Accelerating Analytical Workloads

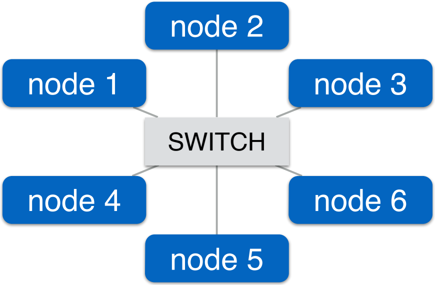
Thomas Neumann

Technische Universität München

April 15, 2014

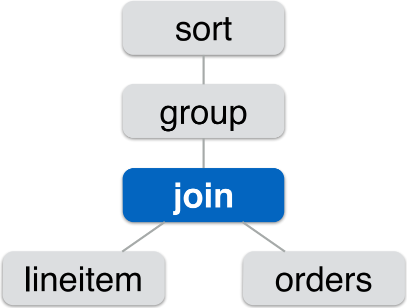
Scale Out in Big Data Analytics

- **Big Data** usually means data is distributed
- **Scale out** to process very large inputs
- but for analytics data has to be **combined** and **aggregated**
- typically map/reduce-based, Hadoop/Hive etc.
- data is copied to processing nodes for aggregations
- not very smart, dominated by network traffic
- smart data movement can speed up processing significantly



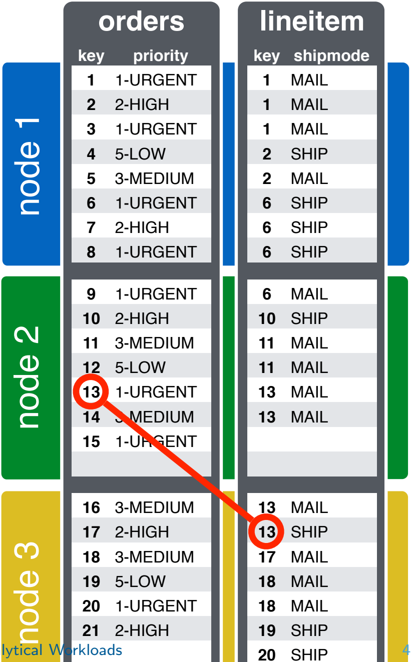
Running Example (1)

- Focus on **analytical** query processing in this talk
- TPC-H query 12 used as **running example**
- Runtime dominated by **join** orders $\bowtie$ lineitem
- Example from well-known benchmark, but applicable for all distributed joins



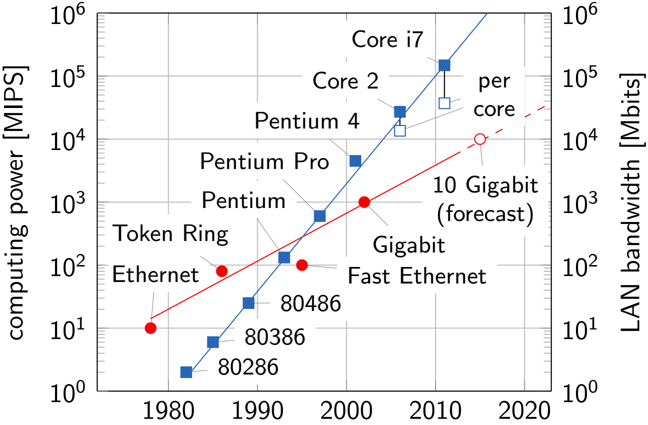
Running Example (2)

- Relations are **equally** distributed across nodes
- We make **no** assumptions on the data distribution
- Thus, tuples may join with tuples on **remote** nodes
- **Communication** over the network required



	orders		lineitem	
	key	priority	key	shipmode
node 1	1	1-URGENT	1	MAIL
	2	2-HIGH	1	MAIL
	3	1-URGENT	1	MAIL
	4	5-LOW	2	SHIP
	5	3-MEDIUM	2	MAIL
	6	1-URGENT	6	SHIP
	7	2-HIGH	6	SHIP
	8	1-URGENT	6	SHIP
node 2	9	1-URGENT	6	MAIL
	10	2-HIGH	10	SHIP
	11	3-MEDIUM	11	MAIL
	12	5-LOW	11	MAIL
	13	1-URGENT	13	MAIL
	14	3-MEDIUM	13	MAIL
	15	1-URGENT		
node 3	16	3-MEDIUM	13	MAIL
	17	2-HIGH	13	SHIP
	18	3-MEDIUM	17	MAIL
	19	5-LOW	18	MAIL
	20	1-URGENT	18	MAIL
	21	2-HIGH	19	SHIP
			20	SHIP

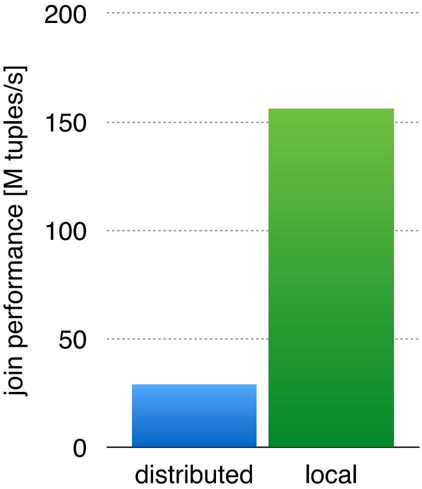
CPU vs. Network



CPU speed has grown much faster than **network bandwidth**

Scale Out: Network is the Bottleneck

- **Single node:** Performance is bound algorithmically
- **Cluster:** Network is bottleneck for query processing
- Investing time and effort in decreasing network traffic pays off
- **Goal:**
  Increase local processing to close the performance gap



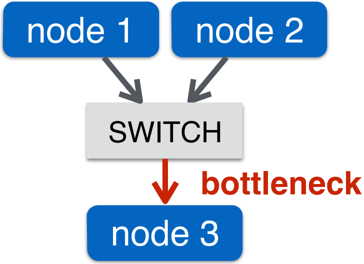
Neo-Join: Network-optimized Join [ICDE14]

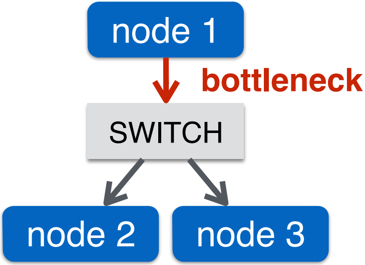
1. **Open Shop Scheduling**
   Efficient network communication

2. **Optimal Partition Assignment**
   Increase local processing

3. **Selective Broadcast**
   Handle value skew

Bandwidth Sharing

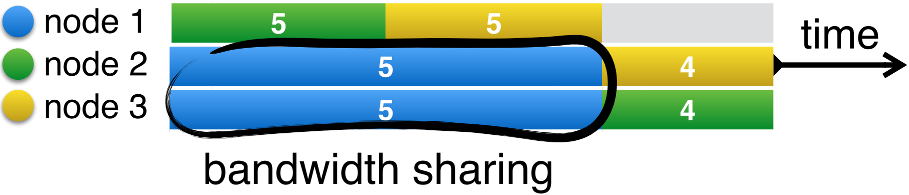




- Simultaneous use of a single link creates a **bottleneck**
- **Reduces bandwidth** by at least a factor of **2**

Naïve Schedule

Technische Universität München

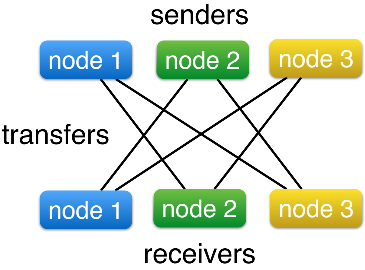


- Node 2 and 3 send to node 1 **at the same time**
- Bandwidth sharing increases **network duration** significantly

Open Shop Scheduling (1)

Avoiding bandwidth sharing translates to **open shop scheduling:**

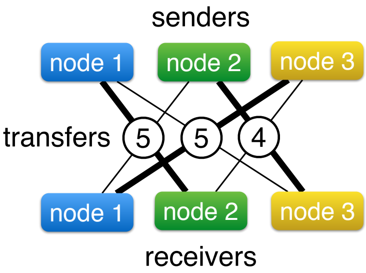
- A **sender** has one **transfer** per **receiver**
- A receiver should receive at most **one** transfer at a time
- A sender should send at most **one** transfer at a time



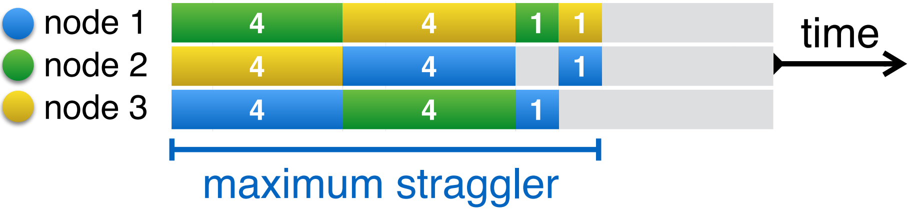
Open Shop Scheduling (2)

Compute optimal schedule:

- **Edge weights** represent total transfer duration
- Scheduler repeatedly finds **perfect matchings**
- Each matching specifies one communication **phase**
- Transfers in a phase will **never** share bandwidth



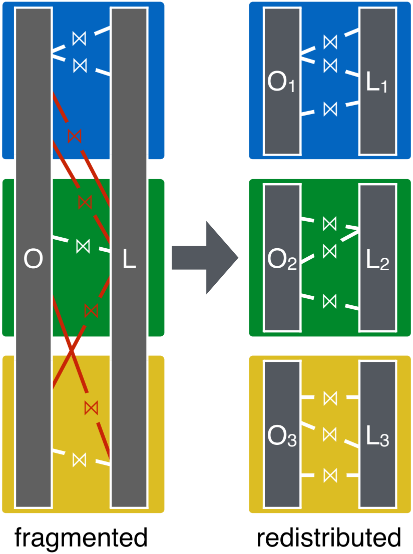
Optimal Schedule



- Open shop schedule achieves minimal **network duration**
- Schedule duration determined by **maximum straggler**

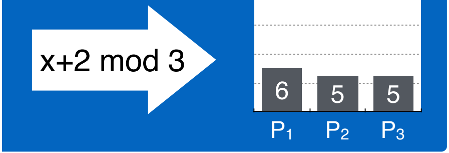
Distributed Join

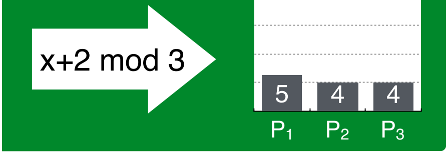
- Tuples may join with tuples on **remote nodes**
- Repartition and redistribute **both relations** for local join
- Tuples will join only with the **corresponding partition**
- Using hash, range, radix, or other **partitioning** scheme
- **In any case:** Decide how to **assign** partitions to nodes

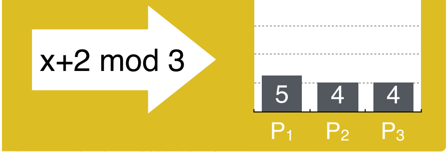


Running Example: Hash Partitioning

	orders key	orders priority	lineitem key	lineitem shipmode
node 1	1	1-URGENT	1	MAIL
	2	2-HIGH	1	MAIL
	3	1-URGENT	1	MAIL
	4	5-LOW	2	SHIP
	5	3-MEDIUM	2	MAIL
	6	1-URGENT	6	SHIP
	7	2-HIGH	6	SHIP
	8	1-URGENT	6	SHIP
node 2	9	1-URGENT	6	MAIL
	10	2-HIGH	10	SHIP
	11	3-MEDIUM	11	MAIL
	12	5-LOW	11	MAIL
	13	1-URGENT	13	MAIL
	14	3-MEDIUM	13	MAIL
	15	1-URGENT		
node 3	16	3-MEDIUM	13	MAIL
	17	2-HIGH	13	SHIP
	18	3-MEDIUM	17	MAIL
	19	5-LOW	18	MAIL
	20	1-URGENT	18	MAIL
	21	2-HIGH	19	SHIP
			20	SHIP



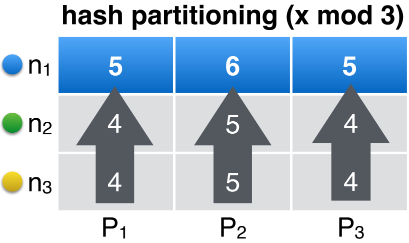


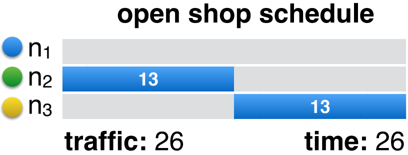


Assign Partitions to Nodes (1)

Option 1: Minimize network traffic

- Assign partition to node that owns its **largest part**
- Only the **small fragments** of a partition sent over the network
- Schedule with minimal network traffic may have **high duration**

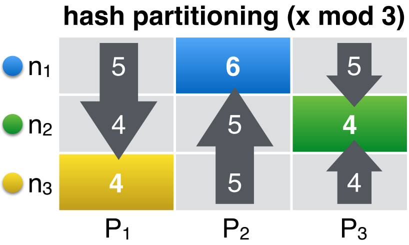




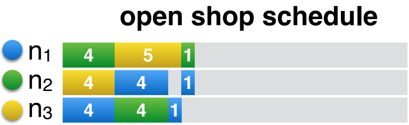
Assign Partitions to Nodes (2)

Option 2: Minimize response time:

- **Query response time** is time from request to result
- Query response time dominated by **network duration**
- To minimize network duration, minimize **maximum straggler**



hash partitioning (x mod 3)



open shop schedule

traffic: 28 **time:** 10

Minimize Maximum Straggler

- Formalized as mixed-integer **linear program**
- Shown to be **NP-hard** in worst case
- But in practice **fast enough** using CPLEX or Gurobi ($<$ 0.5 % overhead for 32 nodes, 200 M tuples each)
- Partition assignment can optimize **any partitioning**

minimize w, subject to

$$
\begin{aligned}
w &\geq \sum_{j=0}^{p-1} h_{ij}(1 - x_{ij}) && 0 \leq i < n \\
w &\geq \sum_{j=0}^{p-1} \left(x_{ij} \sum_{k=0, i \neq k}^{n-1} h_{kj} \right) && 0 \leq i < n \\
1 &= \sum_{i=0}^{n-1} x_{ij} && 0 \leq j < p
\end{aligned}
$$

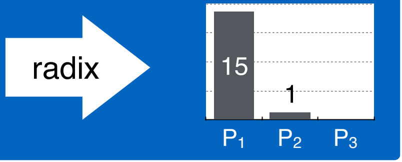
Running Example: Locality

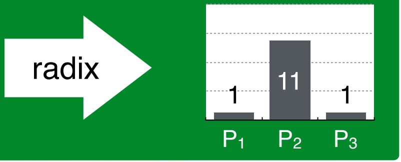
orders

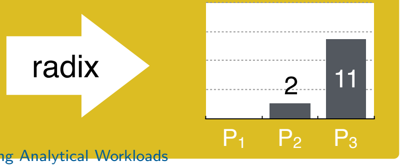
	key	priority
node 1	1	1-URGENT
	2	2-HIGH
	3	1-URGENT
	4	5-LOW
	5	3-MEDIUM
	6	1-URGENT
	7	2-HIGH
	8	1-URGENT
node 2	9	1-URGENT
	10	2-HIGH
	11	3-MEDIUM
	12	5-LOW
	13	1-URGENT
	14	3-MEDIUM
	15	1-URGENT
node 3	16	3-MEDIUM
	17	2-HIGH
	18	3-MEDIUM
	19	5-LOW
	20	1-URGENT
	21	2-HIGH

lineitem

	key	shipmode
node 1	1	MAIL
	1	MAIL
	1	MAIL
	2	SHIP
	2	MAIL
	6	SHIP
	6	SHIP
	6	SHIP
node 2	6	MAIL
	10	SHIP
	11	MAIL
	11	MAIL
	13	MAIL
	13	MAIL
node 3	13	MAIL
	13	SHIP
	17	MAIL
	18	MAIL
	18	MAIL
	19	SHIP
	20	SHIP



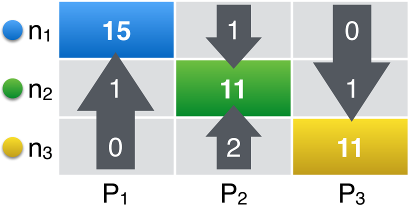




Locality

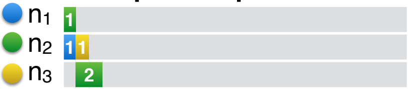
- Running example exhibits **time-of-creation** clustering
- **Radix repartitioning** on most significant bits retains locality
- Partition assignment can **exploit locality**
- Significantly reduces **query response time**

radix partitioning (MSB)



	P_1	P_2	P_3
n_1	15	1	0
n_2	1	11	1
n_3	0	2	11

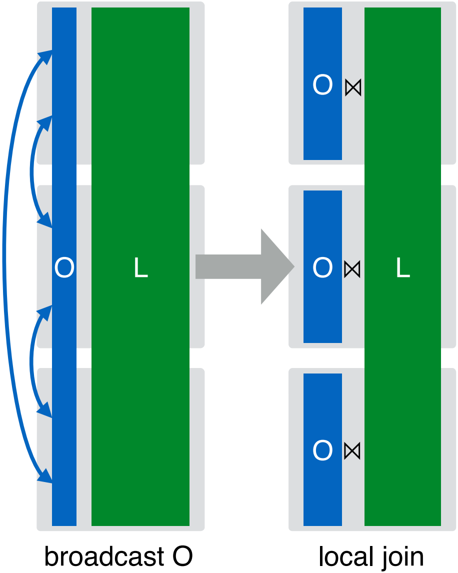
open shop schedule



traffic: 5 **time:** 3

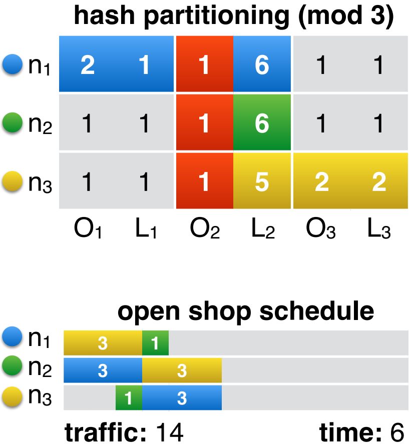
Broadcast

- **Alternative** to data repartitioning
- **Replicate** the smaller relation between all nodes
- Larger relation **remains fragmented** across nodes



Selective Broadcast

- Decide **per partition** whether to assign or broadcast
- **Broadcast** orders for P_2, let line items remain fragmented
- **Assign** the other partitions taking locality into account
- Improves performance for high **skew** and many **duplicates**



hash partitioning (mod 3)

	O_1	L_1	O_2	L_2	O_3	L_3
n_1	2	1	1	6	1	1
n_2	1	1	1	6	1	1
n_3	1	1	1	5	2	2

open shop schedule

traffic: 14 **time:** 6

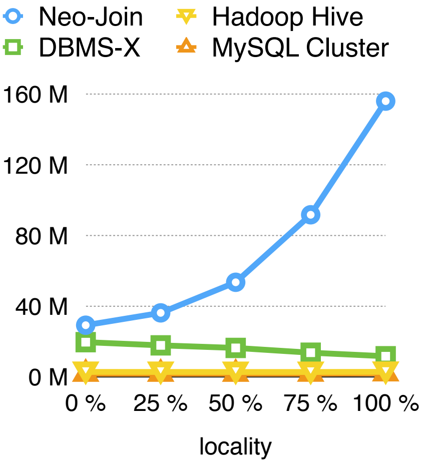
Experimental Setup

- Cluster of 4 nodes
- Core i7, 4 cores, 3.4 GHz, 32 GB RAM
- Gigabit Ethernet
- Tuples consist of 64 bit key, 64 bit payload

Locality

Technische Universität München

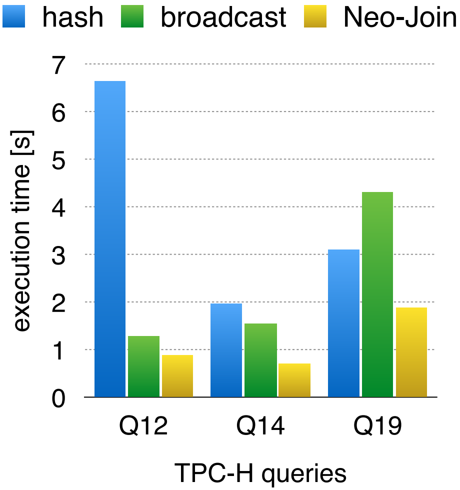
- Vary **locality** from **0 %** (uniform distribution) to **100 %** (range partitioning)
- Neo-Join improves **join performance** from 29 M to 156 M tuples/s (> 500 %)
- 3 nodes, 600 M tuples



TPC-H Results (scale factor 100)

Technische Universität München

- Results for three selected **TPC-H** queries
- **Broadcast** outperforms **hash** for large relation size differences
- Neo-Join always performs better due to **selective broadcast** and **locality**
- 4 nodes, ca. 100GB data



Further Optimizations

Network-aware joining is only one ingredient

- All **Query Processing** steps are important
  - parallel, network aware, maximize locality [PVDB12]
  - group by, sort, cube, ... [DEBUL14, SIGMOD13, PVLDB11]
  - also: smart loading/parsing [PVLDB13]
- **Query Optimization** has a huge impact
  - Reformulate the query into a more efficient form [EDBT14,ICDE12]
  - Involves algebraic optimization, exploiting statistics, etc. [ICDE11]
  - Can improve runtimes by orders of magnitude!

Result is much faster than a naive map/reduce approach.

Conclusion

Analyzing Big Data is challenging

- very large volume, distributed
- many operations require joining data
- network is a bottleneck

We can use optimization techniques to speed up the analysis

- maximize bandwidth
- exploit data characteristics (locality, skew, etc.)
- smart scheduling of operations

Improves over commonly used approaches like Hive by order of magnitudes.