Introduction to Data Management
CSE 344

Lecture 21: Parallel Databases

Announcements

- HW6 is posted: due Friday, March 9
  - Need to access Amazon Web Services
  - You have $100 credit – more than enough
  - Need to learn PigLatin: Thursday/Friday
  - Quickly learn PigLatin: use starter code
- Next four lectures: parallel databases
  - Traditional, MapReduce+PigLatin
- Wednesday, March 7
  - Guest lecture by Prof. Balazinska: No-SQL
- Wednesday, March 9
  - Final review by Paris Koutris

Parallel Computation Today

Two MAJOR trends that are pushing Computer Science toward parallel computation:

1. Change in Moore's law* (exponential growth in transistors per chip density) no longer results in increased clock speeds.
   - Increased hardware performance will be available only through parallelism.
   - Think multicore: 4 cores today, perhaps 64 in a few years.

2. Cloud computing commoditizes access to large compute clusters.
   - Ten years ago, only google could afford 1000 servers;
   - Today you can rent this from Amazon Web Services (AWS)

* Moore's law says that the number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years [Intel co-founder Gordon E. Moore described the trend in his 1965 paper and predicted that it will last for at least 10 years]

Big Data

- Companies, organizations, scientists have data that is too big, too fast, and too complex to be managed without changing tools and processes.

- Relational algebra and SQL are easy to parallelize and parallel DBMSs have already been studied in the 80's!

Data Analytics Companies

As a result, we are seeing an explosion of and a huge success of db analytics companies

- Greenplum founded in 2003 acquired by EMC in 2010; A parallel shared-nothing DBMS (this lecture)
- Vertica founded in 2005 and acquired by HP in 2011; A parallel, column-store shared-nothing DBMS (see 444 for discussion of column-stores)
- DATAllegro founded in 2003 acquired by Microsoft in 2008; A parallel, shared-nothing DBMS
- Aster Data Systems founded in 2005 acquired by Teradata in 2011; A parallel, shared-nothing, MapReduce-based data processing system (next lecture). SQL on top of MapReduce
- Netezza founded in 2000 and acquired by IBM in 2010. A parallel, shared-nothing DBMS.

Great time to be in the data management, data mining/statistics, or machine learning!

Two Approaches to Parallel Data Processing

- Parallel databases, developed starting with the 80s (this lecture)
  - For both OLTP and Decision Support Queries

- Map/reduce, first developed by google, published in 2004 (next lecture)
  - Only for Decision Support Queries

Today we see convergence of the two approaches (Greenplum,Tenzing SQL)

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

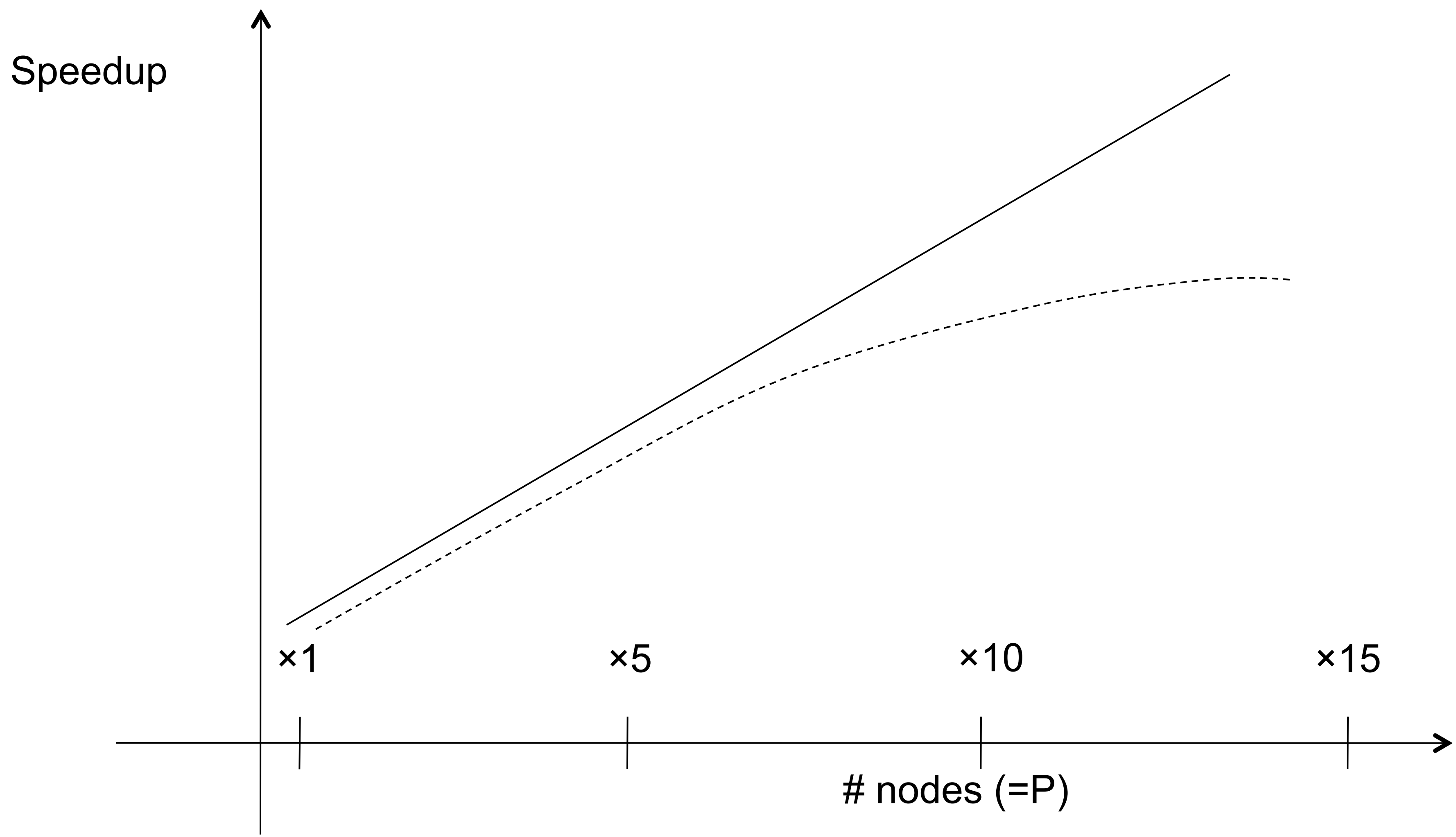
Performance Metrics for Parallel DBMSs

P = the number of nodes (processors, computers)

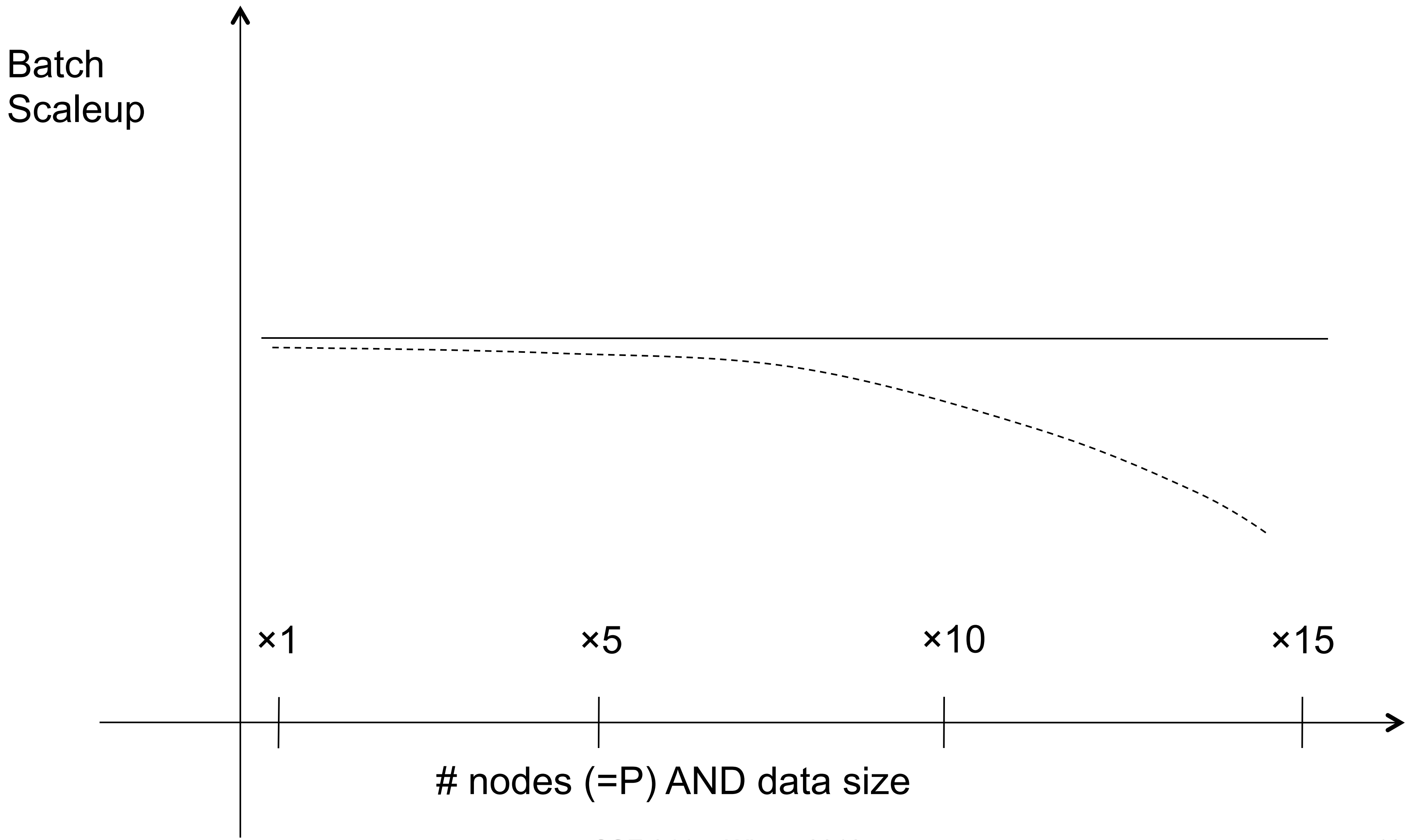
- Speedup:
  - More nodes, same data → higher speed
- Scaleup:
  - More nodes, more data → same speed

- OLTP: “Speed” = transactions per second (TPS)
- Decision Support: “Speed” = query time

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 nodes
- Interference
  - Contention for resources between nodes
- Skew
  - Slowest node becomes the bottleneck

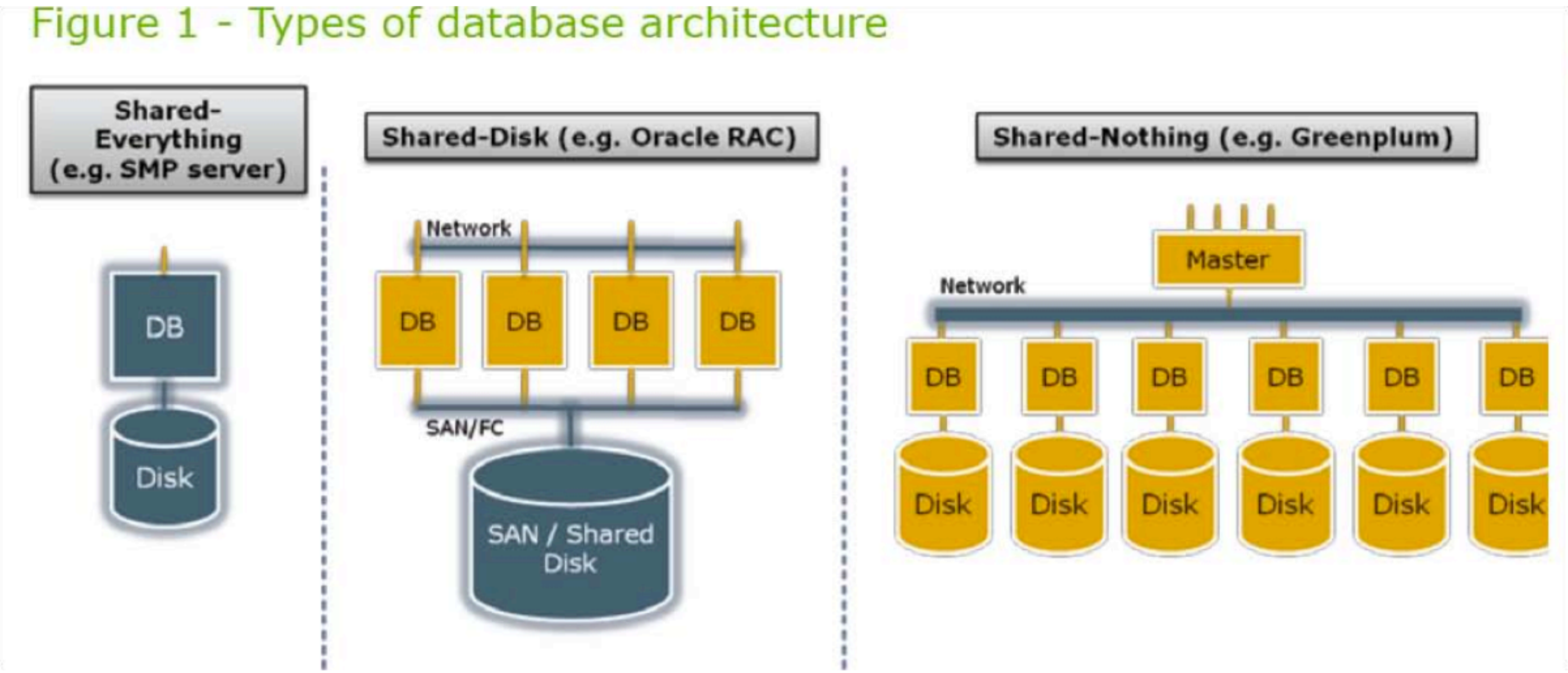
Architectures for Parallel Databases

- Shared memory

- Shared disk

- Shared nothing

Architectures for Parallel Databases



From: Greenplum Database Whitepaper

SAN = "Storage Area Network"

Shared Memory

- Nodes share both RAM and disk
- Dozens to hundreds of processors

Example: SQL Server runs on a single machine and can leverage many threads to get a query to run faster (see query plans on IISQLSRV)

- Easy to use and program
- But very expensive to scale: last remaining cash cows in the hardware industry

Shared Disk

- All nodes access the same disks
- Found in the largest "single-box" (non-cluster) multiprocessors

Oracle dominates this class of systems.

Characteristics:

- Also hard to scale past a certain point: existing deployments typically have fewer than 10 machines

Shared Nothing

- Cluster of machines on high-speed network
- Called "clusters" or "blade servers”
- Each machine has its own memory and disk: lowest contention.

NOTE: Because all machines today have many cores and many disks, then shared-nothing systems typically run many "nodes” on a single physical machine.

Characteristics:
- Today, this is the most scalable architecture.
- Most difficult to administer and tune.

We discuss only Shared Nothing in class

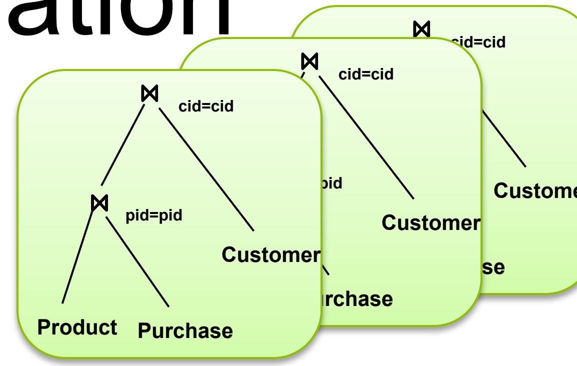
In Class

- You have a parallel machine. Now what?

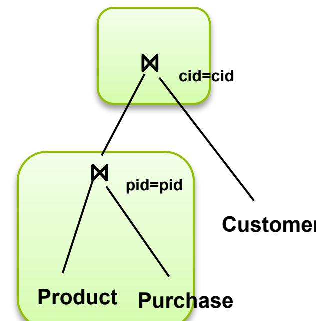
- How do you speed up your database system?

Approaches to Parallel Query Evaluation

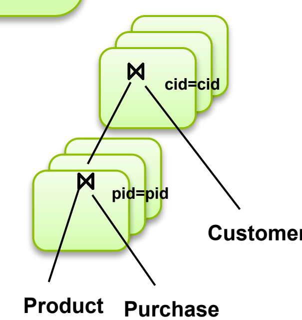
- Inter-query parallelism
  - Each query runs on one processor
  - Only for OLTP queries



- Inter-operator parallelism
  - A query runs on multiple processors
  - An operator runs on one processor
  - For both OLTP and Decision Support



- Intra-operator parallelism
  - An operator runs on multiple processors
  - For both OLTP and 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

- Have a large table R(<u>K</u>, A, B, C)
- Need to partition on a shared-nothing architecture into P chunks $R_1, \ldots, R_P$, stored at the P nodes

- Block Partition: $\mathrm{size}(R_1) \approx \ldots \approx \mathrm{size}(R_P)$

- Hash partitioned on attribute A:
  - Tuple t goes to chunk i, where $i = h(t.A) \bmod P + 1$

- 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 $v_{i-1} < t.A < v_i$

Parallel GroupBy

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

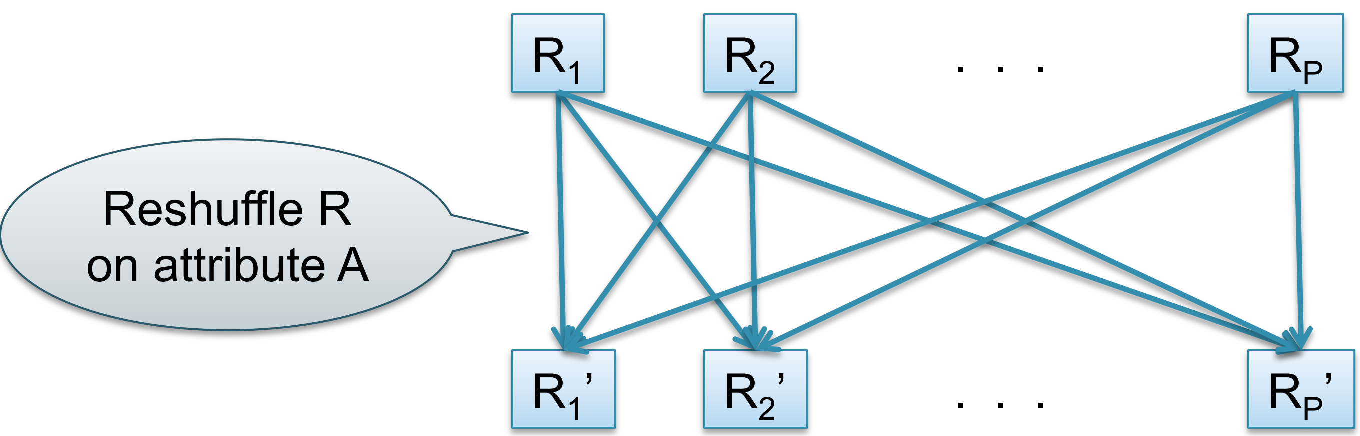
- $\gamma_{A,sum(C)}(R)$

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

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:
  - $\text{size}(R_1) \approx \ldots \approx \text{size}(R_P) \approx \quad \text{size}(R) / P$
  - Linear speedup, constant scaleup

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

Uniform Data v.s. Skewed Data

- Let R(<u>K</u>,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
  - On the attribute A

Uniform

Assuming uniform hash function

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 $R(A,B) \bowtie S(B,C)$

R_1, S_1 $\quad$ R_2, S_2 $\quad$ . . . $\quad$ R_P, S_P

Parallel Join

- In class: compute $R(A,B) \bowtie S(B,C)$