SquirrelJoin: Network Aware Distributed Join Processing with Lazy Partitioning

To appear in VLDB 2017

Lukas Rupprecht **William Culhane** Peter Pietzuch

Imperial College London

LSDS Large-Scale Distributed Systems Group

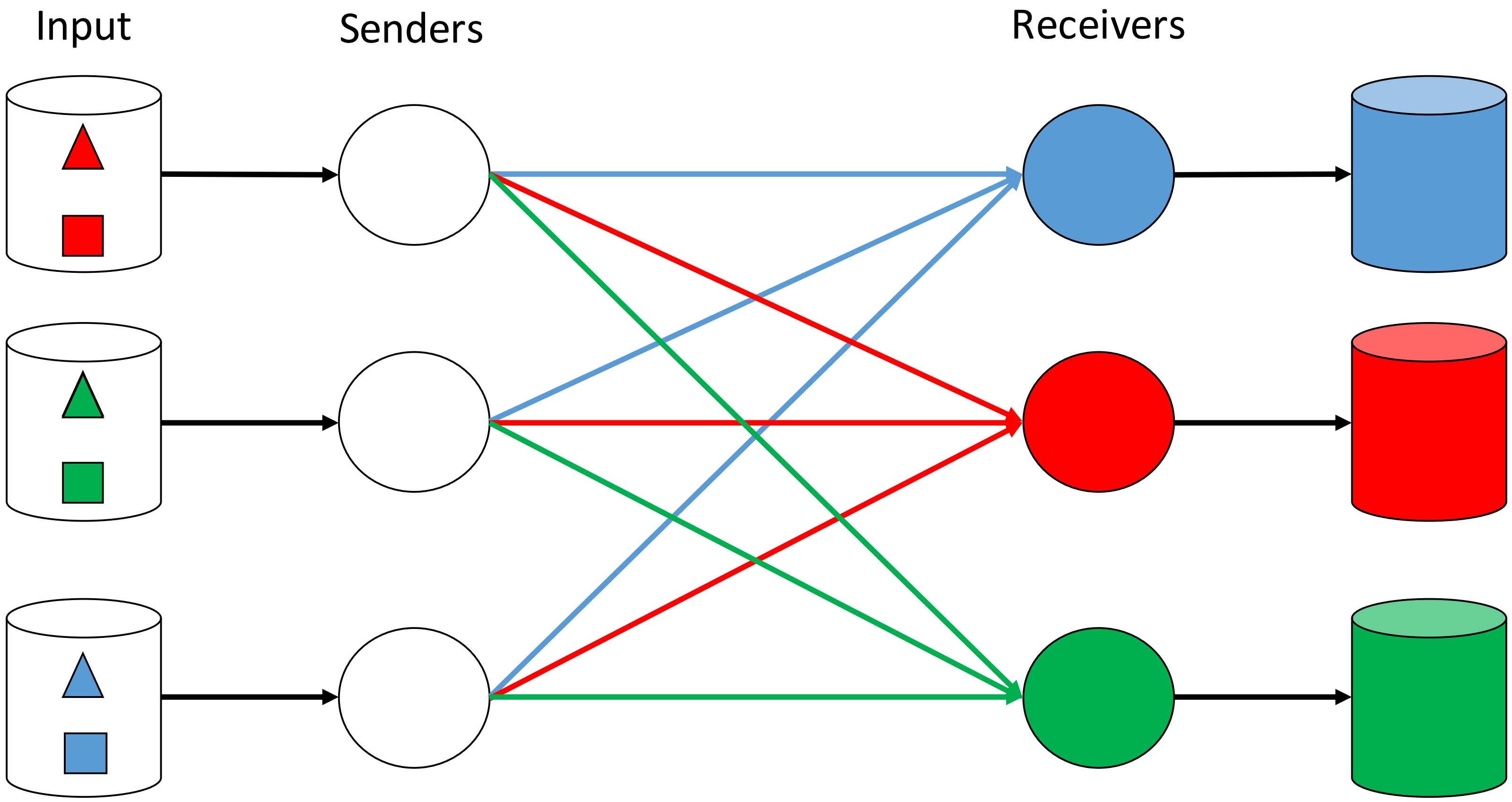
Funded by:

EPSRC Engineering and Physical Sciences Research Council

Background

- Data is getting big
  - AT&T’s phone record database is 312 TB (in 2014)
  - Walmart’s private cloud can process 2.5 PB per hour
- Analysis joins data across tables or databases
- Data is distributed among machines
- Shared networks are susceptible to skew from various applications

Distributed two-phase joins



Input

Senders

Receivers

△ Record from Table 1

□ Record from Table 2

Colors indicates join key partition

Network skew

- "Elephant" flows – 1% of flows, 90% of traffic
- Uneven distribution
  - 50th percentile of 10 concurrent flows per machine
  - 95th percentile of machine have >80 flows
  - 99.99th percentile of 1600 flows
- Daily occurrence

M. Alizadeh et al. Data center TCP (DCTCP). In SIGCOMM, 2010.
A. Greenberg et al. VL2: a scalable and flexible data center network. In SIGCOMM, 2009.
S. Kandula et al. The nature of data center traffic: measurements & analysis. In IMC, 2009.
T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In IMC, 2010.

Joins with network skew

- Up to 70% of join time in network transfers (worse for main memory systems)
- If the network is the bottleneck, slow network links can create stragglers
- Slowest straggler determines the join time

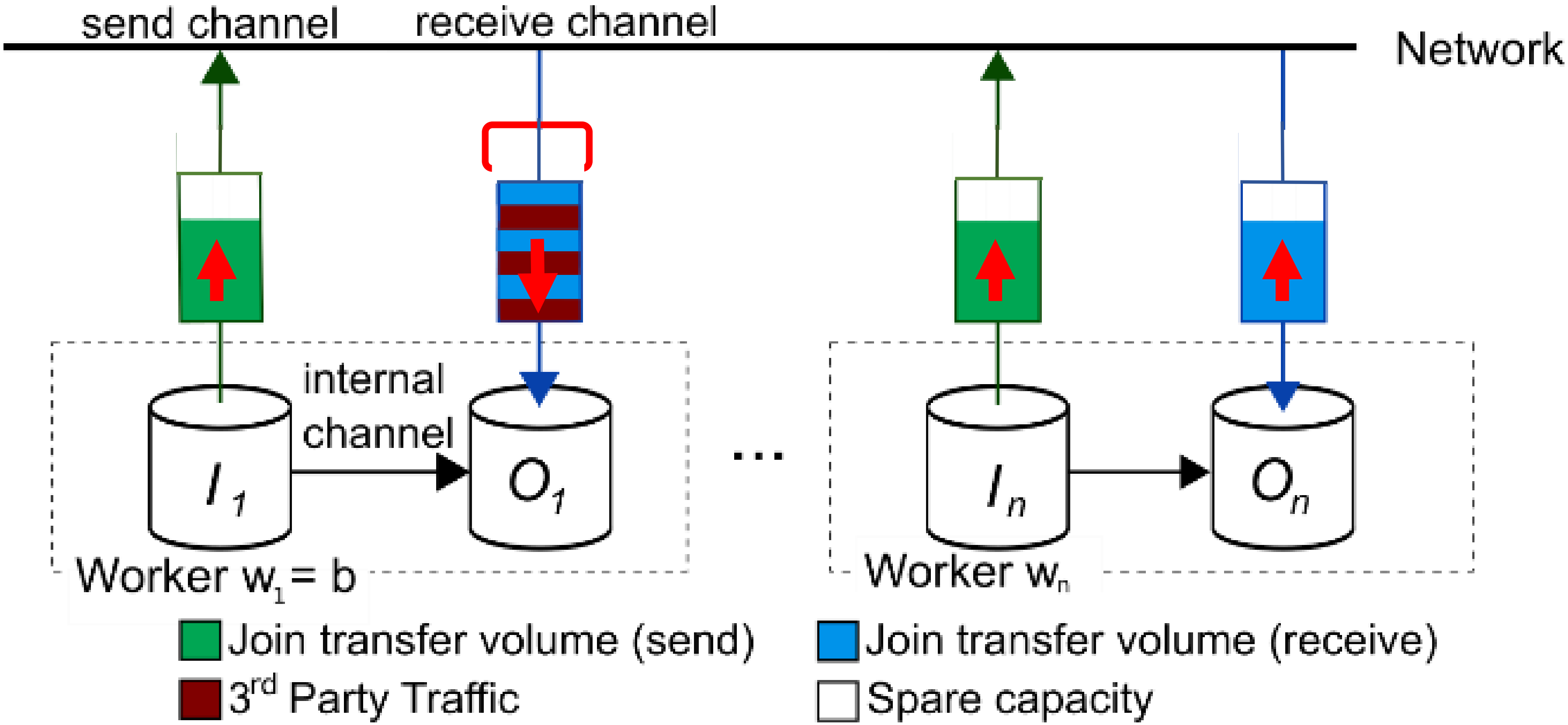
Polychroniou et al. Track join: distributed joins with minimal network traffic. In SIGMOD, 2014.
Rödiger et al. Locality-sensitive operators for parallel main-memory database clusters. In ICDE, 2014.

Problem statement

Solve big data.

Improve the completion time of two phase joins in the presence of network skew by offsetting the effects of that skew to mitigate stragglers with minimal overhead.

How can we address skew with repartitioning?



When do we run out of spare capacity?

Maximum time savings with a priori knowledge

Sender side bottleneck:

$$\frac{1-U_b}{U_b(n-1)+1} = \frac{1-.5}{.5(32-1)+1} = .03$$

Receiver side bottleneck:

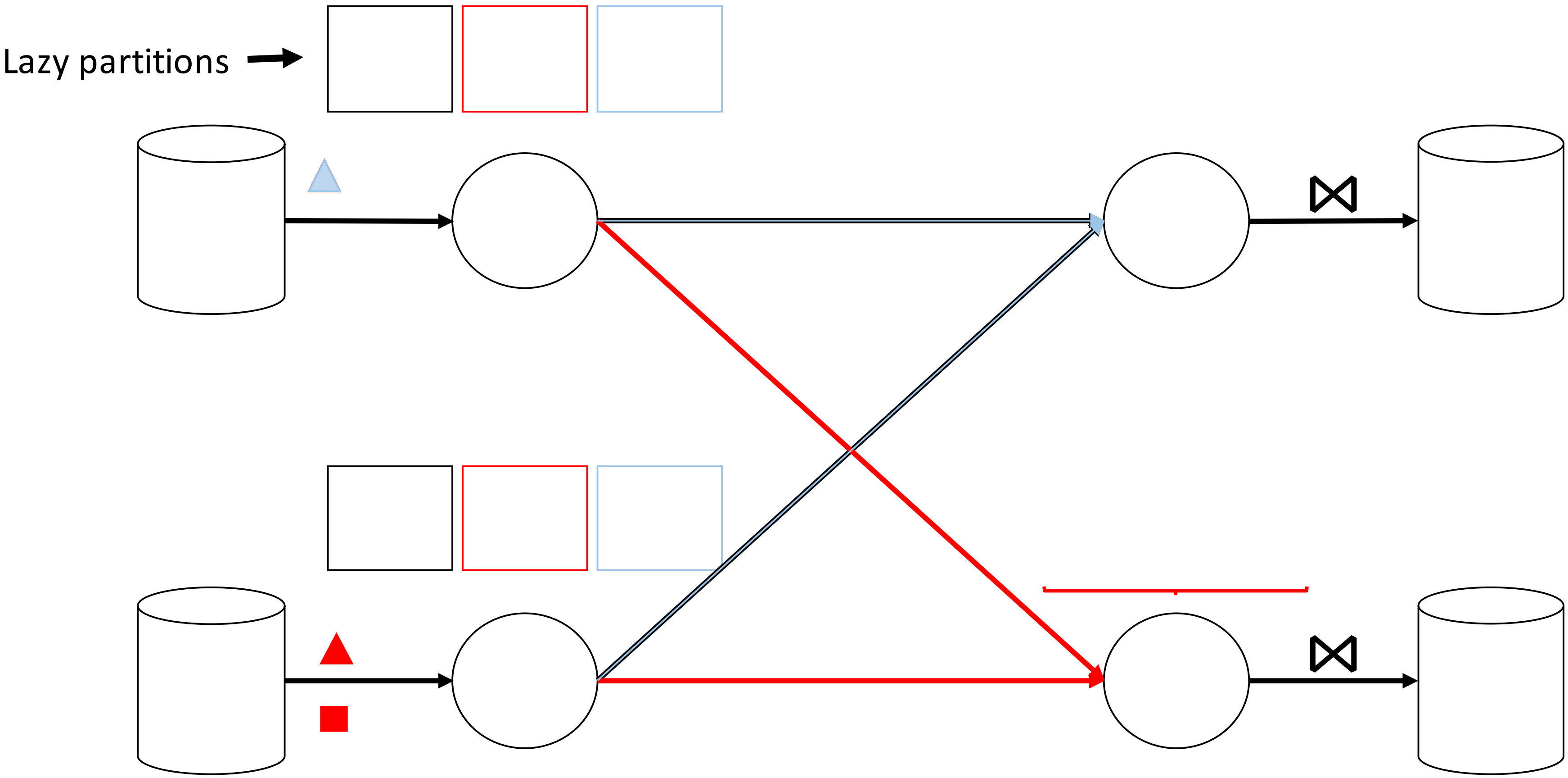
$$\frac{1-U_b}{\frac{U_b m}{n-m}+1} = \frac{1-.5}{\frac{.5*1}{32-1}+1} = .49$$

.5	U_b	- Fraction of bandwidth available on bottlenecks
32	n	- Nodes in the cluster
1	m	- Nodes with bottlenecks

Lazy partitioning

- Repartition *before* records sent via the network
- Buffer all records at senders
- Assign partitions to receivers when:
  - Run out of buffer space
  - Network skew is detected
  - There is no more join traffic
- Partitions with the same join key assigned together
- Buffered records sent after assignment

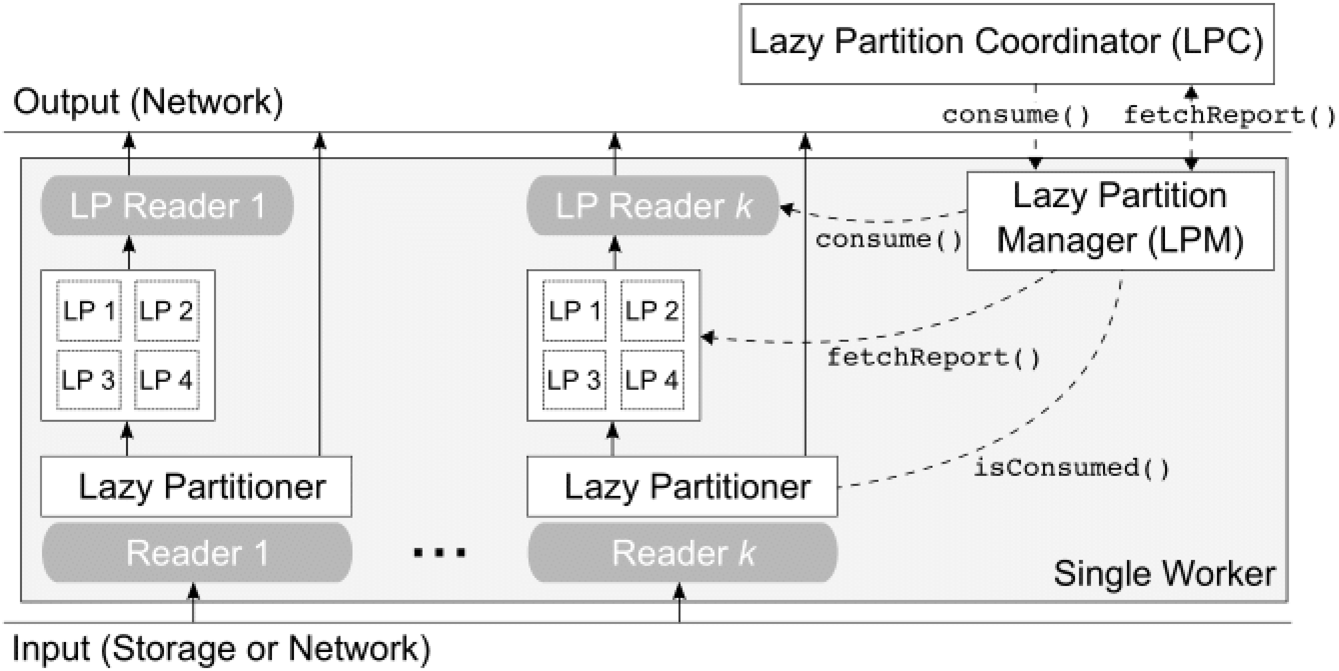
Lazy partitioning visualization



Assigning lazy partitions

```
Input : $\mathbb{W}: \texttt{set}, O_\varnothing: \texttt{set}, \tau_{skew}: \texttt{const}, \tau_{assign}: \texttt{const}, t: \texttt{now()}$
$\delta_{slowest} \leftarrow \max_{w_i \in \mathbb{W}}\{\frac{\rho(O_i,t)}{r(i,t)}\}$    Estimate the completion time of the straggler
$\mathbb{W}_{finished} \leftarrow \varnothing$    Initialize the “finished workers” set
if $\min_{w_i \in \mathbb{W}}\{\frac{\rho(O_i,t)}{r(i,t)}\} < \delta_{slowest}(1 - \tau_{skew})$ then    If there is detectable skew
    foreach $w_i \in \mathbb{W}$ do    Balance each worker
        $\delta_i \leftarrow \frac{\rho(O_i,t)}{r(i,t)}$    Estimate each worker’s completion time
        while $w_i \notin \mathbb{W}_{finished} \wedge \exists\{p_1 \ldots p_{|\mathbb{W}|}\} \in O_\varnothing$ do
            $\delta'_i \leftarrow \frac{\rho(O_i,t)}{r(i,t)} + \frac{|\{p_1 \ldots p_{|\mathbb{W}|}\}|}{r(i,t)}$    Estimate the change if assigned made
            if $\delta'_i > \delta_{slowest} \vee \delta'_i - \delta_i > \tau_{assign}$ then    If the assignment is too big
                $\mathbb{W}_{finished} \leftarrow \mathbb{W}_{finished} \cup \{w_i\}$    Mark the worker as balanced
            else
                $O_\varnothing \leftarrow O_\varnothing \setminus \{p_1 \ldots p_{|\mathbb{W}|}\}$    Assign the partitions to the worker
                $O_i \leftarrow O_i \cup \{p_1 \ldots p_{|\mathbb{W}|}\}$
```

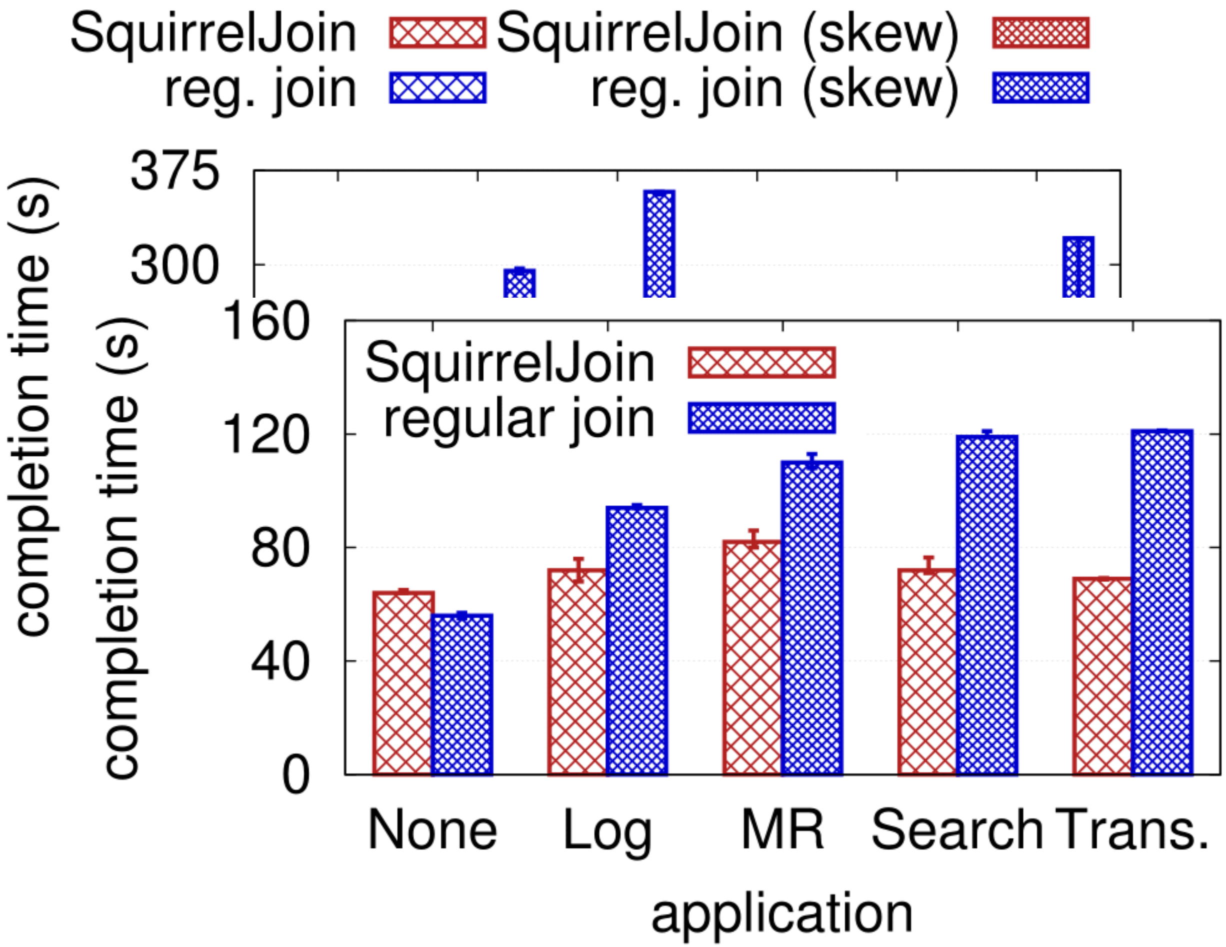
SquirrelJoin architecture



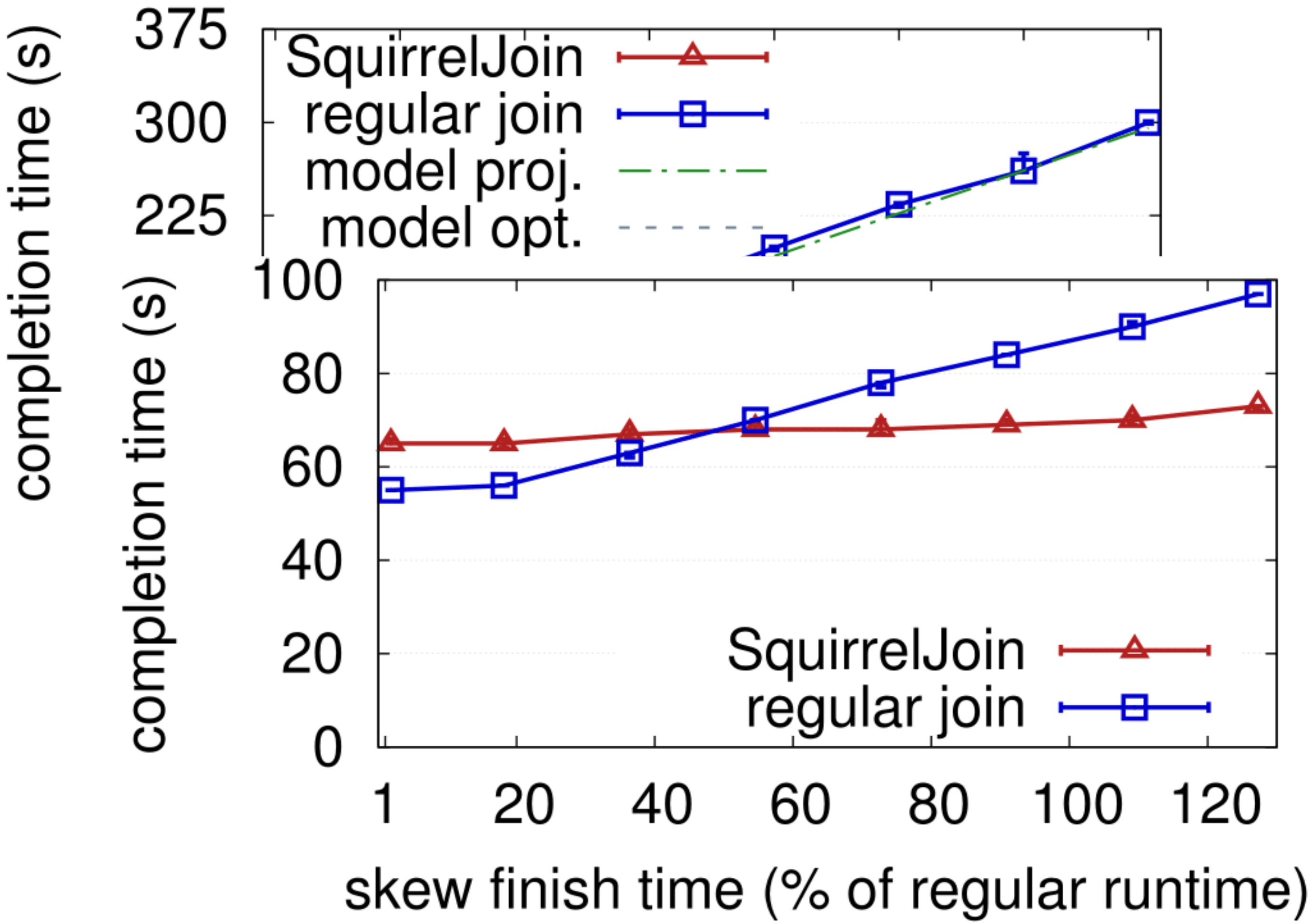
Experimental setup

- Implemented in Flink
- Google cloud ‘n1-standard-16’ machines
  - 16 CPUs at 2.5 GHz
  - 60 GB of RAM
- 1 master, 16 workers (15 saturated flows)
- Virtually limit the network interface to 1 Gbps
- Background applications sending from other machines
- TPC-H queries

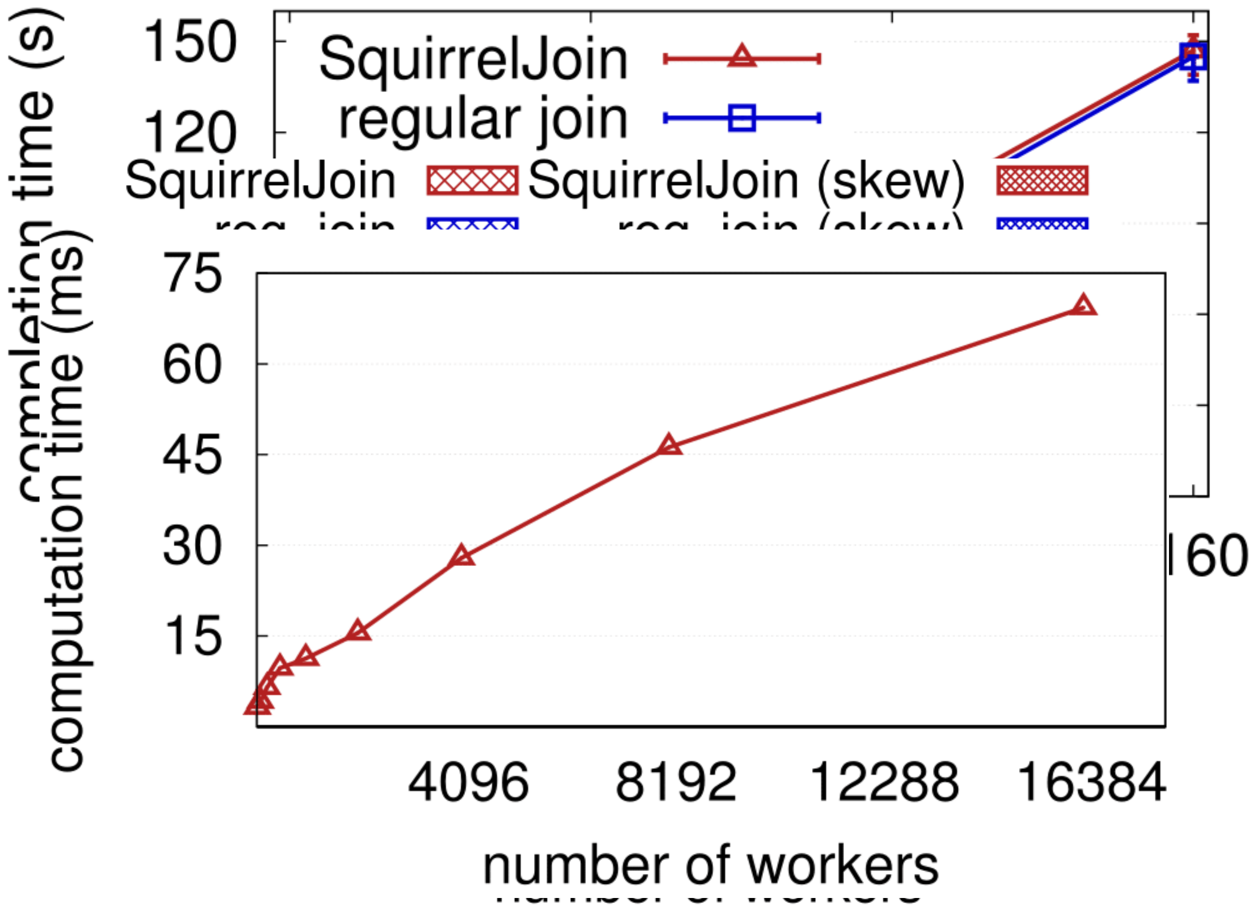
Results – workloads



Results – skew intensity



Results – scalability



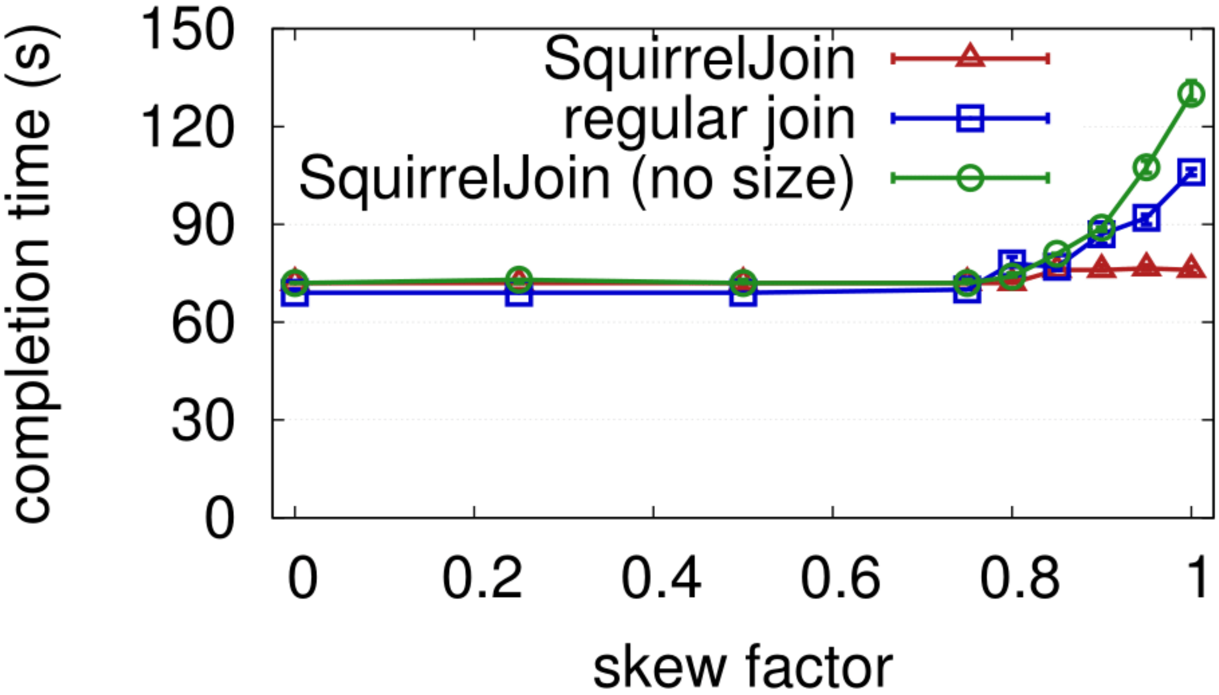
Conclusions

- Network skew is a problem for distributed joins
- Repartitioning can impact join completion time in the presence of skew (for receivers)
- Lazy partitioning is an elegant solution which captures most of the gains available from repartitioning
- SquirrelJoin is a lazy partitioning implementation in Flink which shows a successful impact on many real world workloads

Questions?

Data skew

- Track of assigned partition sizes and progress rates
- Synch point between the two phases (tradeoffs)
- What if the larger table is skewed:



Can we address sender side skew?

- Data is fixed at senders (file location)
- Using HDFS and replication we may be able to lazy partition input without hitting the network
- Requires identifying sender vs. receiver skew

Is the network really the bottleneck?

- We saturated a 1Gbps link
- Spark plus RAID SSD can saturate a 10 Gbps link
- IBM DB2 running in main memory can saturate a 40 Gbps link
- There is a recurring cycle of network vs local system improvement