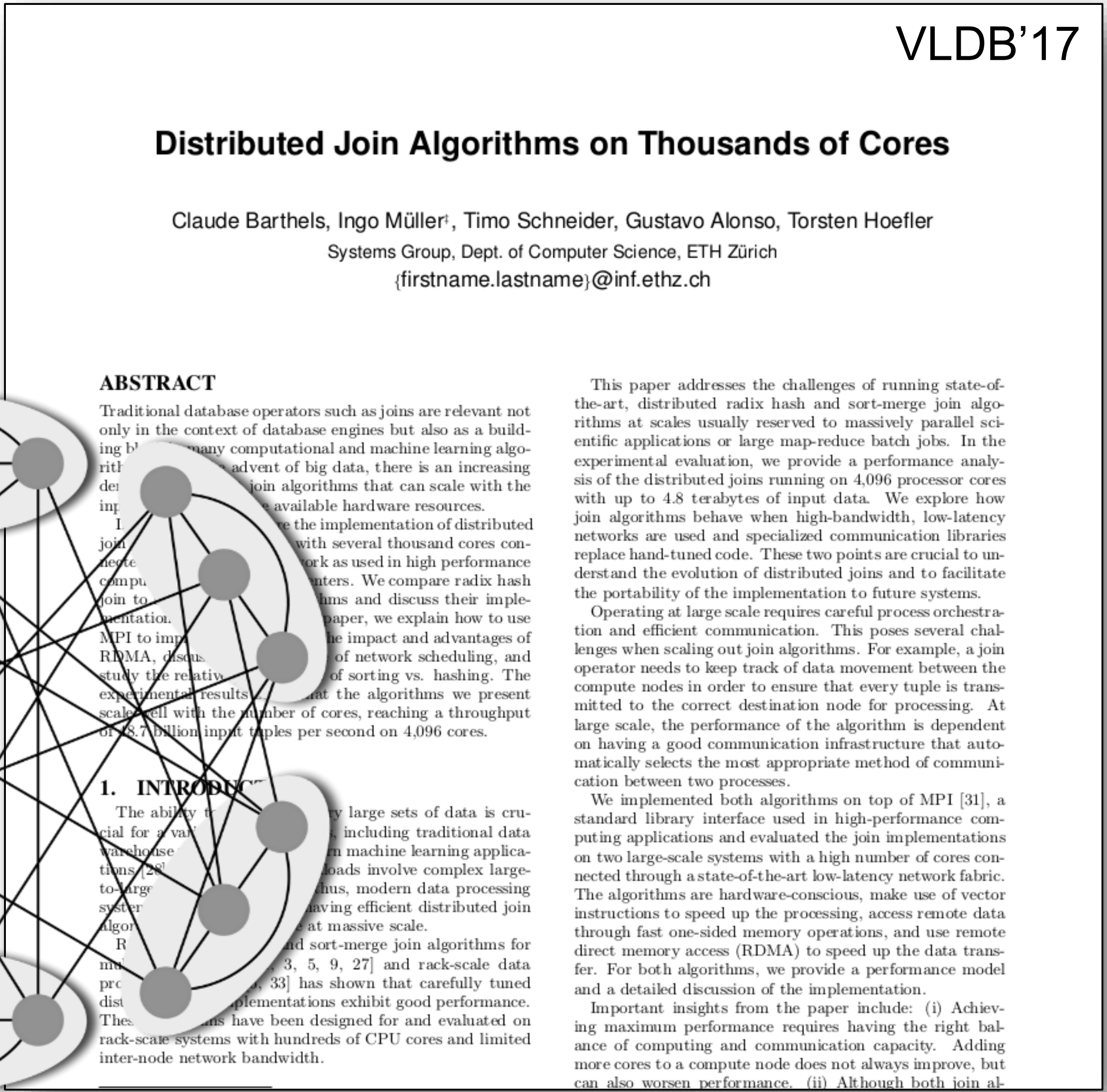
HiPINEB Panel – MSN vs. ICN (DC vs. HPC networks)

Torsten Hoefler (on behalf of nobody, not even my institution or myself!)

Who am I?




VLDB'17

Distributed Join Algorithms on Thousands of Cores

Claude Barthels, Ingo Müller, Timo Schneider, Gustavo Alonso, Torsten Hoefler
Systems Group, Dept. of Computer Science, ETH Zürich
{firstname.lastname}@inf.ethz.ch

ABSTRACT

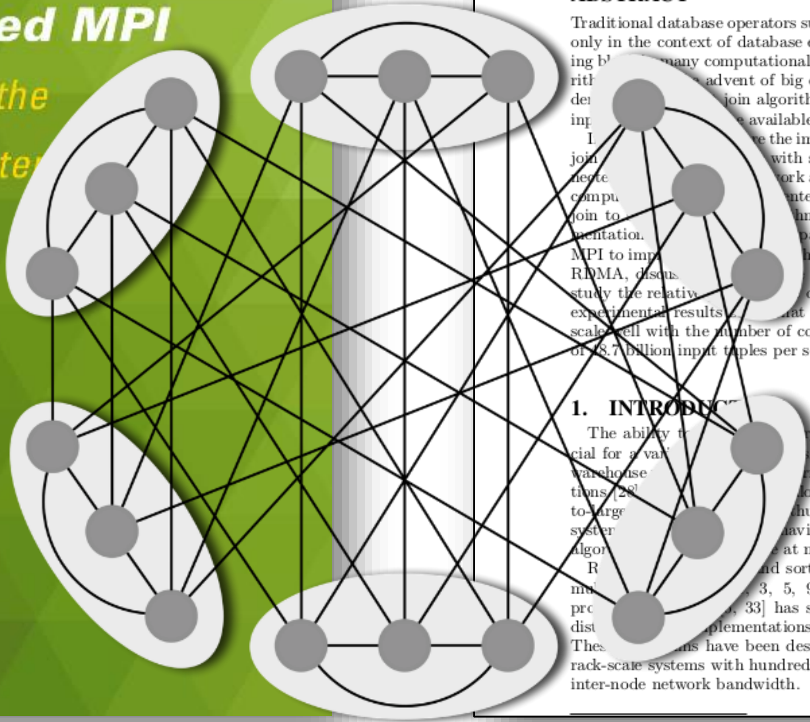
Traditional database operators such as joins are relevant not only in the context of database engines but also as a build-

This paper addresses the challenges of running state-of-the-art, distributed radix hash and sort-merge join algorithms at scales usually reserved to massively parallel scientific applications or large map-reduce batch jobs. In the experimental evaluation, we provide a performance analysis of the distributed joins running on 4,096 processor cores with up to 4.8 terabytes of input data. We explore how join algorithms behave when high-bandwidth, low-latency networks are used and specialized communication libraries replace hand-tuned code. These two points are crucial to understand the evolution of distributed joins and to facilitate the portability of the implementation to future systems.

Operating at large scale requires careful process orchestration and efficient communication. This poses several challenges when scaling out join algorithms. For example, a join operator needs to keep track of data movement between the compute nodes in order to ensure that every tuple is transmitted to the correct destination node for processing. At large scale, the performance of the algorithm is dependent on having a good communication infrastructure that automatically selects the most appropriate method of communication between two processes.

We implemented both algorithms on top of MPI [31], a standard library interface used in high-performance computing applications and evaluated the join implementations on two large-scale systems with a high number of cores connected through a state-of-the-art low-latency network fabric. The algorithms are hardware-conscious, make use of vector instructions to speed up the processing, access remote data through fast one-sided memory operations, and use remote direct memory access (RDMA) to speed up the data transfer. For both algorithms, we provide a performance model and a detailed discussion of the implementation.

Important insights from the paper include: (i) Achieving maximum performance requires having the right balance of computing and communication capacity. Adding more cores to a compute node does not always improve, but can also worsen performance. (ii) Although both join al-



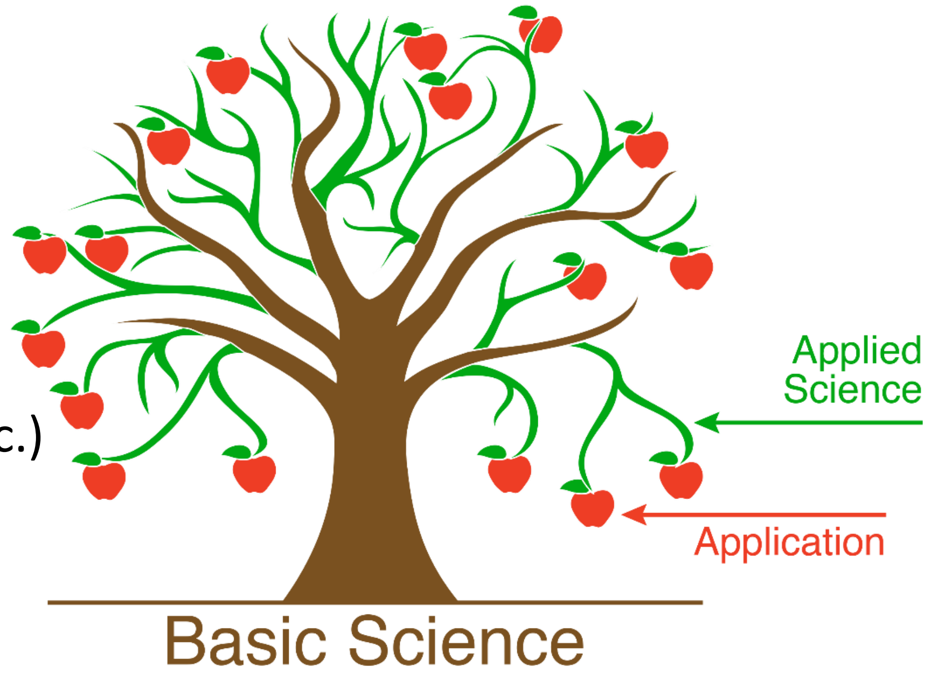
Define how MSN and ICN are similar and different.

- **Parameters:**
  - Bandwidth, bandwidth, latency
- **Machine characteristic**
  - (Loose) collection of racks
  - Incremental upgrade
  - Highly available during upgrade
- **Network characteristics**
  - Multi-vendor
  - Heterogeneous (ToR vs. Spine vs. external)
  - Ethernet
  - Lossy!
- **Protocols**
  - TCP/IP

- **Parameters:**
  - Latency, bandwidth, message rate
- **Machine characteristics**
  - Single machine (supercomputer)
  - Rolled in as an atomic unit, upgrades limited
  - Shut down for extended periods
- **Network characteristics**
  - Single-vendor
  - Homogeneous
  - Ethernot
  - Lossless!
- **Protocols**
  - Specialized, highly optimized, proprietary

Should research funding agencies support research in MSN or ICN?

- **Let me be overly controversial/extreme ☺**
  - DC research: given a set of rather complex somewhat random constraints (Ethernet, ECMP, MPLS, TRILL, OpenFlow, RoCE, RoCEv2, VXLAN, NVGRE, SPB, …), figure out how to improve parallel and distributed computing workloads
  - HPC: clean-slate approach, design network and protocols from scratch to fit requirements of parallel computing workloads (pioneers adaptive routing in hardware etc.)
- **Which one would you rather fund?**
  - Fundamental research or system cobbling? ☺
- **Where will your funding have highest societal impact?**
  - Facebook, Google, etc. (with their own xx billion budgets) vs.
  - Basic sciences (climate (!), drugs, cancer, physics, astronomy etc.)
  - … not immediately clear?



Bandwidth: Do we need more bandwidth for either MSN or ICN?

- Yes
- Why? → Latency hiding!

```
for (int i = 0; i < steps; ++i) {
  for (int idx = from; idx < to; idx += jstride)
    out[idx] = -4.0 * in[idx] +
      in[idx + 1] + in[idx - 1] +
      in[idx + jstride] + in[idx - jstride];

  if (lsend)
    dcuda_put_notify(ctx, wout, rank - 1,
      len + jstride, jstride, &out[jstride], tag);
  if (rsend)
    dcuda_put_notify(ctx, wout, rank + 1,
      0, jstride, &out[len], tag);

  dcuda_wait_notifications(ctx, wout,
    DCUDA_ANY_SOURCE, tag, lsend + rsend);

  swap(in, out);
  swap(win, wout);
}
```

computation

communication [2]

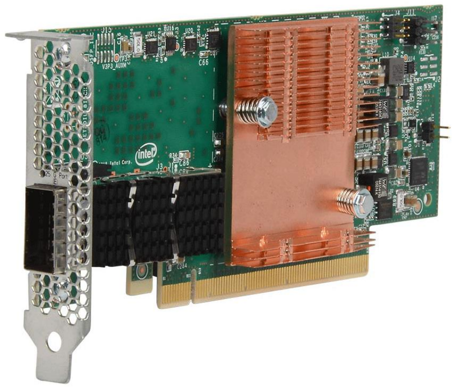
- iterative stencil kernel
- thread specific idx

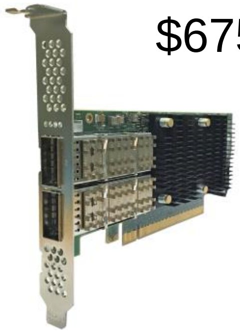
- map ranks to blocks
- device-side put/get operations
- notifications for synchronization
- shared and distributed memory

[1] T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, SC16

MSN are more cost-conscious, compared with ICN. Should MSN providers invest in HPC ICN to help drive down cost?

- **You mean they're cheap? ☺**
- **Cheap is not necessarily good!**
  - **Let's look at 100Gbit/s networking**





$675-$1,950 / NIC

$495 / NIC

$7,325 / 16 ports

$5,535 / 24 ports

Ethernet

OmniPath

- **Side note: operators tell me that the network is *not* the major cost in last-generation supercomputers (only 10-15%)**
  - **Corollary: we should start talking about GPUs, deep learning, and pricing ☺**

Will MSN and ICN converge in the future? If so, when and what will that network look like?

- **Yes …**
  - Economy of scale, no matter what it will be, it will be called Ethernet
  - But what is Ethernet? CEE/PFC or not?
  - The fundamental differences remain

    Lossless vs. lossy transport

    Adaptive vs. static routing

    Bare-metal vs. virtualized/tunneled
- **No …**
  - HPC's clean-slate approach fosters innovation

    Less cruft … (this is why I love this field as a scientist)
  - HPC may always leap ahead as it did in computer architecture

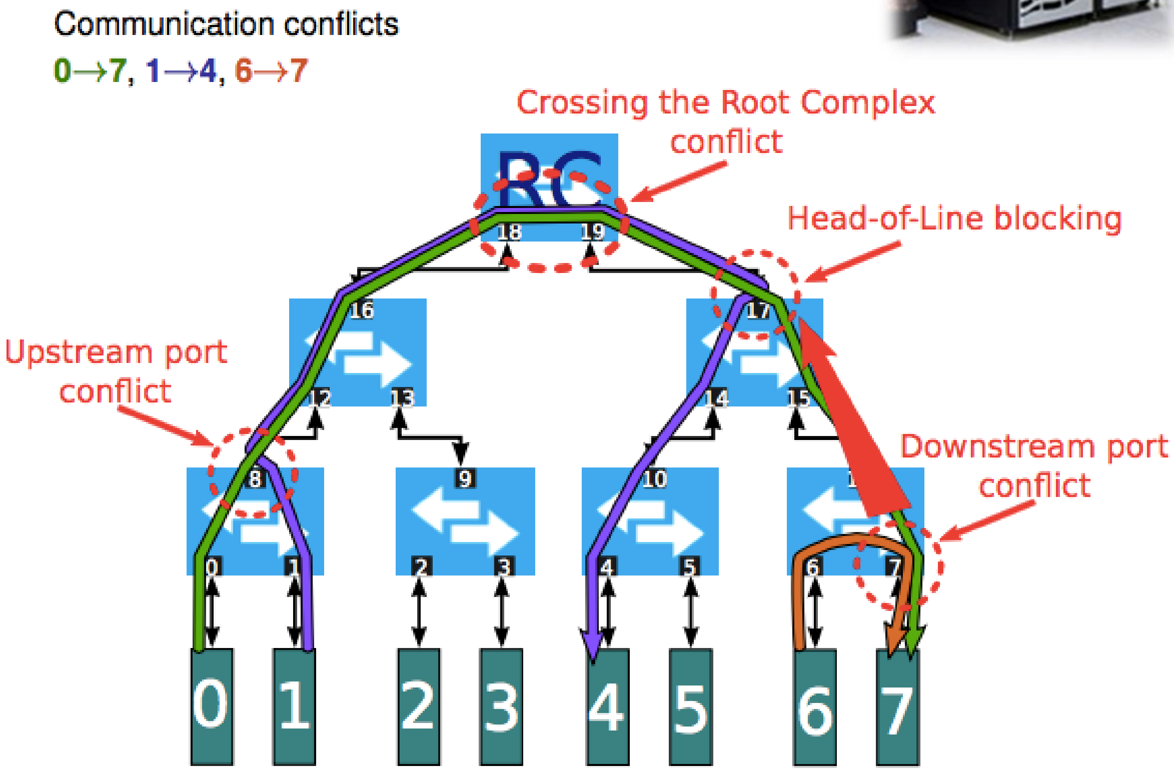
    Vectorization, GPUs, FPGAs …
  - … and network architecture

    Packet-level adaptive routing, lossless transport, RDMA

One last point …

- **It's all about the endpoints anyway!**
  - Most performance is lost at the endpoint, not in the network
    We have very good networks/topologies (Slim Fly [1] of course!)
  - E.g., latency – 50-70ns/hop, 600ns at endpoints
    … who to blame?

Example:
Cray CS Storm – MeteoSwiss supercomputer
2 cabinets, 12 hybrid computing nodes per cabinet
2 Intel Haswell 12-core CPUs per node
8 NVIDIA Tesla K80 GPU accelerators per node [2]



[1] M. Besta, T. Hoefler: Slim Fly: A Cost Effective Low-Diameter Network Topology , SC14
[2] M. Martinasso, et al.: A PCIe Congestion-Aware Performance Model for Densely Populated Accelerator Servers SC16