CSE 544: Principles of Database Systems

Parallel Databases

Announcements

- Project proposals were due last night
- MapReduce paper review due on Wednesday
- HW2 due on Sunday, May 6th

Overview of Today's Lecture

- Discuss in class the *Consistent Selectivity Estimation* paper
- Parallel databases (Chapter 22.1 – 22.5)

Parallel v.s. Distributed Databases

- Parallel database system:
  - Improve performance through parallel implementation
  - Will discuss in class
- Distributed database system:
  - Data is stored across several sites, each site managed by a DBMS capable of running independently
  - Will not discuss in class

Parallel DBMSs

- Goal
  - Improve performance by executing multiple operations in parallel
- Key benefit
  - Cheaper to scale than relying on a single increasingly more powerful processor
- Key challenge
  - Ensure overhead and contention do not kill performance

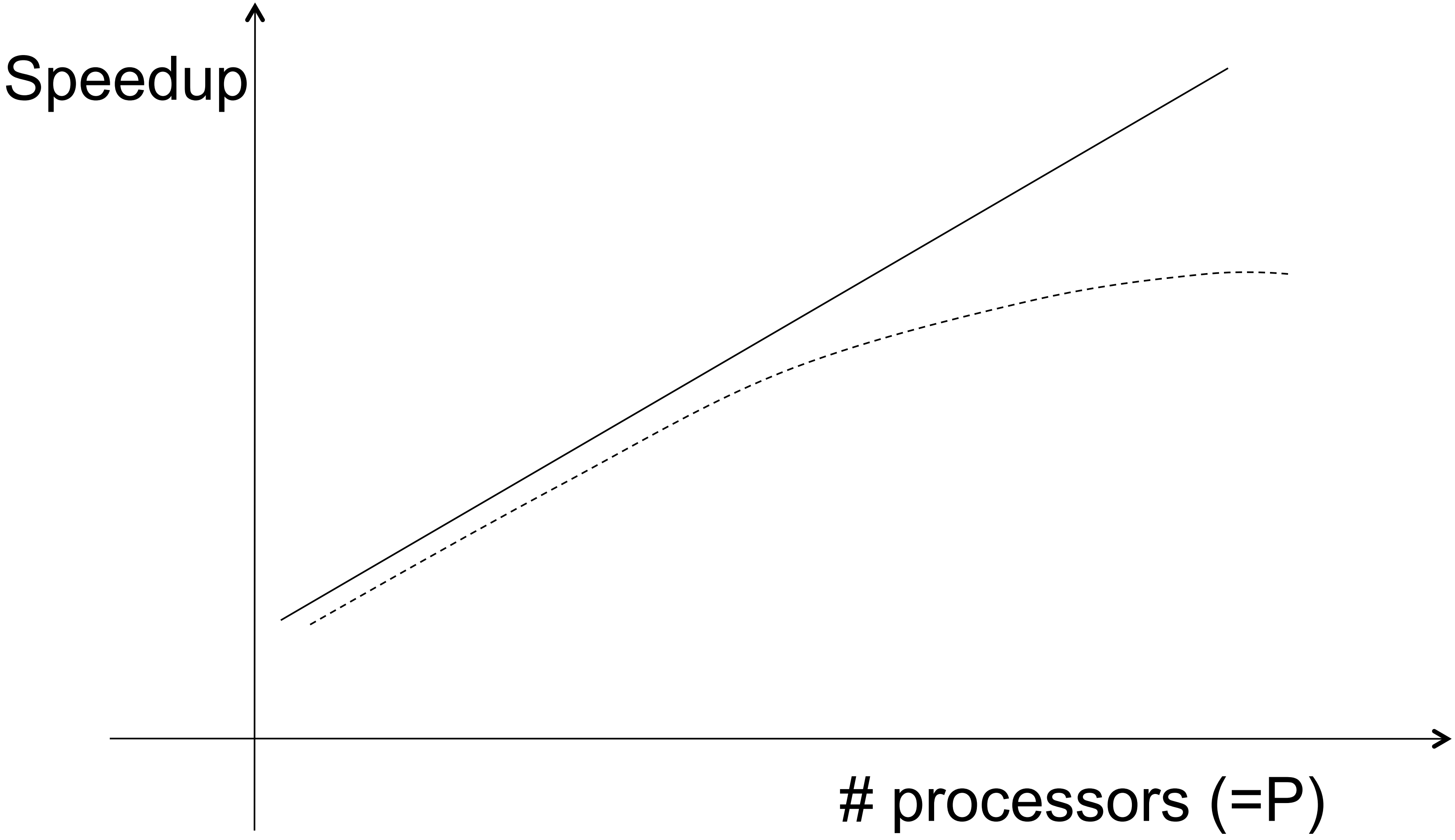
Performance Metrics for Parallel DBMSs

- Speedup
  - More processors → higher speed

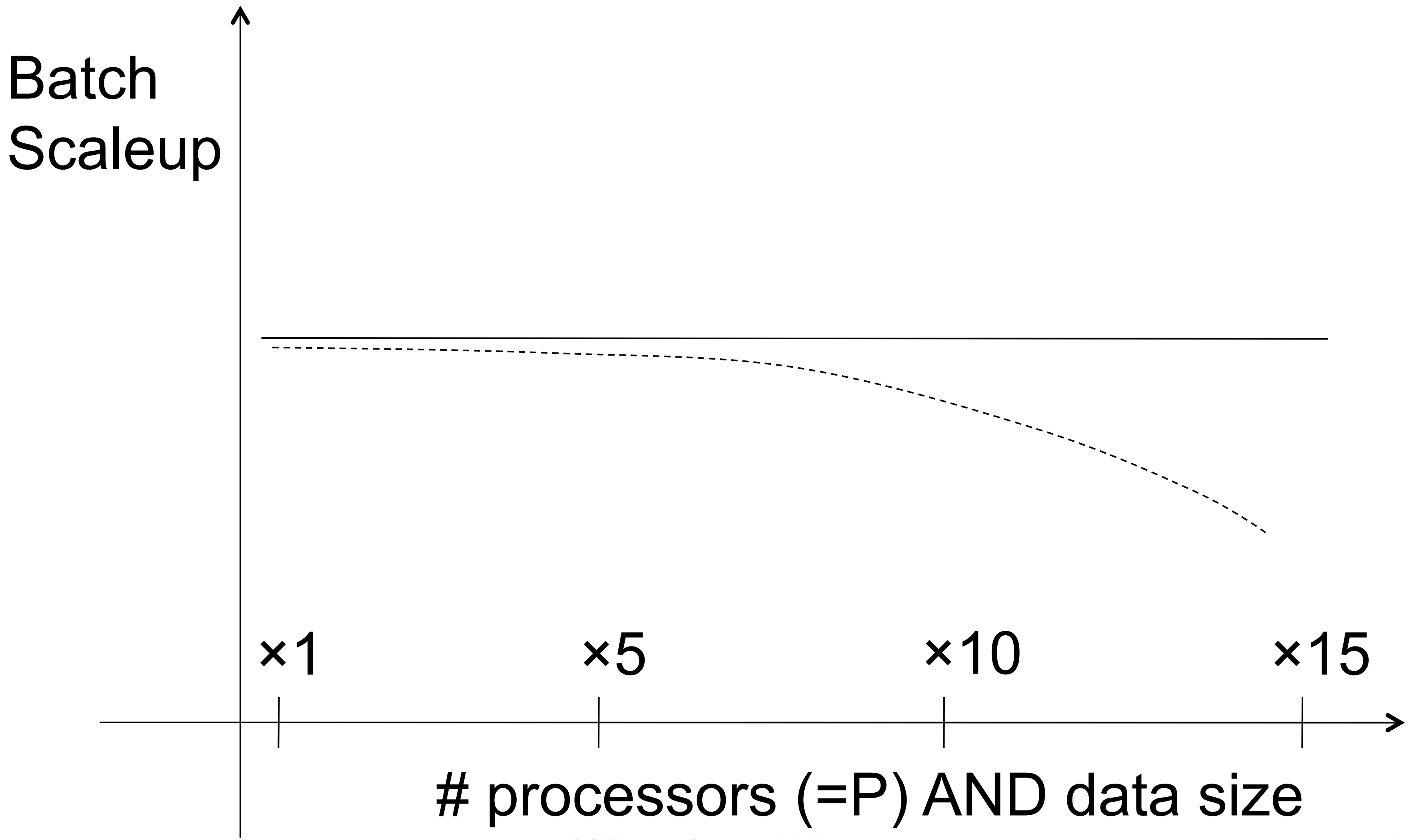
- Scaleup
  - More processors → can process more data

- Batch scaleup/speedup
  - Decision Support: individual query should run faster (speedup) or same speed (scaleup)
- Transaction scaleup/speedup
  - OLTP: Transactions Per Second (TPS) should increase (speedup) or should stay constant (scaleup)

Linear v.s. Non-linear Speedup



Linear v.s. Non-linear Scaleup



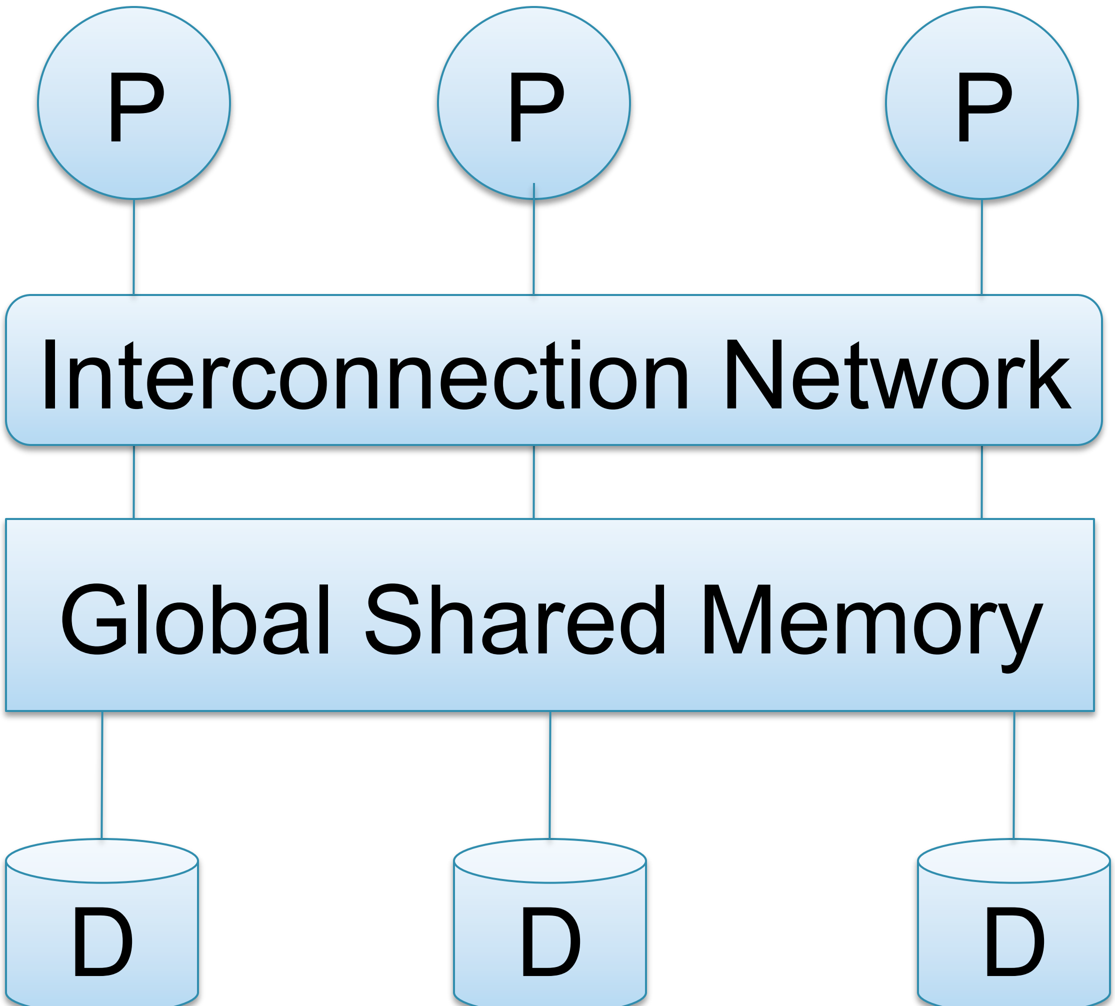
Challenges to Linear Speedup and Scaleup

- Startup cost
  - Cost of starting an operation on many processors
- Interference
  - Contention for resources between processors
- Skew
  - Slowest processor becomes the bottleneck

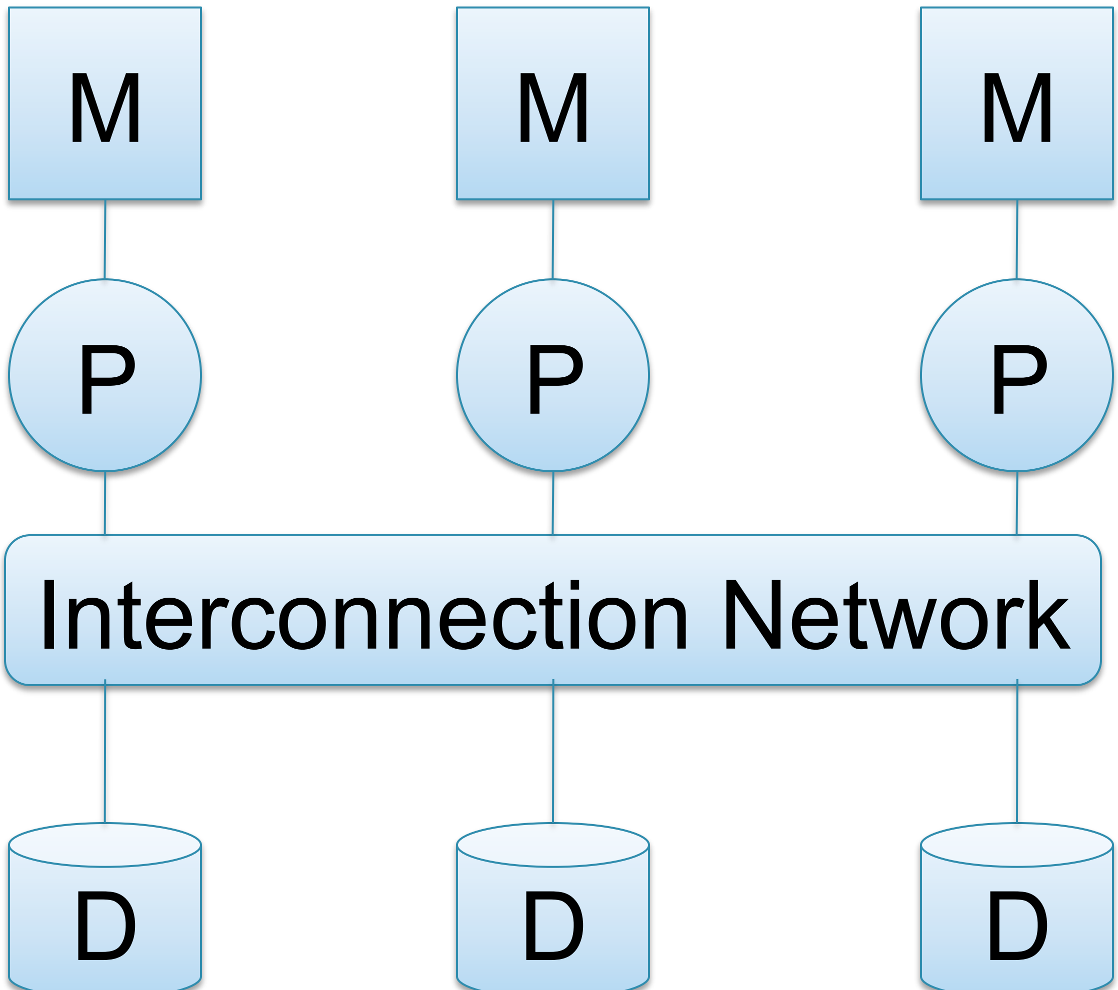
Architectures for Parallel Databases

- Shared memory
- Shared disk
- Shared nothing

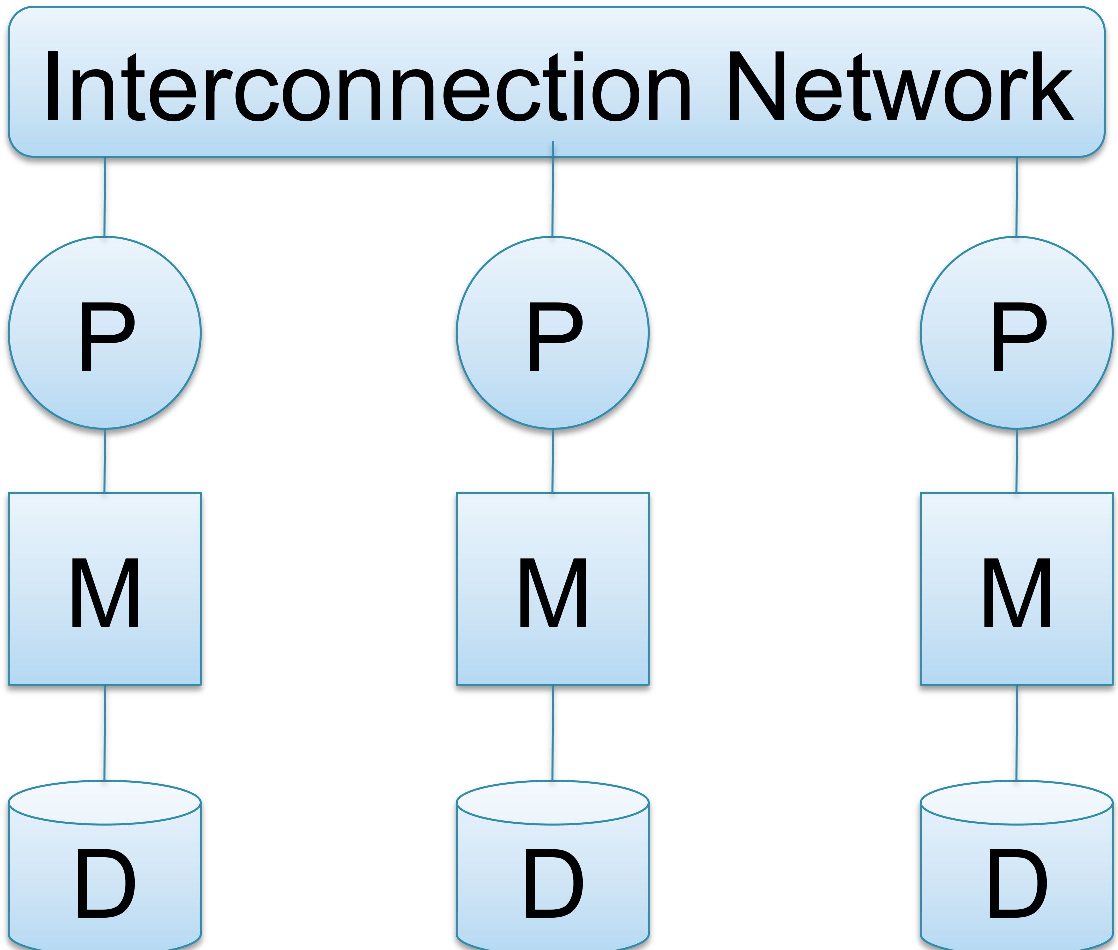
Shared Memory



Shared Disk



Shared Nothing

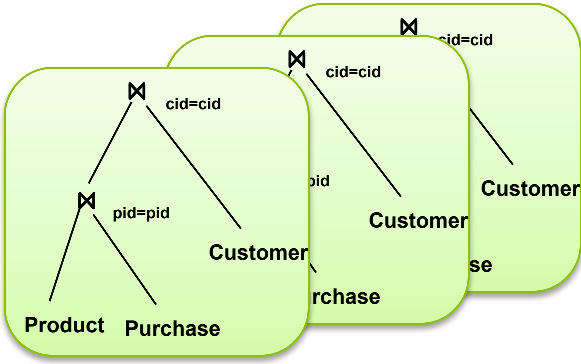


Shared Nothing

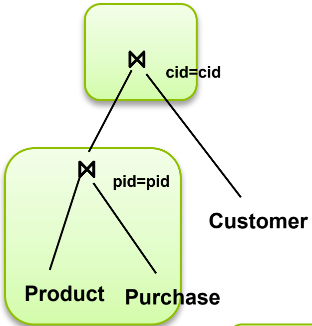
- Most scalable architecture
  - Minimizes interference by minimizing resource sharing
  - Can use commodity hardware
  - Terminology: processor = server = node
  - P = number of nodes
- Also most difficult to program and manage

Taxonomy

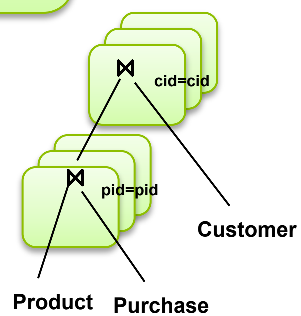
- Inter-query parallelism
  - Transaction per node
  - OLTP



- Inter-operator parallelism
  - Operator per node
  - Both OLTP and Decision Support



- Intra-operator parallelism
  - Operator on multiple nodes
  - Decision Support



We study only intra-operator parallelism: most scalable

Review in Class

Basic query processing on one node.

Given relations R(A,B) and S(B, C), compute:

- Selection: $\sigma_{A=123}(R)$
- Group-by: $\gamma_{A,sum(B)}(R)$
- Join: $R \bowtie S$

Horizontal Data Partitioning

- Partition a table R(K, A, B, C) into P chunks $R_1, \ldots, R_P$, stored at the P nodes
- Block Partition: $size(R_1) \approx \ldots \approx size(R_P)$
- Hash partitioned on attribute A:
  - Tuple t goes to chunk $i = (h(t.A) \bmod P) + 1$
- Range partitioned on attribute A:
  - Partition the range of A into $-\infty = v_0 < v_1 < \ldots < v_P = \infty$
  - Tuple t goes to chunk i, if $v_{i-1} \leq t.A < v_i$

Parallel GroupBy

R(K,A,B,C), discuss in class how to compute these GroupBy's, for each of the partitions

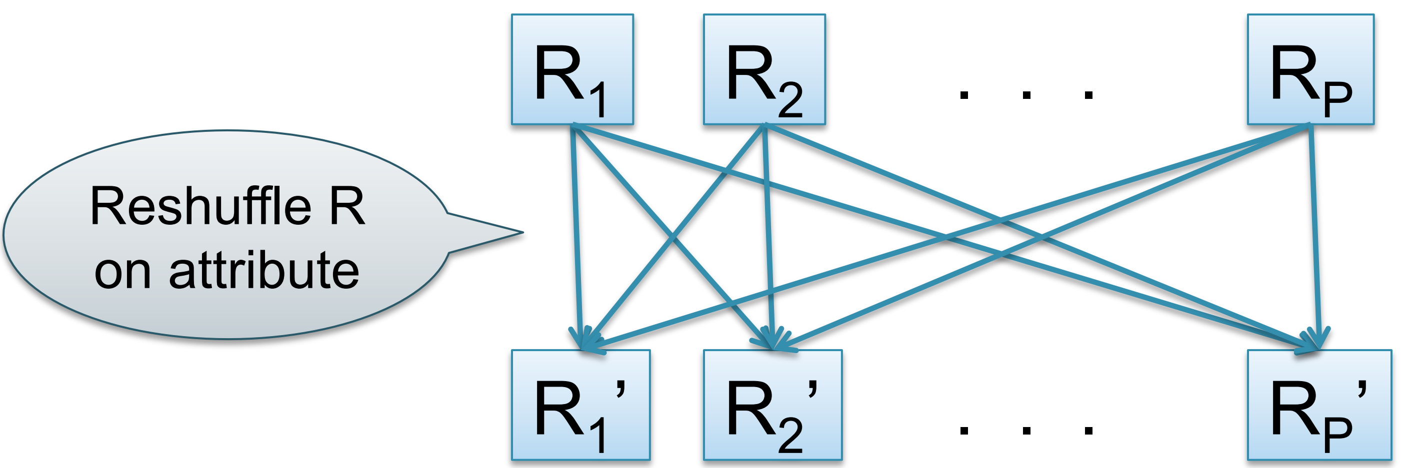
- $\gamma_{A,sum(C)}(R)$

- $\gamma_{B,sum(C)}(R)$

Parallel GroupBy

$\gamma_{A,sum(C)}(R)$

- If R is partitioned on A, then each node computes the group-by locally
- Otherwise, hash-partition R(<u>K</u>,A,B,C) on A, then compute group-by locally:



Speedup and Scaleup

- The runtime is dominated by the time to read the chunks from disk, i.e. size(R_i)
- If we double the number of nodes P, what is the new running time of $\gamma_{A,sum(C)}(R)$?
- If we double both P and the size of the relation R, what is the new running time?

Uniform Data v.s. Skewed Data

- Uniform partition:
  - $size(R_1) \approx \ldots \approx size(R_P) \approx size(R) / P$
  - Linear speedup, constant scaleup

- Skewed partition:
  - For some i, $size(R_i) \gg size(R) / P$
  - Speedup and scaleup will suffer

Uniform Data v.s. Skewed Data

- Let R(K,A,B,C); which of the following partition methods may result in skewed partitions?

- Block partition

- Hash-partition
  - On the key K
  - On the attribute A

- Range-partition
  - On the key K
  - On the attribute A

Uniform Data v.s. Skewed Data

- Let R(K,A,B,C); which of the following partition methods may result in skewed partitions?

- Block partition — Uniform

- Hash-partition
  - On the key K — Uniform — Assuming uniform hash function
  - On the attribute A — May be skewed — E.g. when all records have the same value of the attribute A, then all records end up in the same partition

- Range-partition
  - On the key K
  - On the attribute A — May be skewed — Difficult to partition the range of A uniformly.

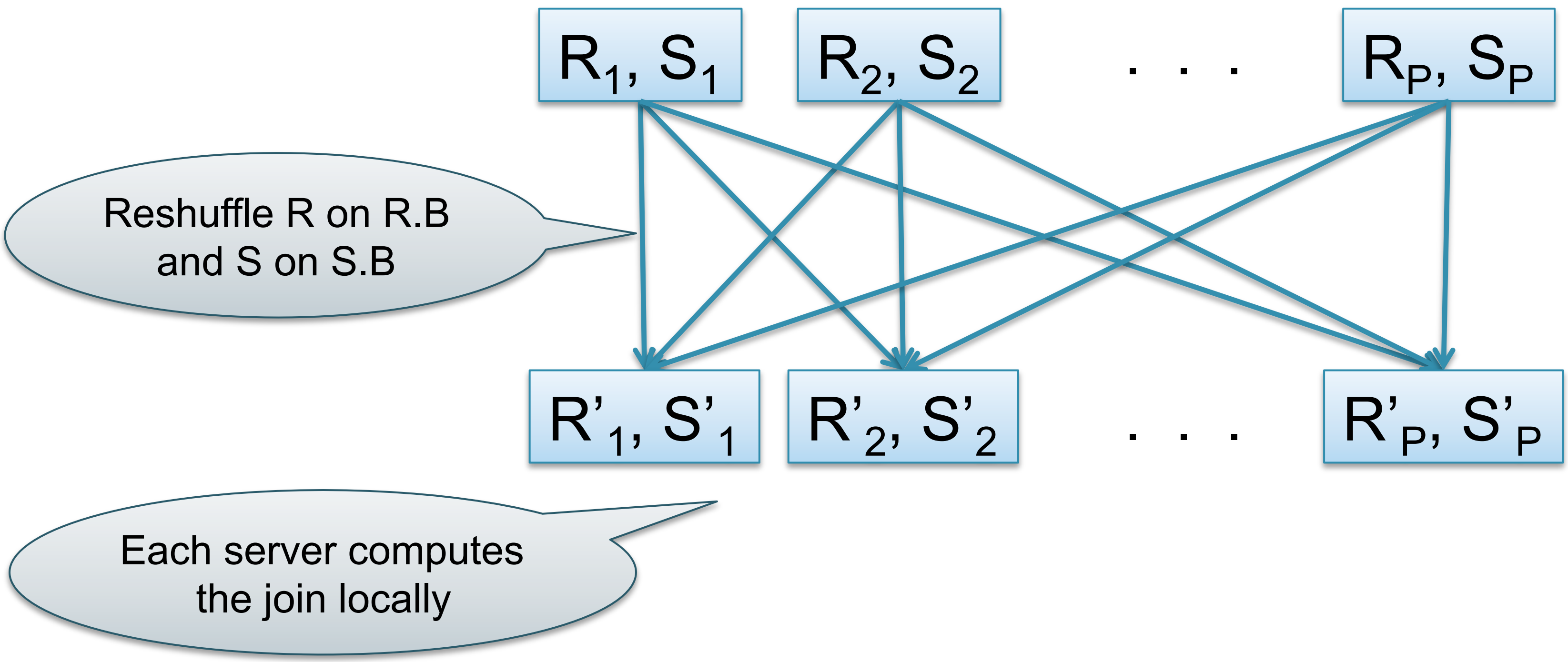
Parallel Join

- In class: compute R(A,B) ⋈ S(B,C)

R_1, S_1 | R_2, S_2 | . . . | R_P, S_P

Parallel Join

- In class: compute R(A,B) ⋈ S(B,C)



Parallel Query Plans

- Same relational operators
- Add special split and merge operators
  - Handle data routing, buffering, and flow control
- Example: exchange operator
  - Inserted between consecutive operators in the query plan