Introduction to Data Management
CSE 344

Lecture 23:

Parallel Databases

Announcements

- HW7 due tomorrow
- HW8 is posted (1 late day, setup in tomorrow's section)
- Review session schedule (problem solving)

Topic	Date	Venue/Time	Who
RC/RA/Datalog	3/6 (Th) Sections	Sections	Yi-Shu
BCNF/ ERD	3/10 (M)	CSE 303, 4:30-5:30pm	Vaspol
Transaction	3/13 (Th) Sections	Sections	Yi-Shu
Parallel DB/MR	3/14 (F) Class	Class	Sudeepa

- Additional review session: March 15 (Sat), 2-4 pm, CSE 303, Sudeepa, office hour format (review of lecture/ assignments/old exams together if you have questions)

HW8

- Will take more hours than other HWs, start early
  - complex setup, queries run for many hours
- MapReduce (Hadoop) w/ declarative language (Pig)
- Cluster will run in Amazon's cloud (AWS)
  - Give your credit card
  - Click, click, click… and you have a MapReduce cluster
- We will analyze a real 0.5TB graph
- Processing the entire data takes hours
  - Problems #1,#2,#3: queries on a subset only
  - Problem #4: entire data

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.”

Outline

- Today: Query Processing in Parallel DBs
- Next Lecture: Parallel Data Processing at Massive Scale (MapReduce)
  - Reading assignment:
    Chapter 2 (Sections 1,2,3 only) of Mining of Massive Datasets, by Rajaraman and Ullman
    http://i.stanford.edu/~ullman/mmds.html

What we did in last lecture

- Why parallel processing?
- What are the possible architectures for a parallel database system?
- What are speedup and scaleup?

Basic Query Processing: Quick Review in Class

Basic query processing on one node.

Given relations R(A,B) and S(B, C), no indexes, how do we compute:

- Selection: $\sigma_{A=123}(R)$

- Group-by: $\gamma_{A,sum(B)}(R)$

- Join: $R \bowtie S$

Basic Query Processing: Quick Review in Class

Basic query processing on one node.

Given relations R(A,B) and S(B, C), no indexes, how do we compute:

- Selection: $\sigma_{A=123}(R)$
  - Scan file R, select records with A=123
- Group-by: $\gamma_{A,sum(B)}(R)$
  - Scan file R, insert into a hash table using attr. A as key
  - When a new key is equal to an existing one, add B to the value
- Join: $R \bowtie S$
  - Scan file S, insert into a hash table using attr. B as key
  - Scan file R, probe the hash table using attr. B

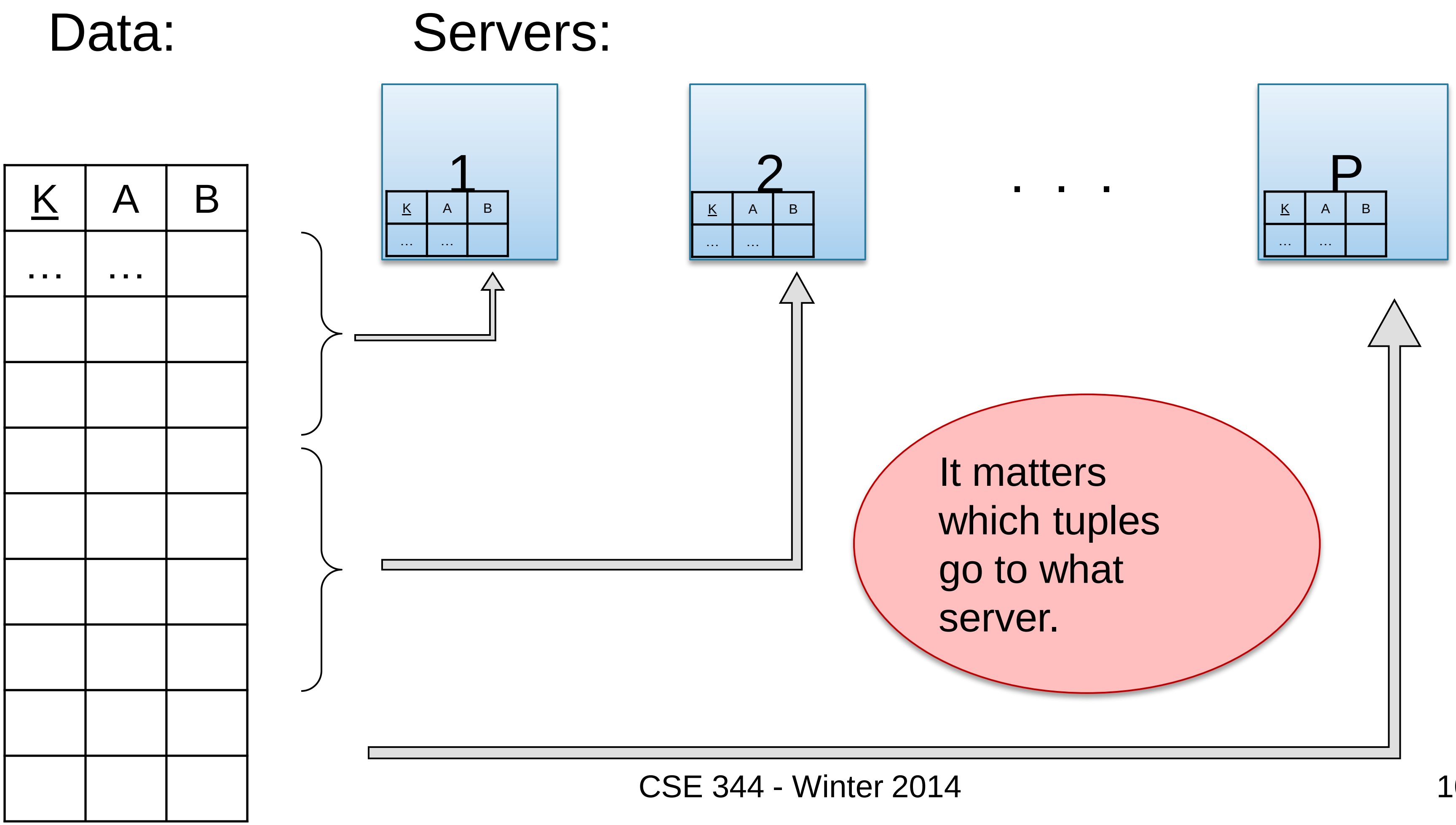
Parallel Query Processing

How do we compute these operations on a shared-nothing parallel db?

- Selection: $\sigma_{A=123}(R)$ (that's easy, won't discuss…)
- Group-by: $\gamma_{A,sum(B)}(R)$
- Join: $R \bowtie S$

Before we answer that: how do we store R (and S) on a shared-nothing parallel db?

Horizontal Data Partitioning



Horizontal Data Partitioning

- Block Partition:
  - Partition tuples arbitrarily s.t. $\text{size}(R_1) \approx \ldots \approx \text{size}(R_P)$
- 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

Data: R(K,A,B,C)

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

Discuss in class how to compute in each case:

- R is hash-partitioned on A
- R is block-partitioned
- R is hash-partitioned on K (key)

Q. Which one can leverage locality of tuples (less communication)?

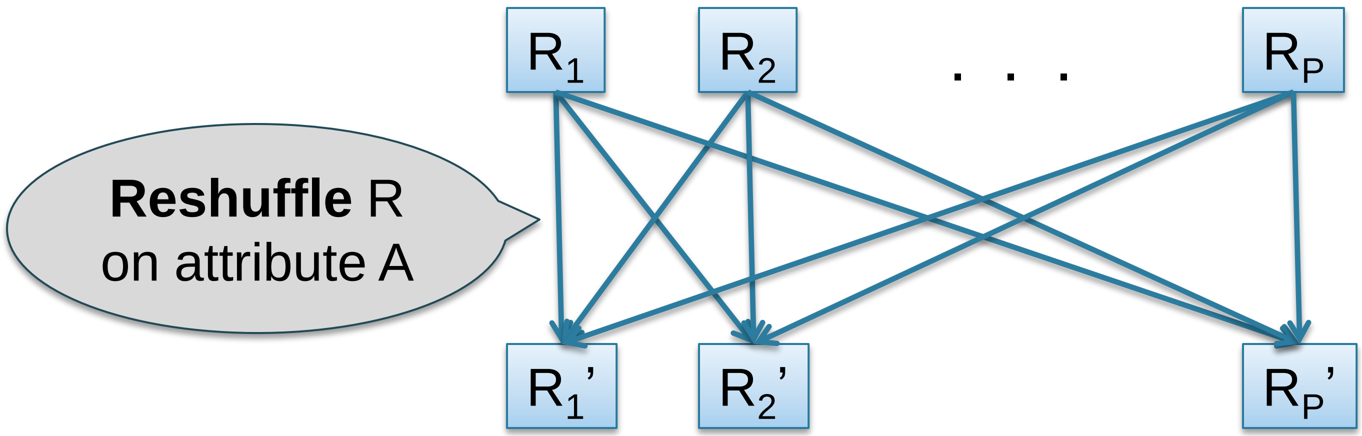
What will the others do?

Parallel GroupBy

Data: R(K,A,B,C)

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

- R is block-partitioned or hash-partitioned on K



Parallel Join

- Data: R(<u>K1</u>,A, B), S(<u>K2</u>, B, C)
- Query: R(<u>K1</u>,A,B) $\bowtie_{B = B}$ S(<u>K2</u>,B,C)

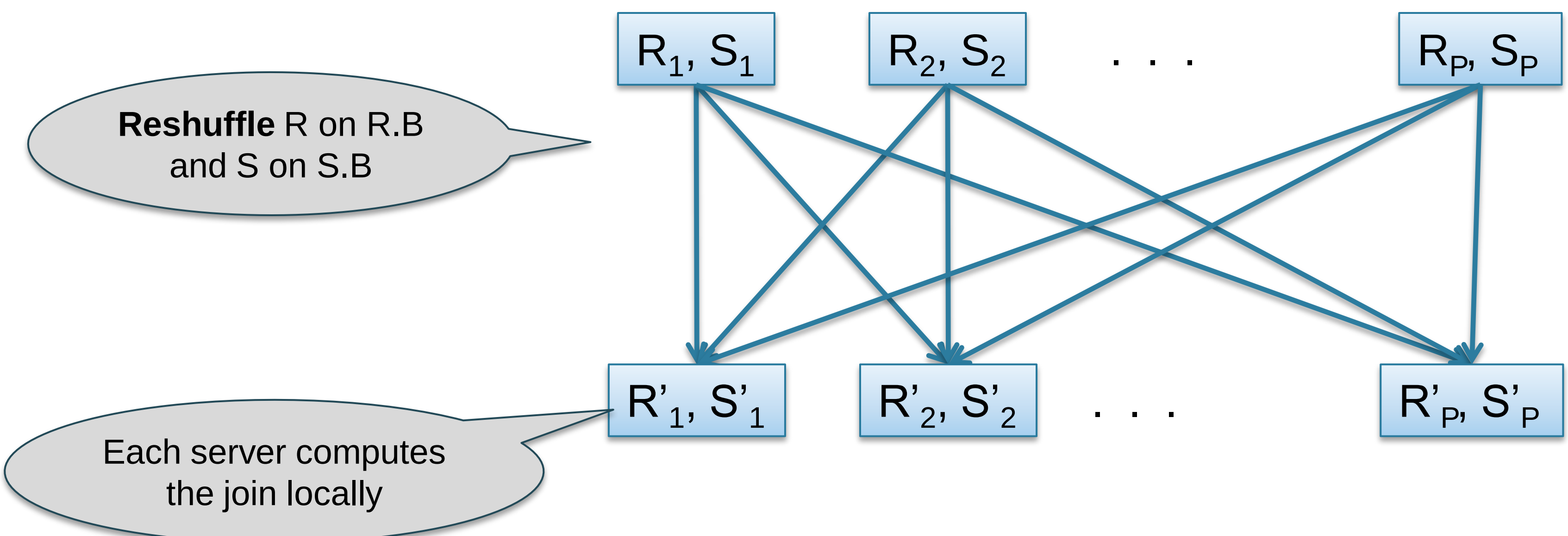
Initially, both R and S are horizontally partitioned on K1 and K2

R_1, S_1 R_2, S_2 R_P, S_P

Parallel Join

- **Data**: R(K1,A, B), S(K2, B, C)
- **Query**: R(K1,A,B) ⋈ S(K2,B,C)

Initially, both R and S are horizontally partitioned on K1 and K2



Speedup and Scaleup

- Consider:
  - Query: $\gamma_{A,sum(C)}(R)$
  - Runtime: dominated by reading chunks from disk
- If we double the number of nodes P, what is the new running time?

- If we double both P and the size of R, what is the new running time?

Speedup and Scaleup

- Consider:
  - Query: $\gamma_{A,sum(C)}(R)$
  - Runtime: dominated by reading chunks from disk
- If we double the number of nodes P, what is the new running time?
  - Half (each server holds ½ as many chunks)
- If we double both P and the size of R, what is the new running time?
  - Same (each server holds the same # of chunks)

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
  - 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 DBMS

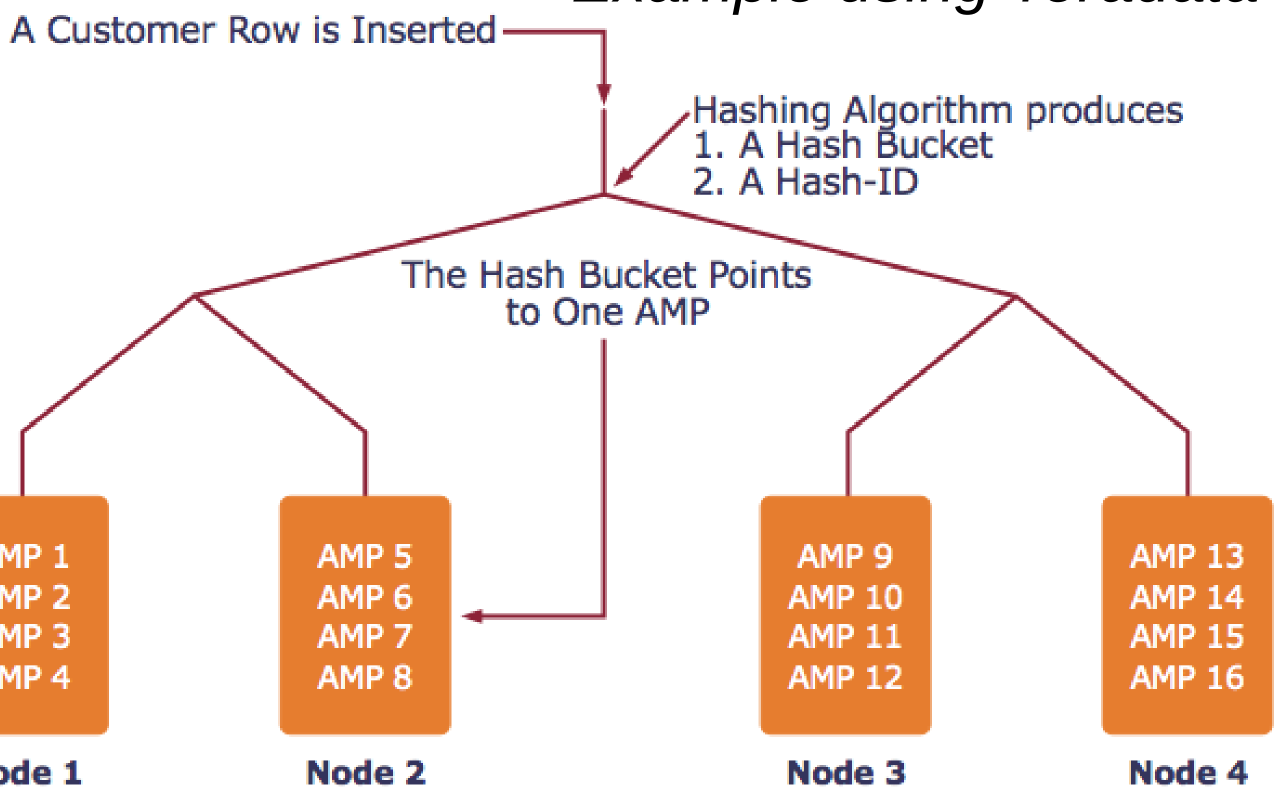
- Parallel query plan: tree of parallel operators
  Intra-operator parallelism
  - Data streams from one operator to the next
  - Typically all cluster nodes process all operators

- Can run multiple queries at the same time
  Inter-query parallelism
  - Queries will share the nodes in the cluster

- Notice that user does not need to know how his/her SQL query was processed

Loading Data into a Parallel DBMS

Example using Teradata System

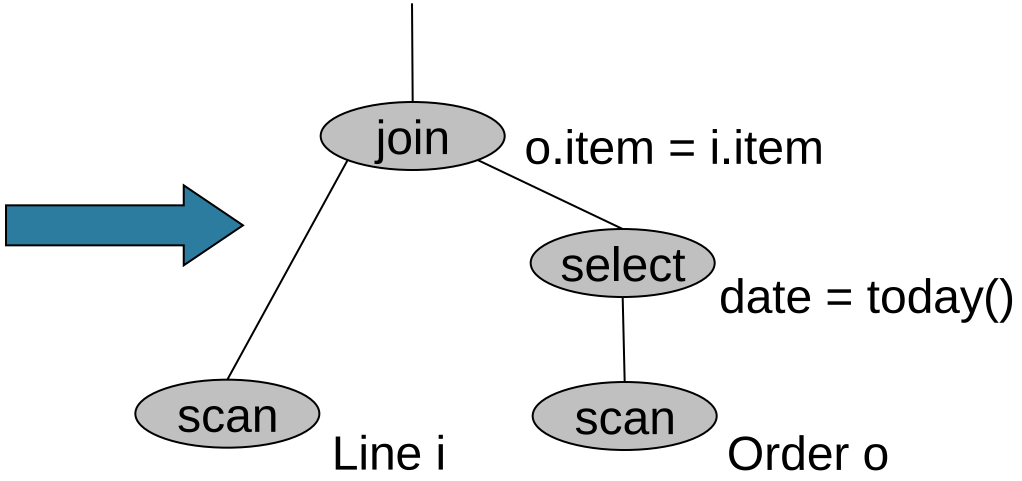


AMP = "Access Module Processor" = unit of parallelism

Example Parallel Query Execution

Find all orders from today, along with the items ordered

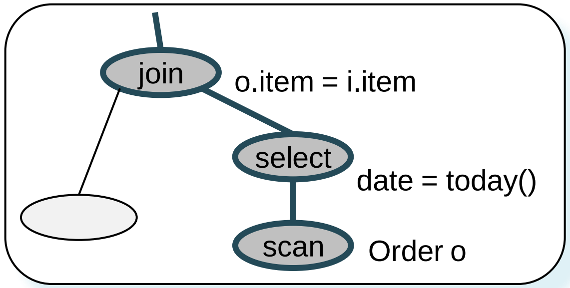
```
SELECT *
  FROM Order o, Line i
 WHERE o.item = i.item
   AND o.date = today()
```

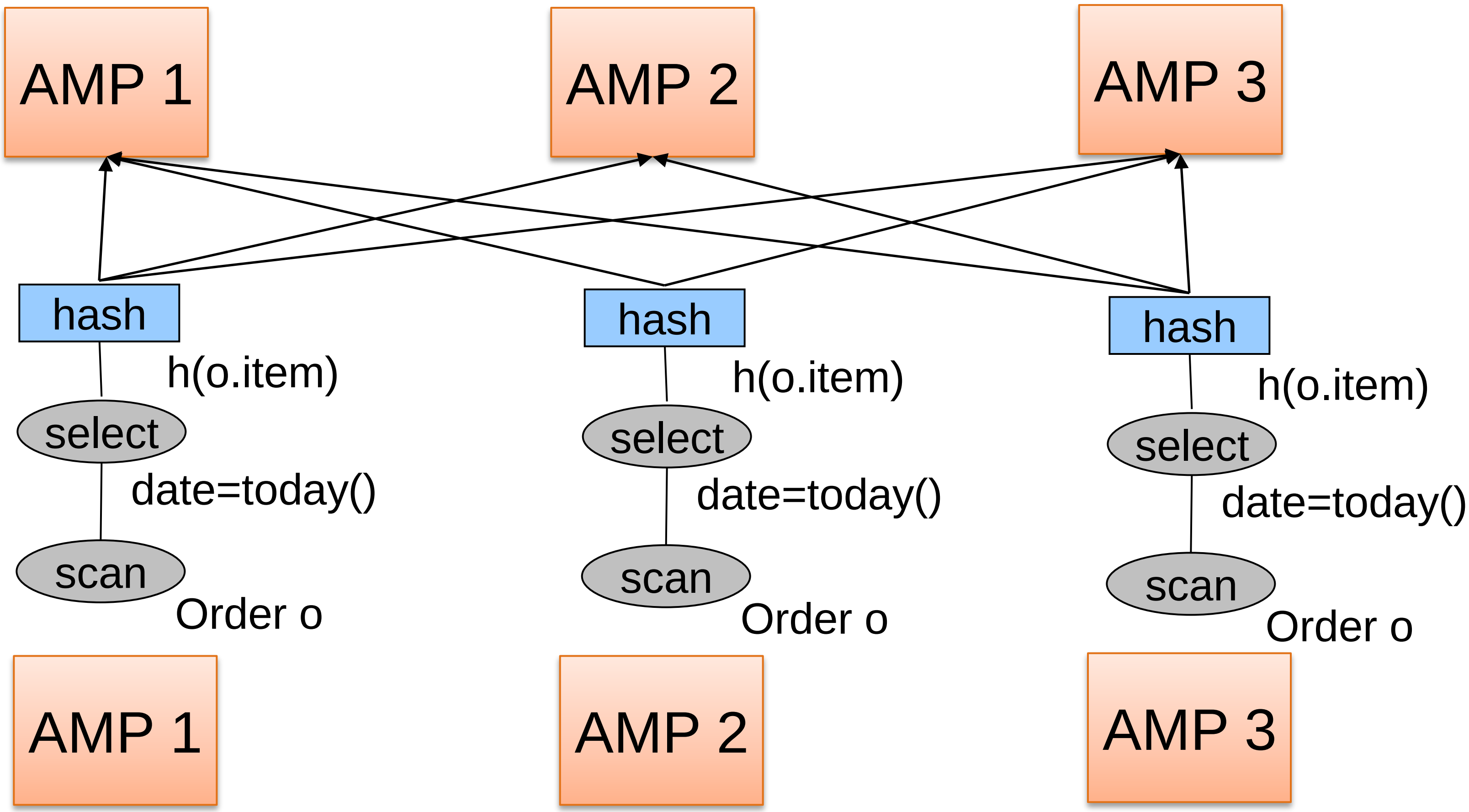


Order(oid, item, date), Line(item, …)

Example Parallel Query Execution

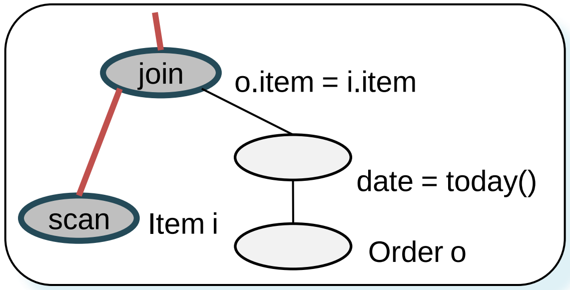
Scan, select, hash Order



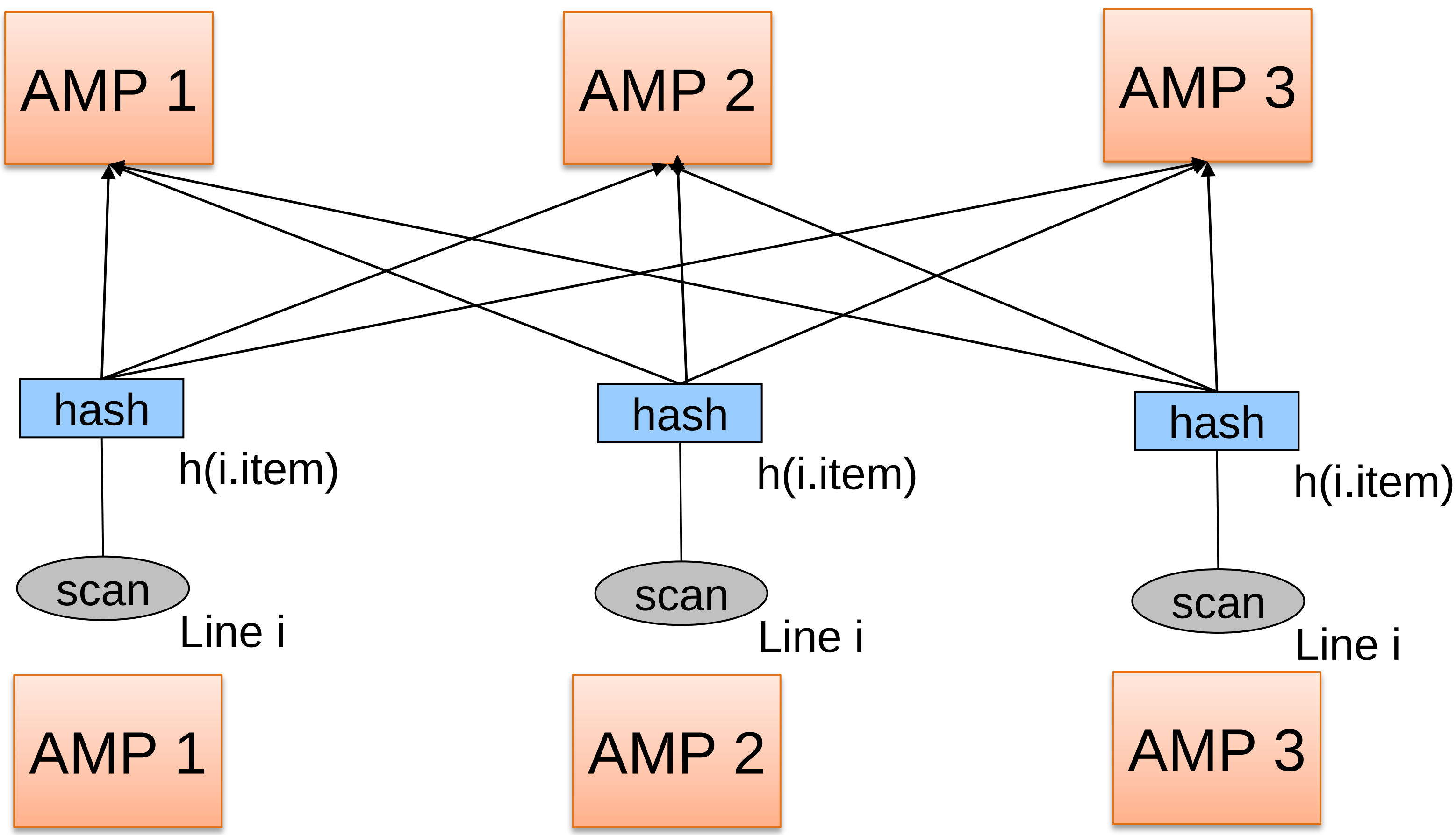


Order(oid, item, date), Line(item, …)

Example Parallel Query Execution

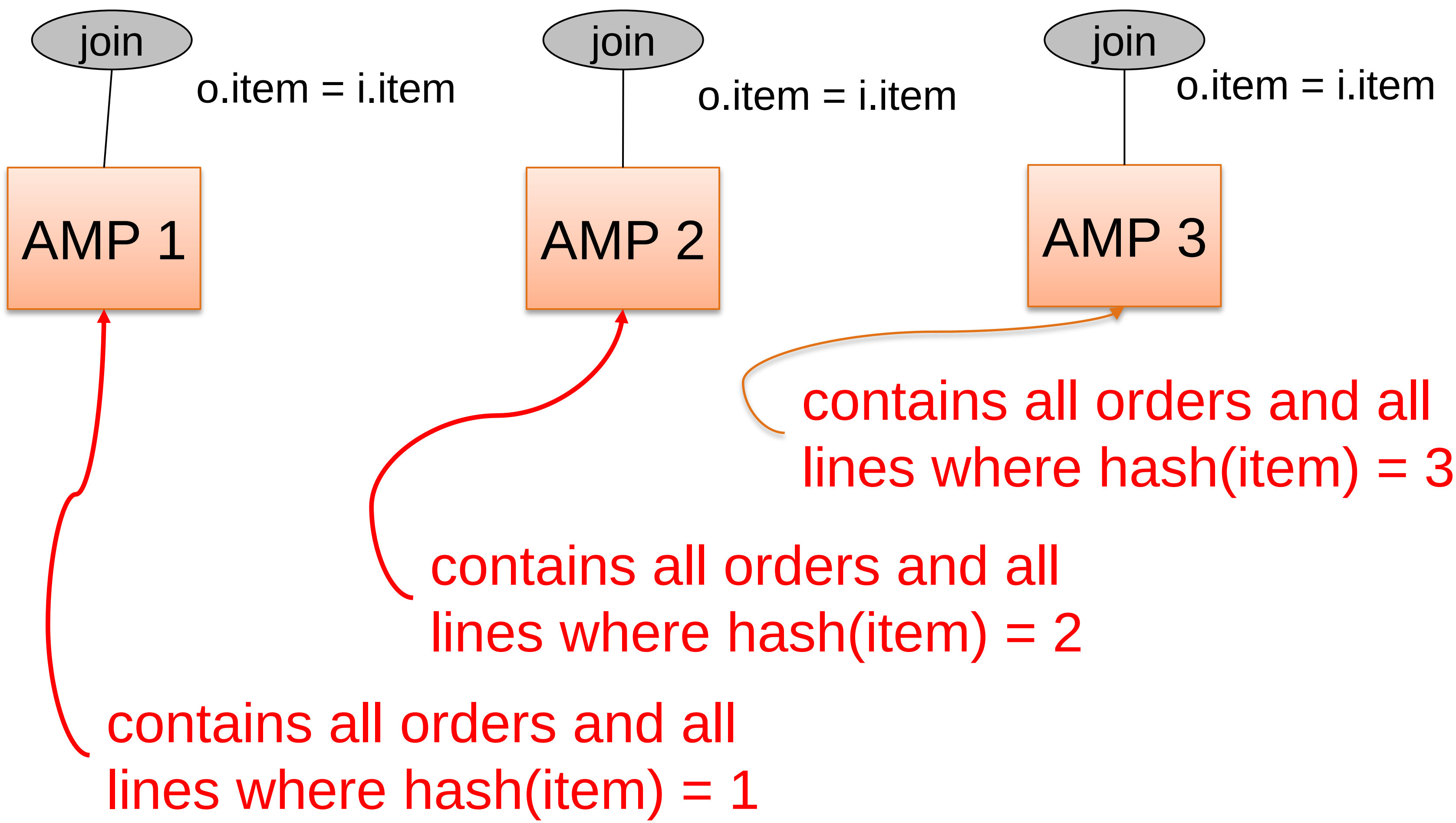


Scan, hash Line



Order(<u>oid</u>, item, date), Line(item, …)

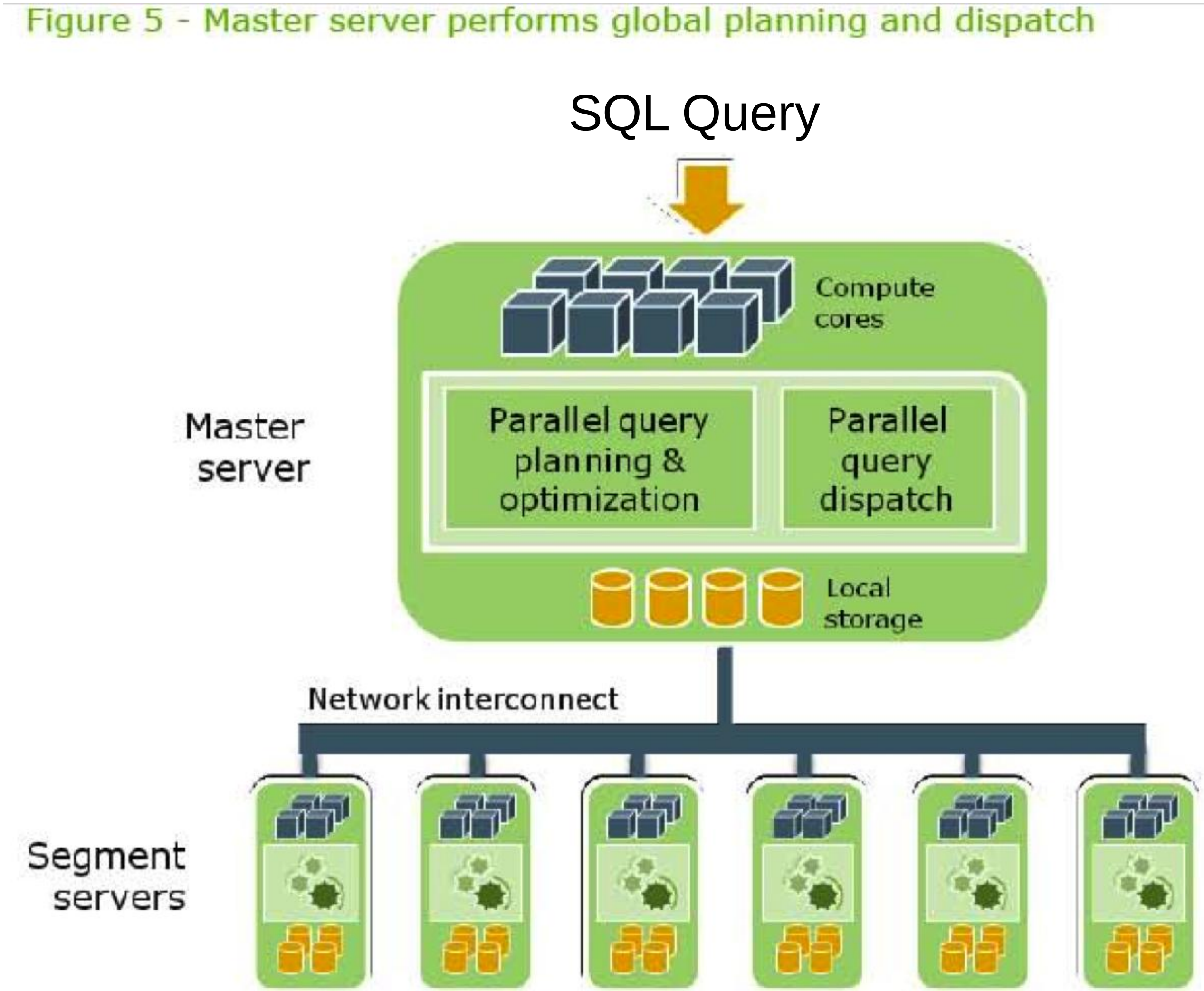
Example Parallel Query Execution



Parallel Dataflow Implementation

- Use relational operators unchanged
- Add a special *shuffle* operator
  - Handle data routing, buffering, and flow control
  - Inserted between consecutive operators in the query plan
  - Two components: ShuffleProducer and ShuffleConsumer
  - Producer pulls data from operator and sends to n consumers
    - Producer acts as driver for operators below it in query plan
  - Consumer buffers input data from n producers and makes it available to operator through getNext interface
- You will use this extensively in 444

Review: Parallel DBMS



From: Greenplum Database Whitepaper