Introduction to Data Management CSE 344

Lecture 24: Parallel Databases

Announcements

- HW7 due on Wednesday
- HW8 will be posted soon
  - Will take more hours than other HWs (complex setup, queries run for many hours)
  - No late days
  - Plan ahead!
  - Hold on to the email that Daseul sent yesterday
- Next four lectures: parallel databases
  - Traditional, MapReduce+PigLatin

Parallel Computation Today

Two Major Forces Pushing towards Parallel Computing:

- Change in Moore's law
- Cloud computing

Parallel Computation Today

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

* 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]

Parallel Computation Today

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

Jeff Dean, SOCC'2010:

Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K w/cheap compression algorithm	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Memory access

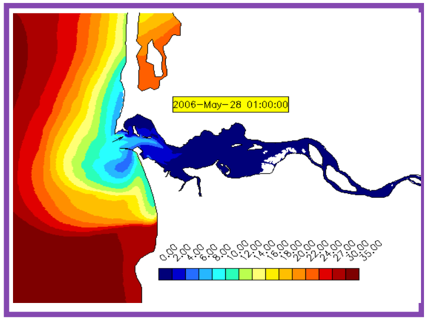
Local access is significantly faster than communication

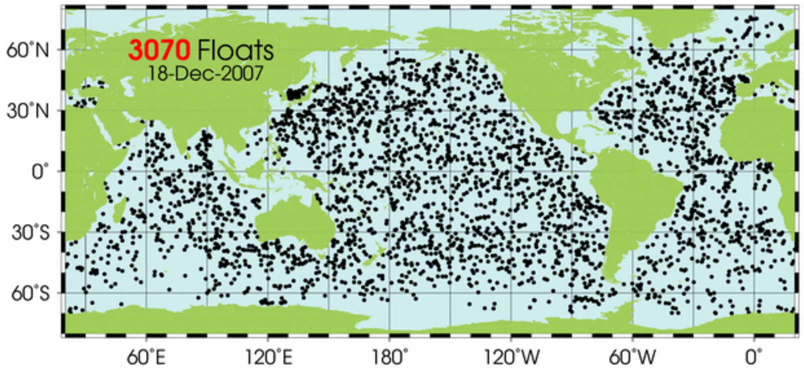
Communication

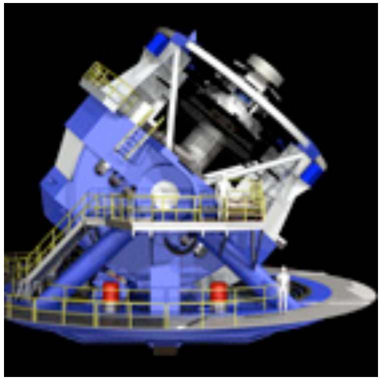
Science is Facing a Data Deluge!

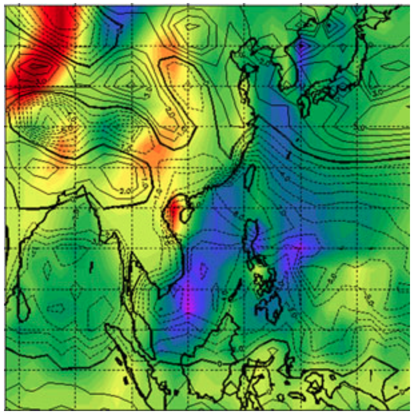
- Astronomy: Large Synoptic Survey Telescope LSST: 30TB/night (high-resolution, high-frequency sky surveys)
- Physics: Large Hadron Collider 25PB/year
- Biology: lab automation, high-throughput sequencing
- Oceanography: high-resolution models, cheap sensors, satellites
- Medicine: ubiquitous digital records, MRI, ultrasound

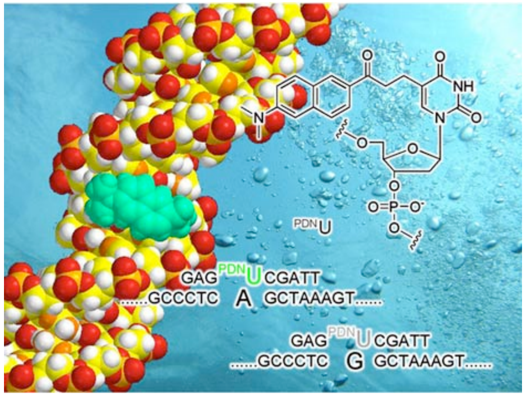
Science is Facing a Data Deluge!









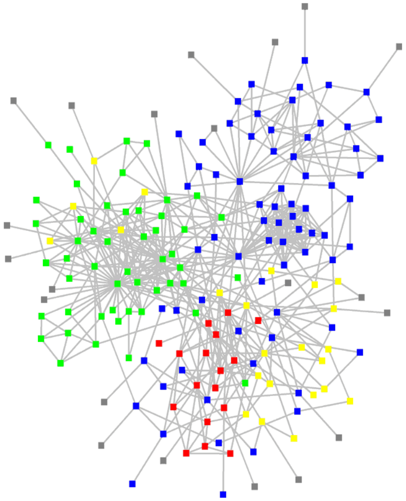


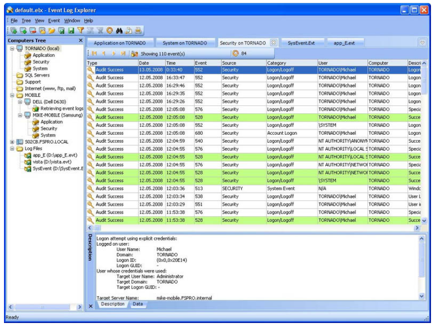
Industry is Facing a Data Deluge!

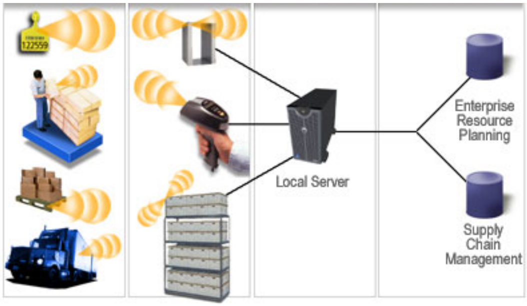
Clickstreams, search logs, network logs, social networking data, RFID data, etc.

- Facebook:
  - 15PB of data in 2010
  - 60TB of new data every day
- Google:
  - In May 2010 processed 946PB of data using MapReduce
- Twitter, Google, Microsoft, Amazon, Walmart, etc.

Industry is Facing a Data Deluge!







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 Kinds to Parallel Data Processing

- Parallel databases, developed starting with the 80s (this lecture)
  - OLTP (Online Transaction Processing)
  - OLAP (Online Analytic Processing, or Decision Support)
- MapReduce, first developed by Google, published in 2004 (next lecture)
  - Only for Decision Support Queries

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

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

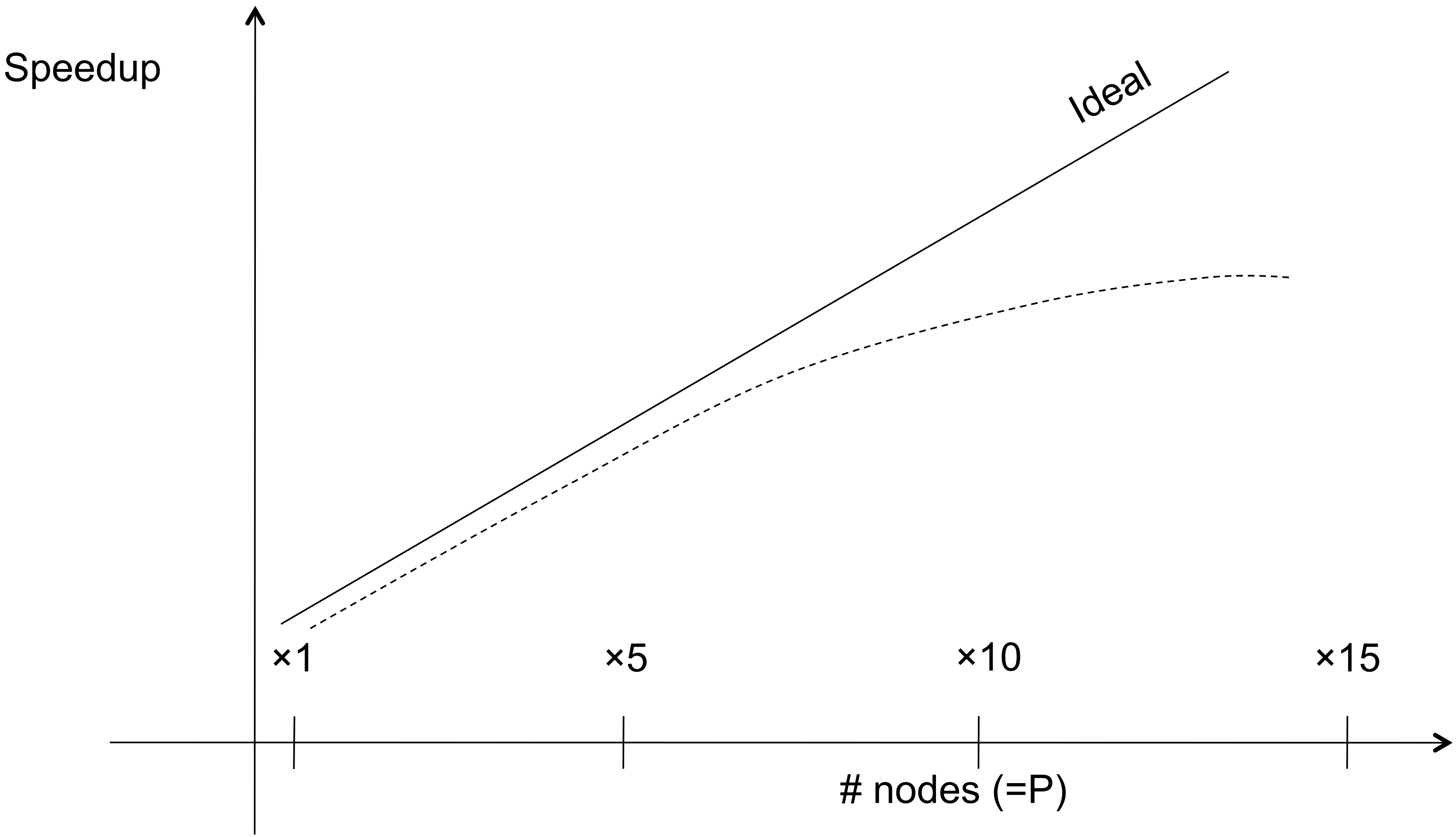
Performance Metrics for Parallel DBMSs

P = the number of nodes (processors, computers)

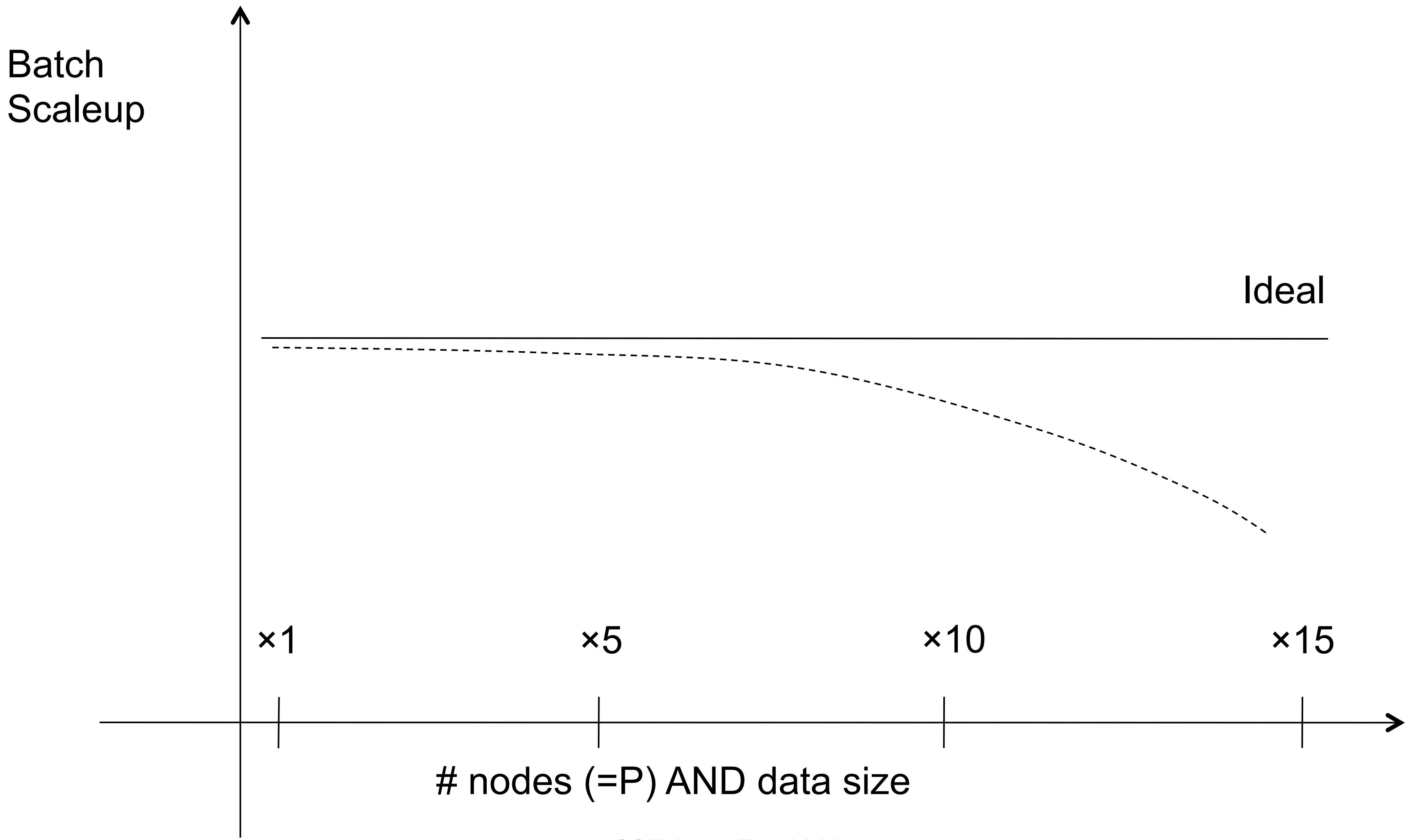
- Speedup:
  - More nodes, same data → higher speed
- Scaleup:
  - More nodes, more data → same speed

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

Linear v.s. Non-linear Speedup



Linear v.s. Non-linear Scaleup



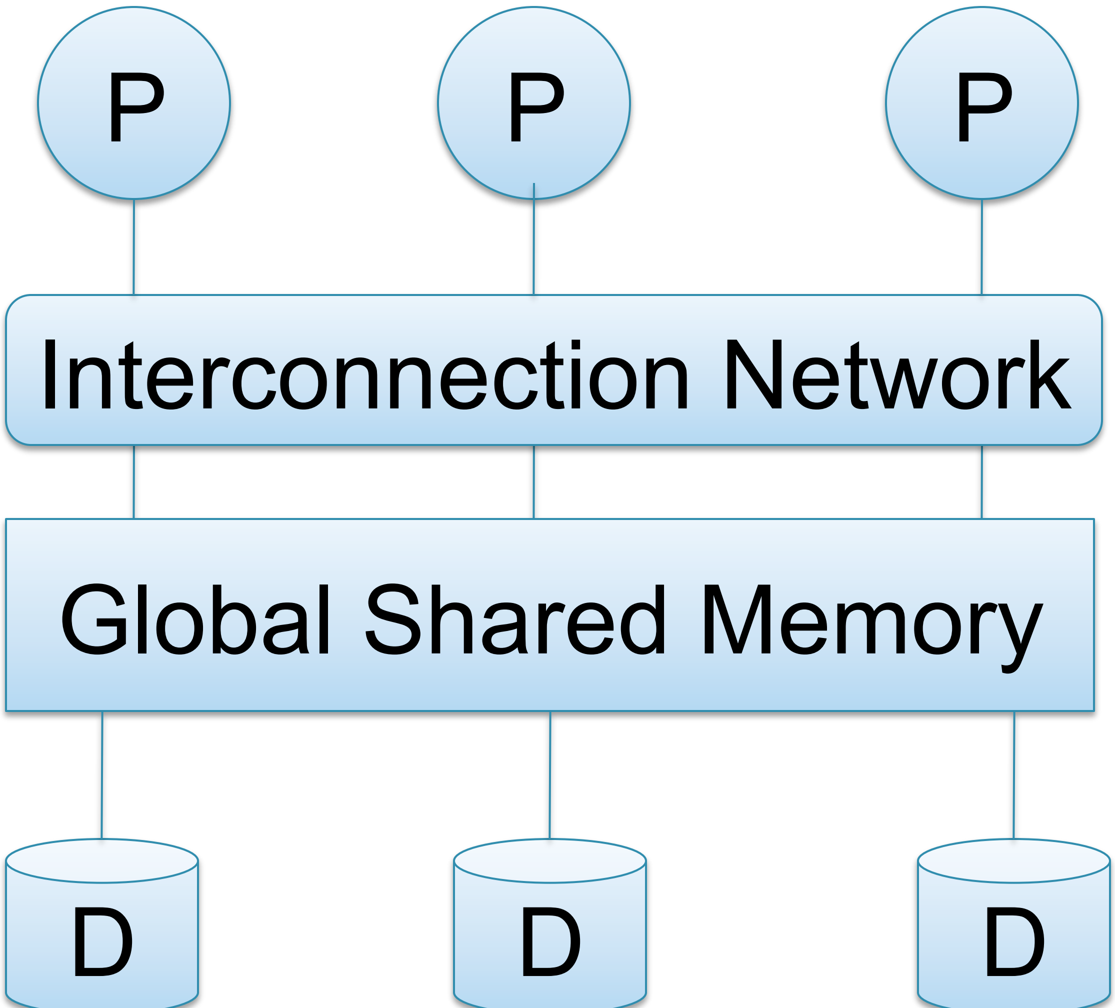
Challenges to Linear Speedup and Scaleup

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

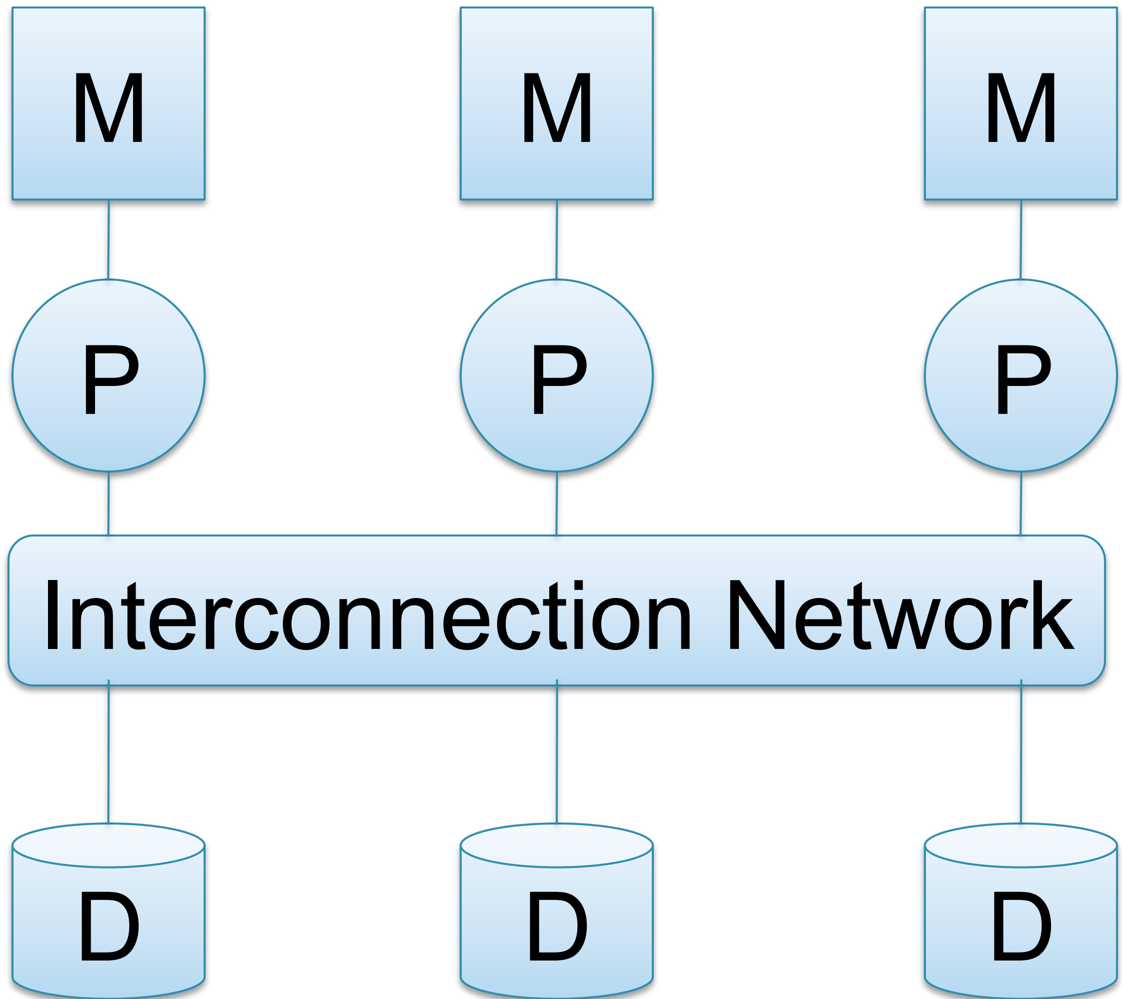
Architectures for Parallel Databases

- Shared memory
- Shared disk
- Shared nothing

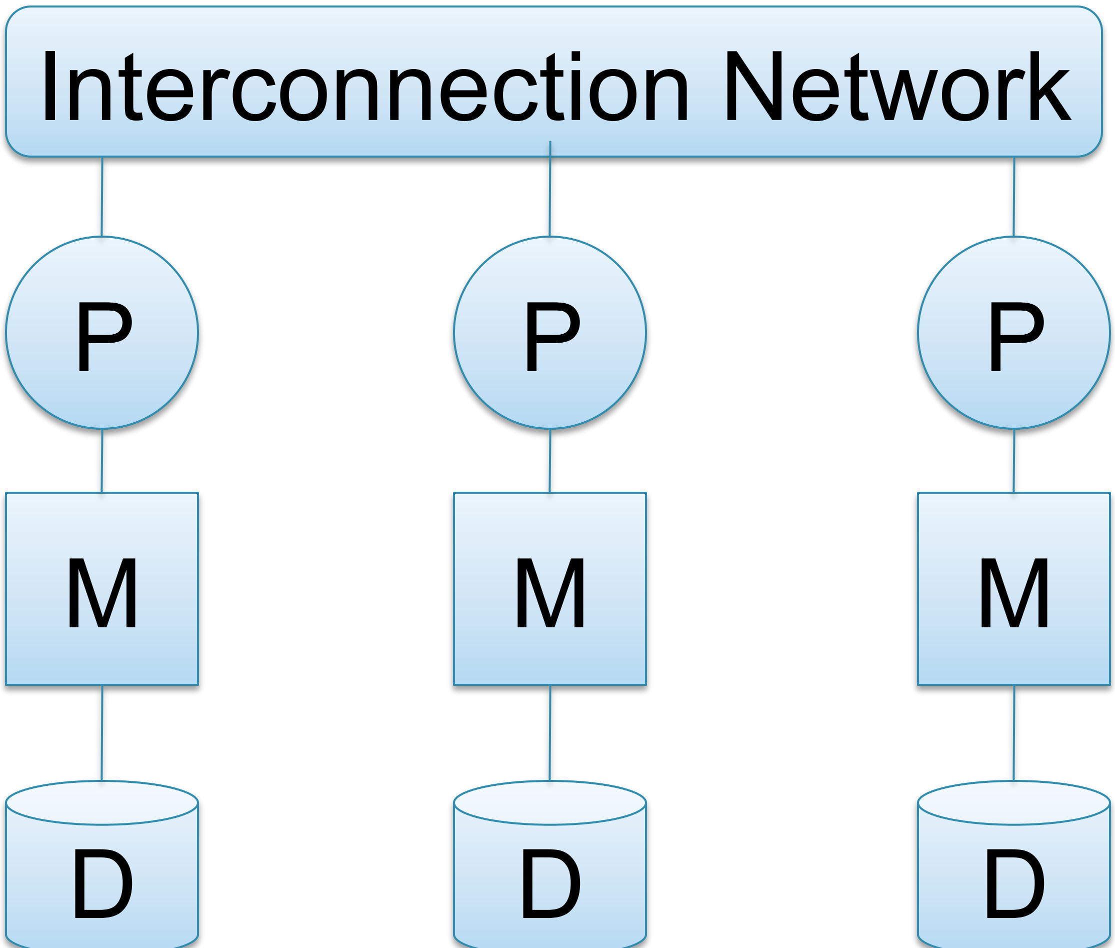
Shared Memory



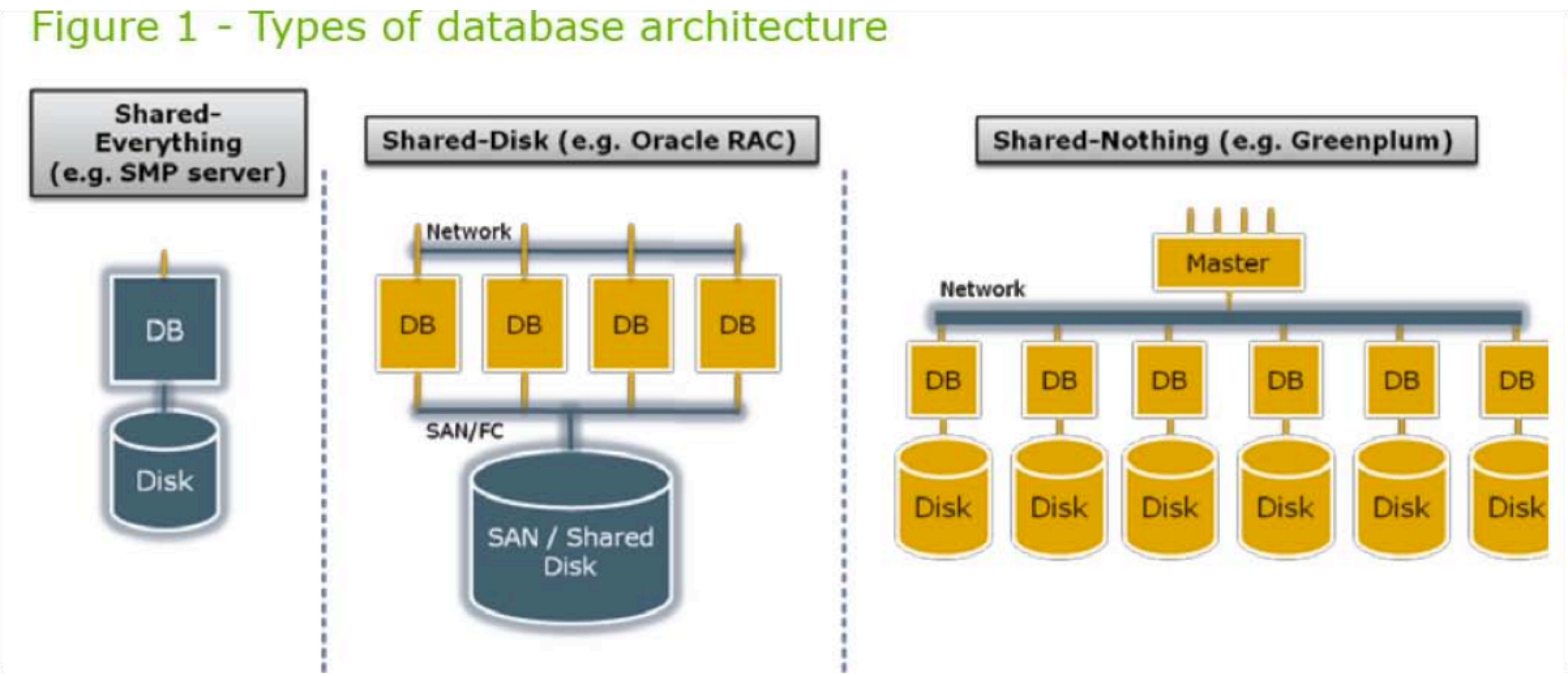
Shared Disk



Shared Nothing



A Professional Picture…



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)

- 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:

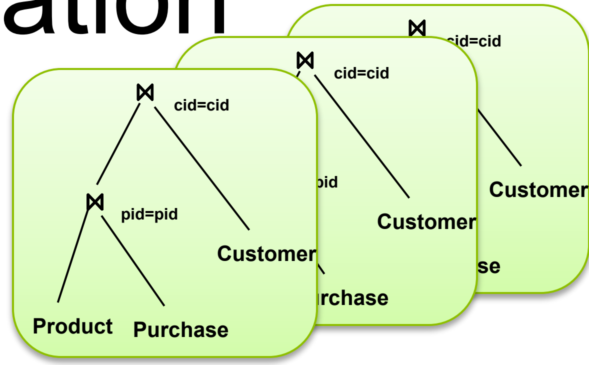
- Today, this is the most scalable architecture.
- Most difficult to administer and tune.

We discuss only Shared Nothing in class

In Class

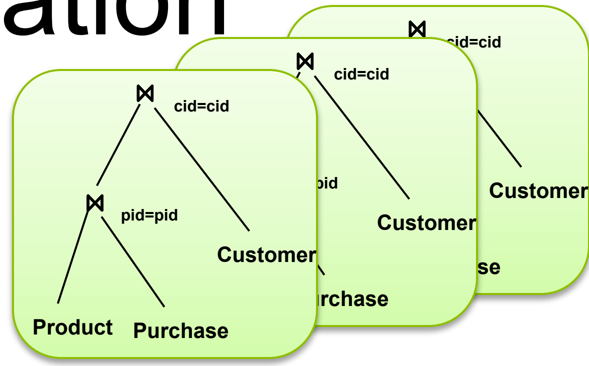
- You have a parallel machine. Now what?
- How do you speed up your DBMS?

Approaches to Parallel Query Evaluation

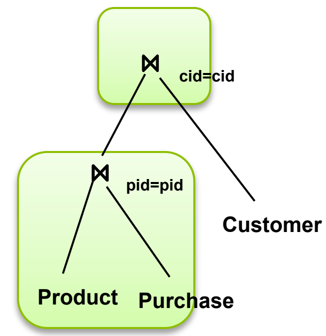


- Inter-query parallelism
  - Transaction per node
  - OLTP

Approaches to Parallel Query Evaluation



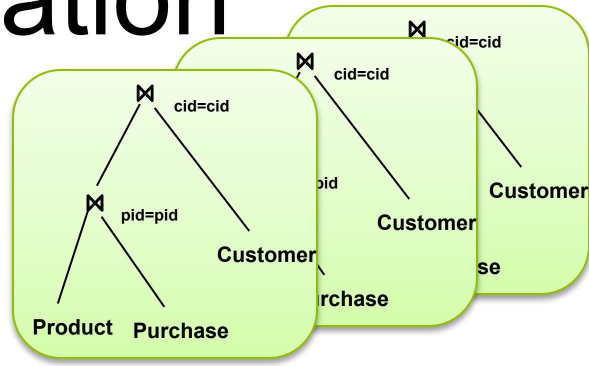
- Inter-query parallelism
  - Transaction per node
  - OLTP



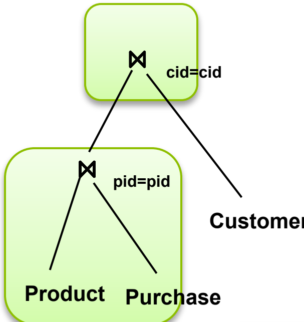
- Inter-operator parallelism
  - Operator per node
  - Both OLTP and Decision Support

Approaches to Parallel Query Evaluation

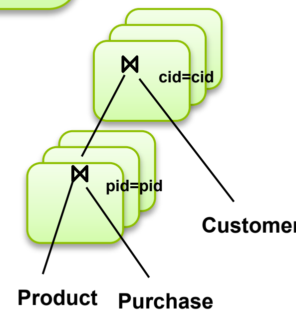
- Inter-query parallelism
  - Transaction per node
  - OLTP



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

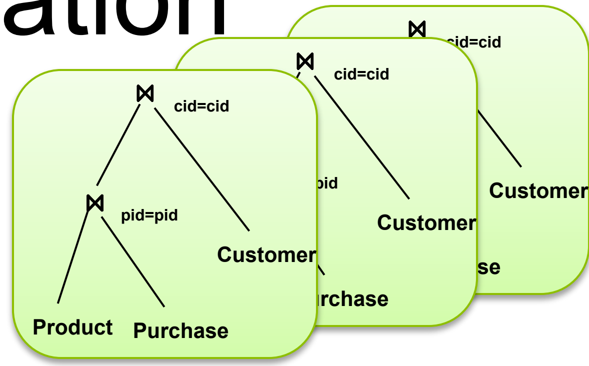


- Intra-operator parallelism
  - Operator on multiple nodes
  - Decision Support

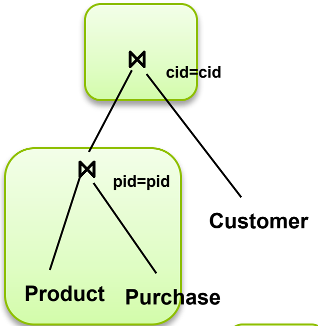


Approaches to Parallel Query Evaluation

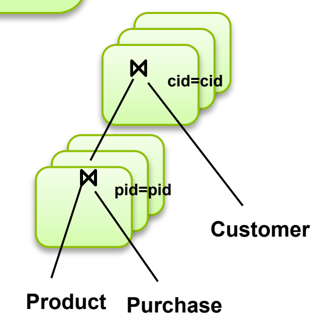
- Inter-query parallelism
  - Transaction per node
  - OLTP



- 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

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

- Block Partition: size(R_1)≈ … ≈ size(R_P)

- Hash partitioned on attribute A:
  - Tuple t goes to chunk i, where i = h(t.A) mod 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

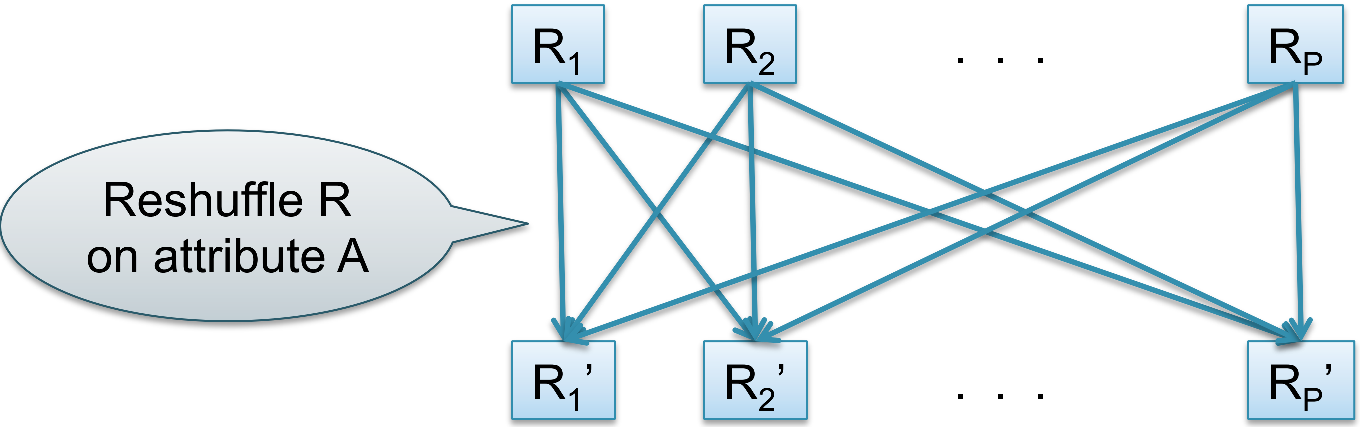
R(<u>K</u>,A,B,C), discuss in class how to compute these GroupBy's, for each of the partitions

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

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:
  - $size(R_1) \approx \ldots \approx size(R_P) \approx \quad size(R) / P$
  - 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(<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.

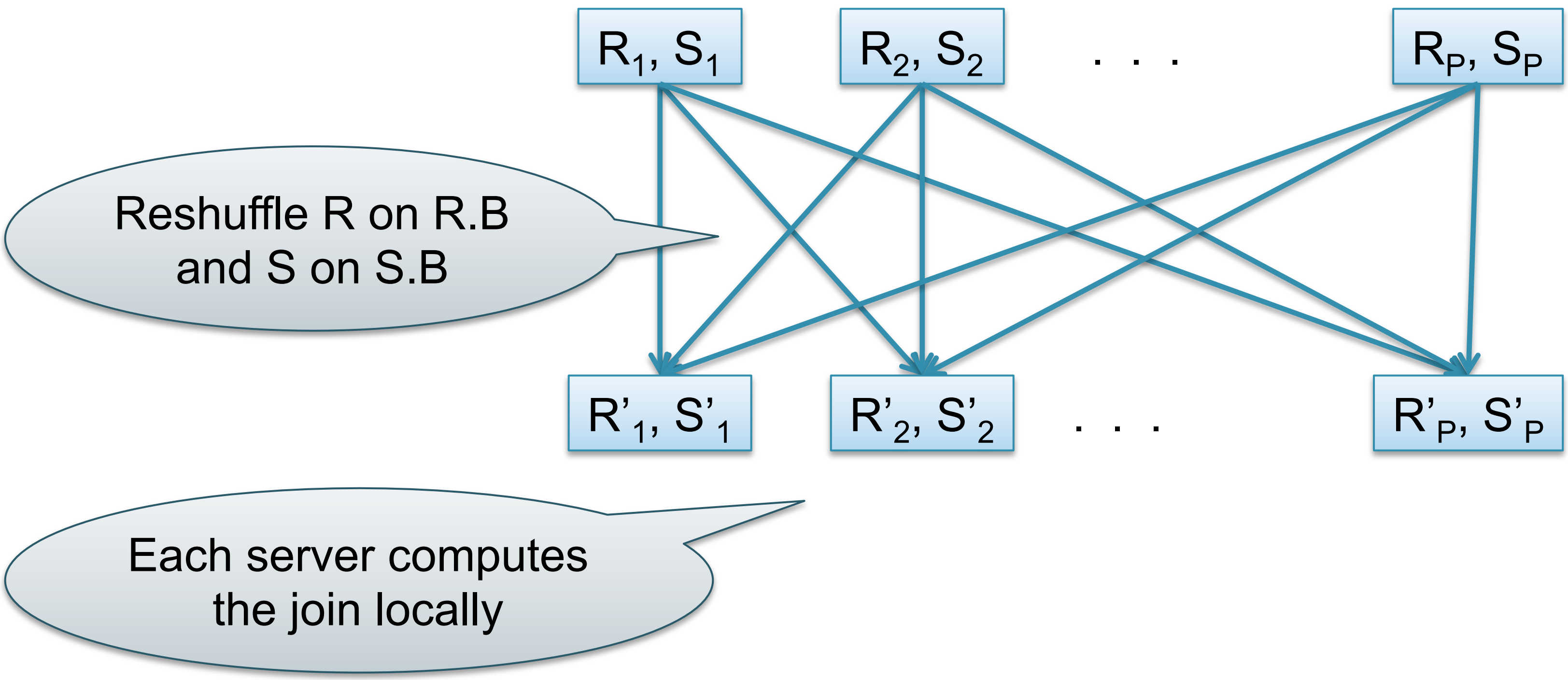
Parallel Join

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



Amazon Warning

- “We **HIGHLY** recommend you remind students to turn off any instances after each class/session – as this can quickly diminish the credits and start charging the card on file. **You are responsible for the overages.”**
- “AWS customers can now use **billing alerts** to help monitor the charges on their AWS bill. You can get started today by visiting your Account Activity page to enable monitoring of your charges. Then, you can set up a billing alert by simply specifying a bill threshold and an e-mail address to be notified as soon as your estimated charges reach the threshold.”