CSE 344

JULY 23RD

PARALLEL DATABASES

ADMINISTRIVIA

- **HW5 due Wednesday**
- **Sign up for Amazon credits**
  - need for HW6. can take a while
- **Midterm on Friday**
  - Practice exam on web site
  - Videos from last 2 weeks all on web site
  - No need to memorize cost formulas but do need to understand them

WHY COMPUTE IN PARALLEL?

Multi-cores:

- Most processors have multiple cores
- This trend will likely increase in the future

Big data: too large to fit in main memory

- Disk has more space but is slow
- Distributed query processing on 100x-1000x servers
- Widely available now using cloud services

PERFORMANCE METRICS FOR PARALLEL DBS

Nodes = processors or computers

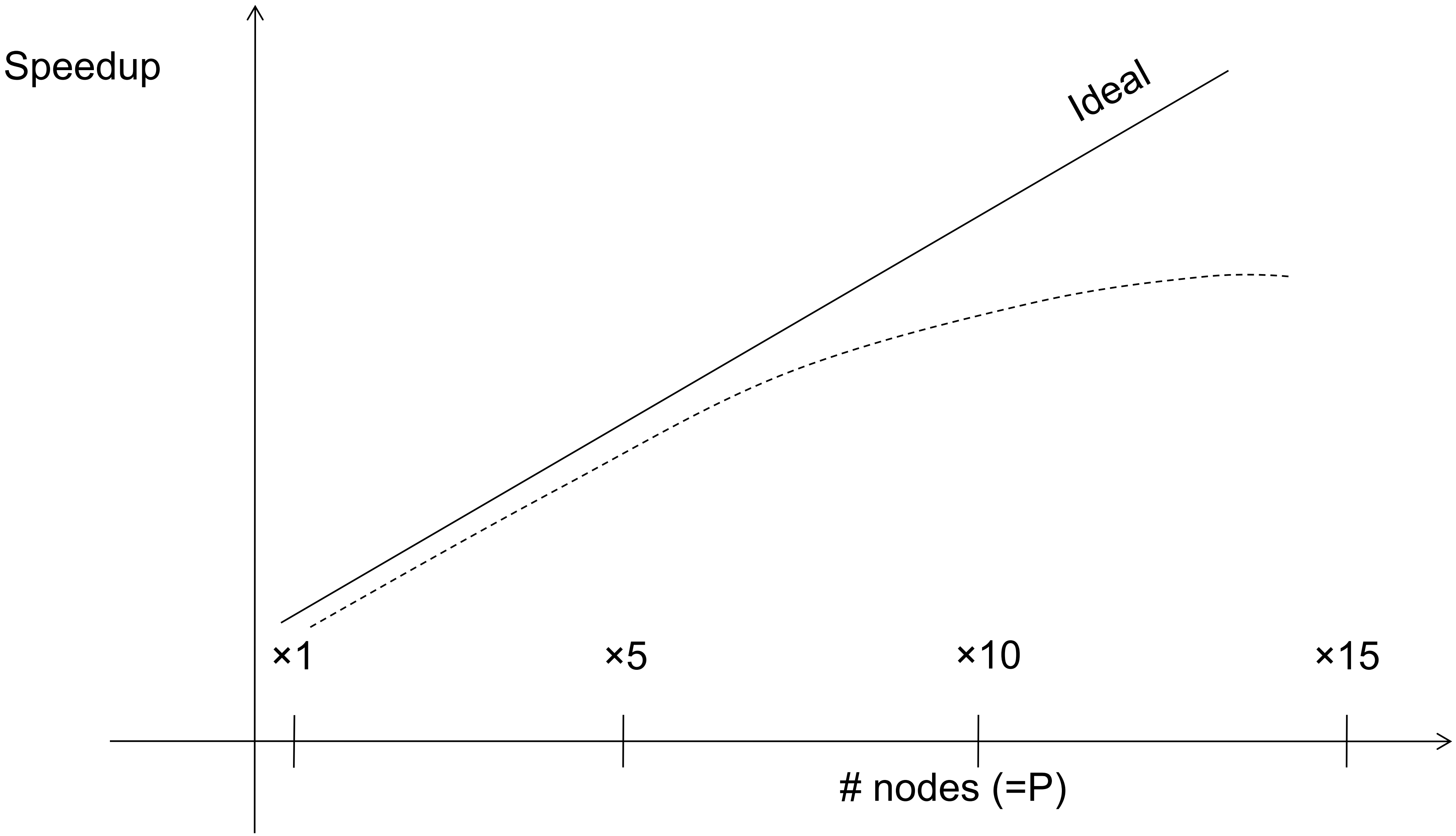
Speedup:

- More nodes, same data ➔ higher speed

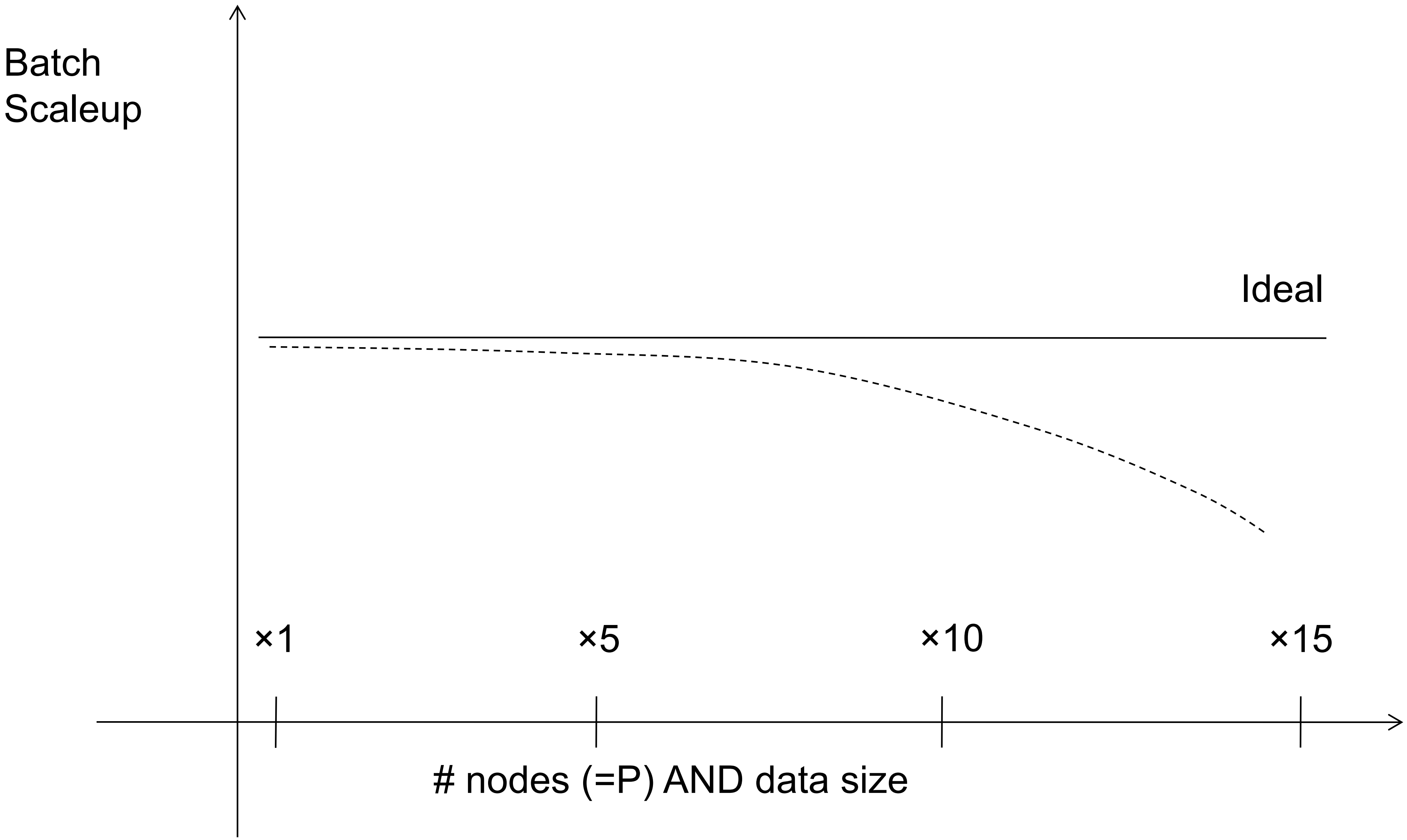
Scaleup:

- More nodes, more data ➔ same speed

LINEAR V.S. NON-LINEAR SPEEDUP



LINEAR V.S. NON-LINEAR SCALEUP



WHY SUB-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

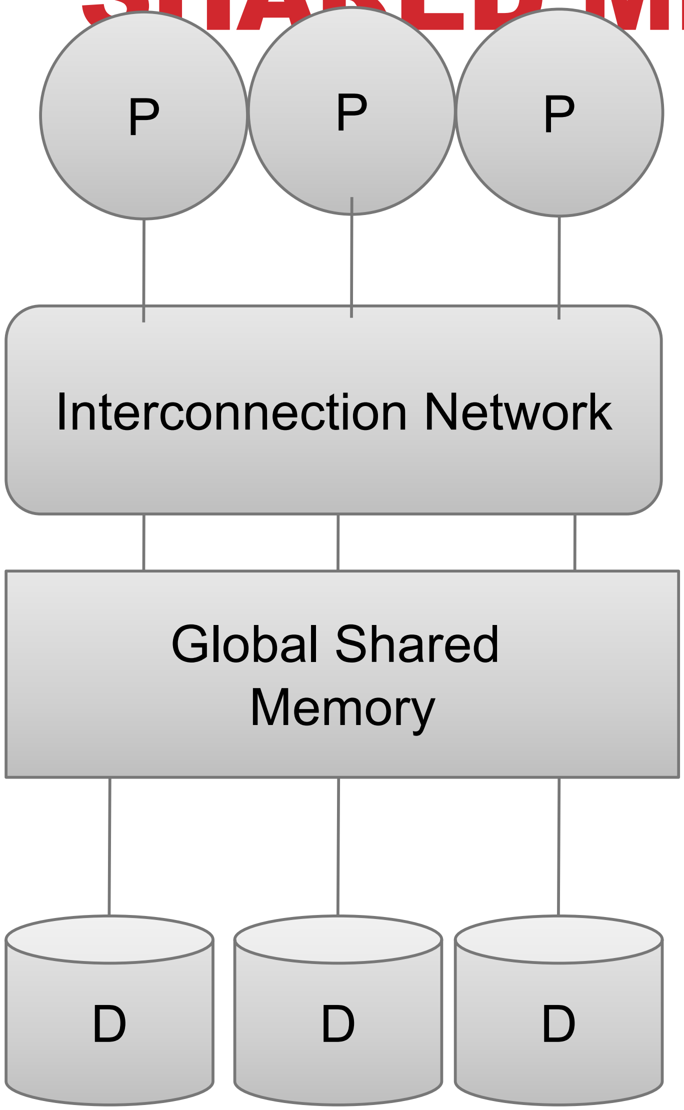
ARCHITECTURES FOR PARALLEL DATABASES

Shared memory

Shared disk

Shared nothing

SHARED MEMORY



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 speed up a query

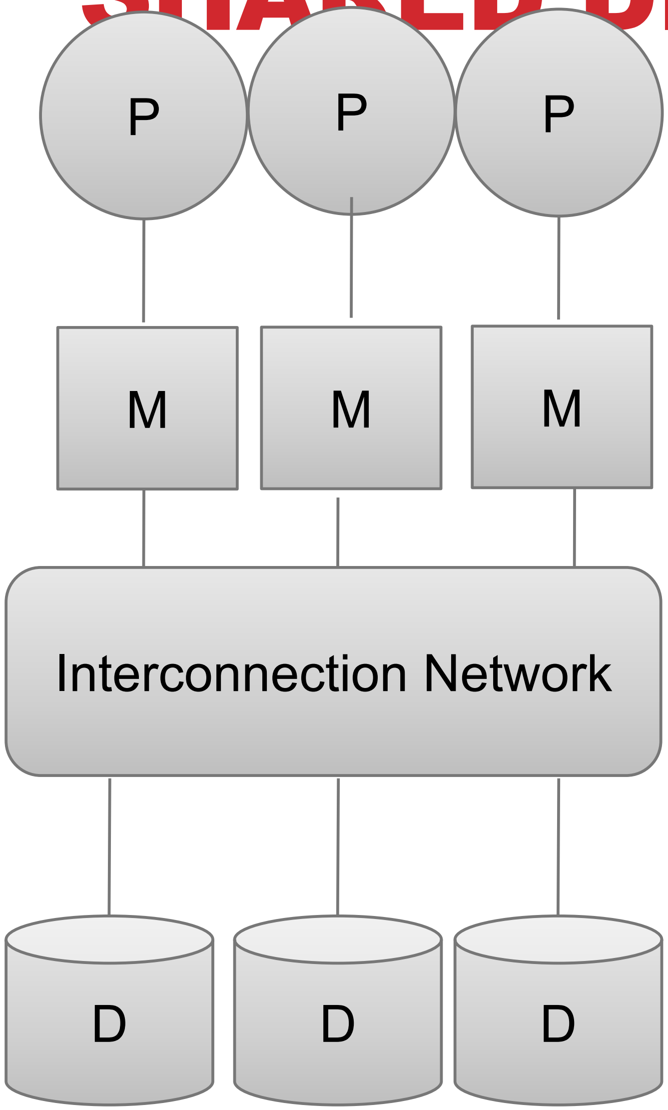
check your HW3 query plans

Easy to use and program

Expensive to scale

- last remaining cash cows in the hardware industry

SHARED DISK



All nodes access the same disks

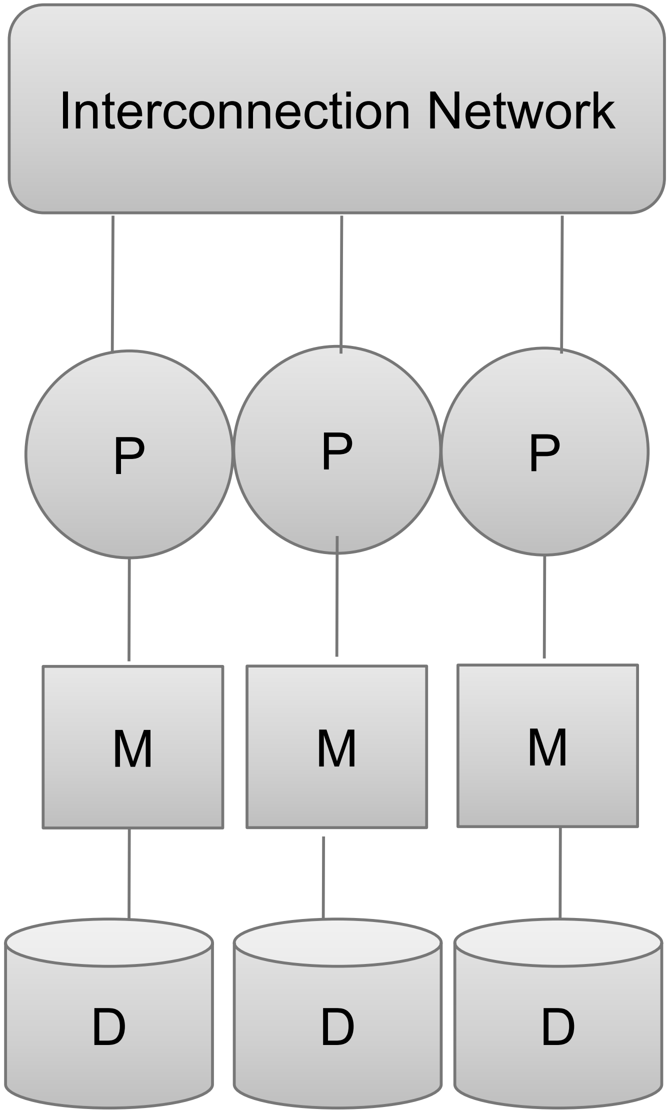
Found in the largest "single-box" (non-cluster) multiprocessors

Example: Oracle

No need to worry about shared memory

Hard to scale: existing deployments typically have fewer than 10 machines

SHARED NOTHING



Cluster of commodity machines on high-speed network

Called "clusters" or "blade servers”

Each machine has its own memory and disk: lowest contention.

Example: Google, Microsoft Cloud

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

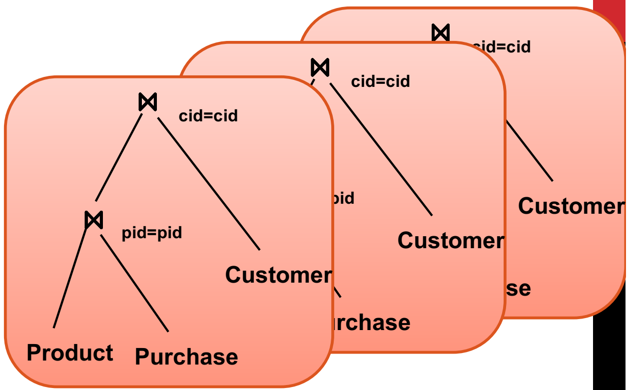
We discuss only Shared Nothing in class

Most difficult to administer and tune.

APPROACHES TO PARALLEL QUERY EVALUATION

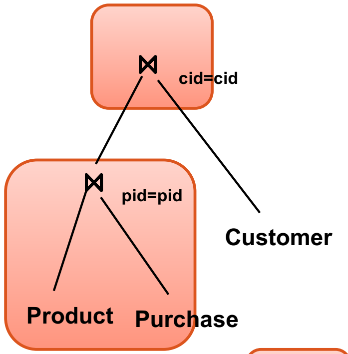
Inter-query parallelism

- Transaction per node
- Good for transactional workloads



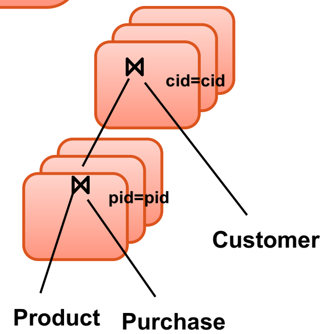
Inter-operator parallelism

- Operator per node
- Good for analytical workloads



Intra-operator parallelism

- Operator on multiple nodes
- Good for both?



We study only intra-operator parallelism: most scalable

DISTRIBUTED QUERY PROCESSING

Data is horizontally partitioned on many servers

Operators may require data reshuffling

First let's discuss how to distribute data across multiple nodes / servers

SINGLE NODE QUERY PROCESSING (REVIEW)

Given relations R(A,B) and S(B, C), no indexes:

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 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 B as key
- Scan file R, probe the hash table using B

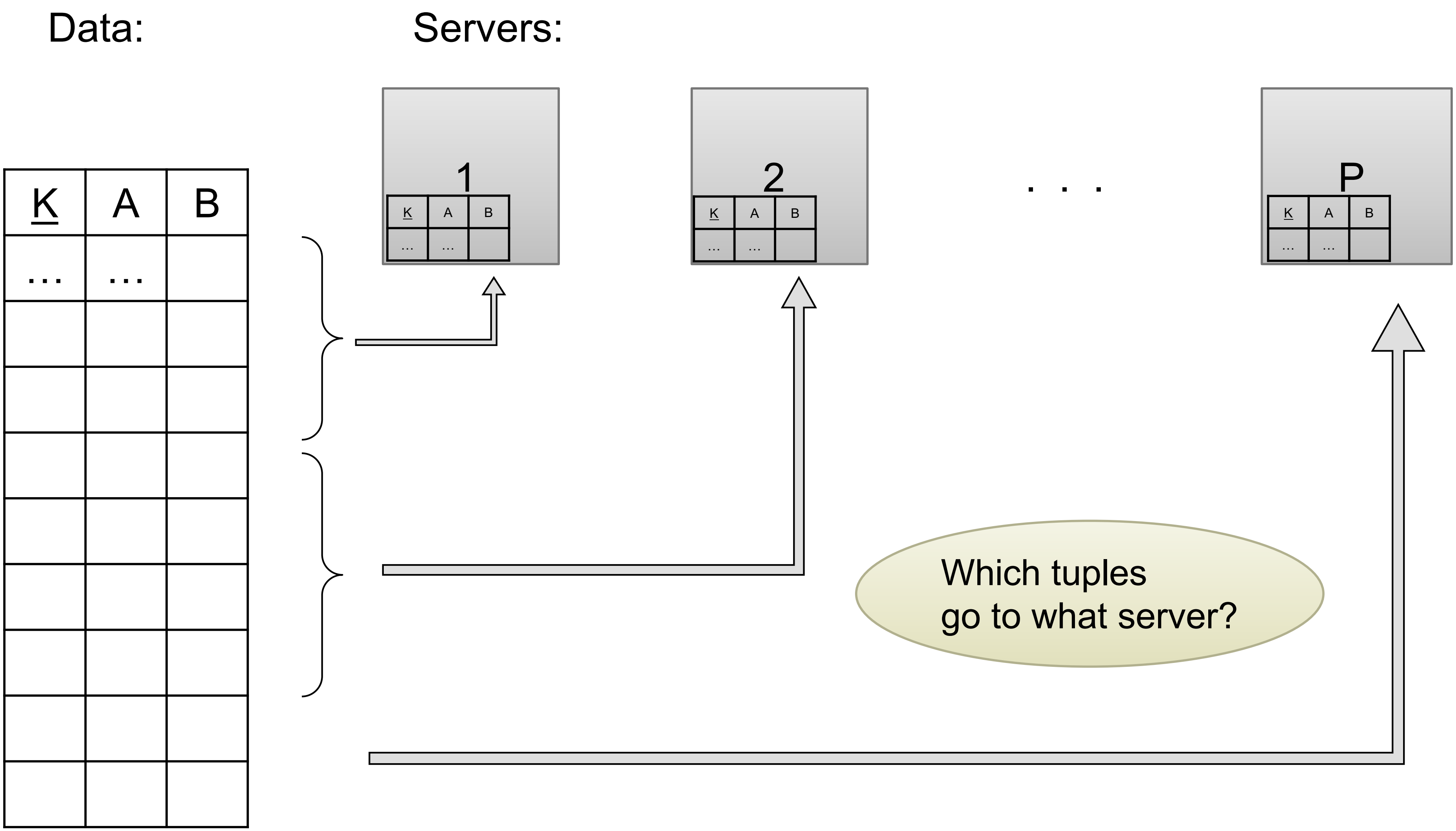
HORIZONTAL DATA PARTITIONING

Data:

K	A	B
…	…	

Servers:

HORIZONTAL DATA PARTITIONING



HORIZONTAL DATA PARTITIONING

Block Partition:

- Partition tuples arbitrarily s.t. $size(R_1) \approx \ldots \approx size(R_P)$

Hash partitioned on attribute A:

- Tuple t goes to chunk i, where i = h(t.A) mod P + 1
- Recall: calling hash fn's is free in this class

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

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 — Uniform

Hash-partition — Uniform

Assuming good hash function and mode is not too large

- On the key K
- On the attribute A

Range partition — 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

Keep this in mind in the next few slides

PARALLEL EXECUTION OF RA OPERATORS: GROUPING

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

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

How to compute group by if:

R is hash-partitioned on A ?

R is block-partitioned ?

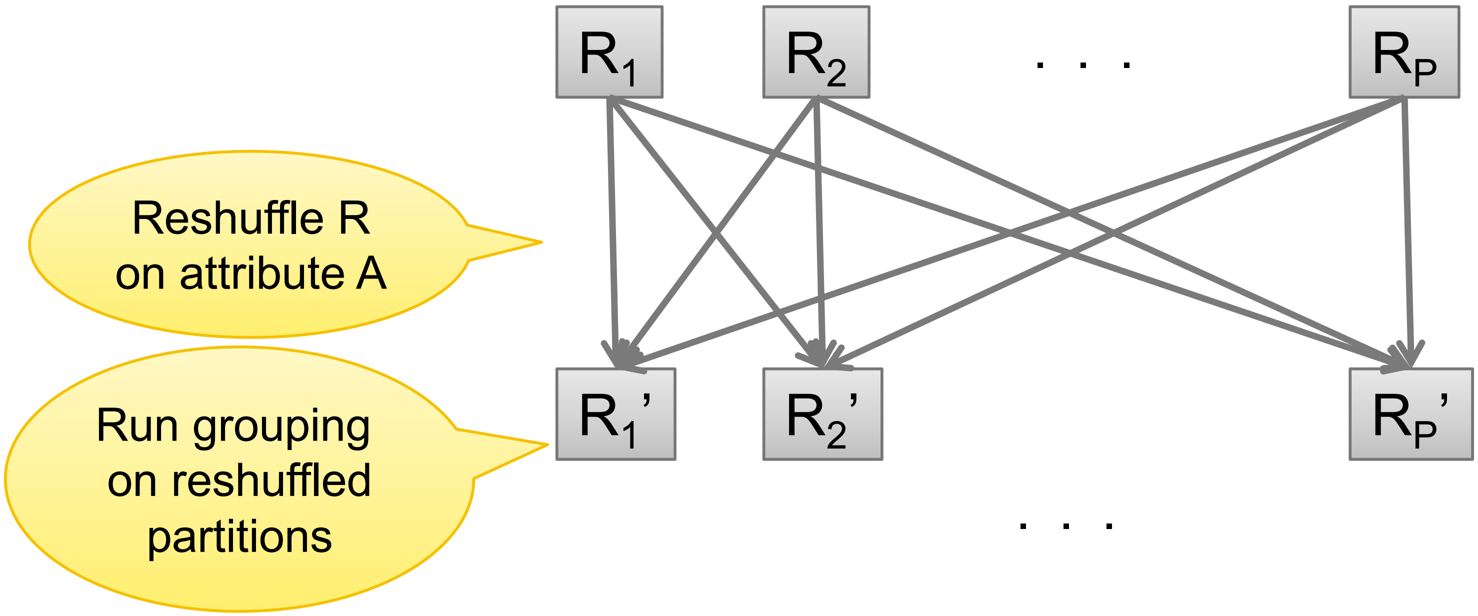
R is hash-partitioned on K ?

PARALLEL EXECUTION OF RA OPERATORS: GROUPING

Data: R(<u>K</u>,A,B,C)

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

R is block-partitioned or hash-partitioned on K



SPEEDUP AND SCALEUP

Consider:

- Query: $\gamma_{A,sum(C)}(R)$
- Runtime: only consider Disk I/O costs

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)

But only if the data is without skew!

SKEWED DATA

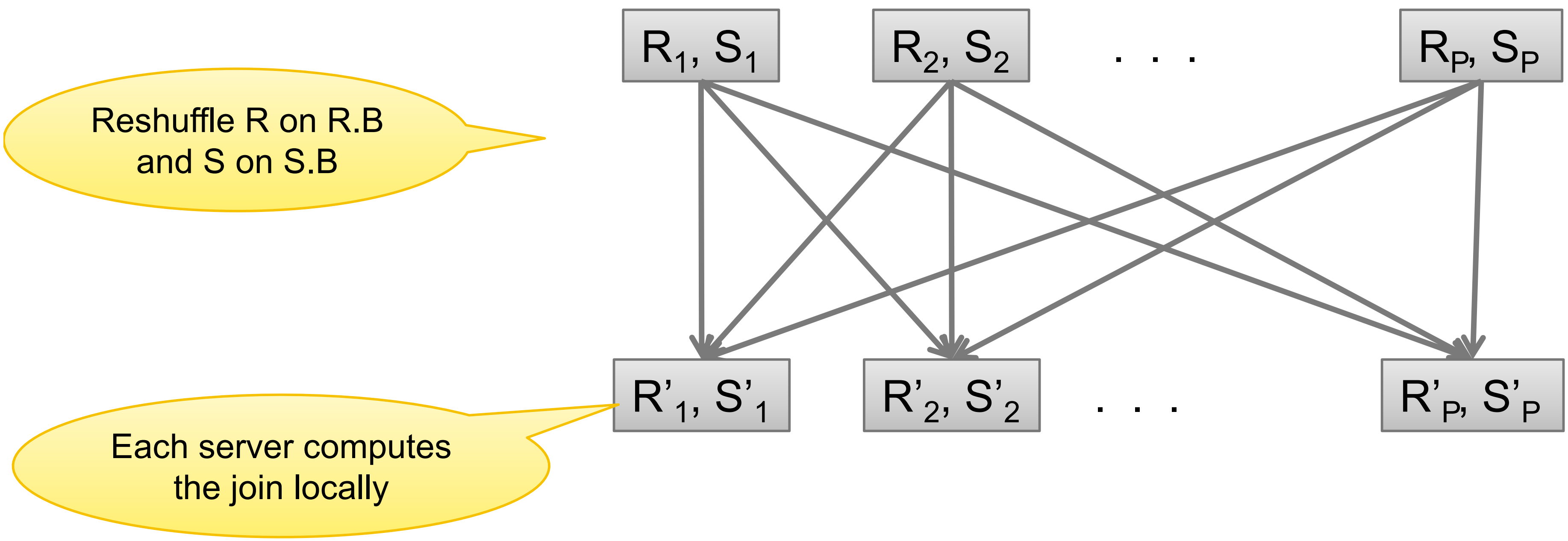
- **R(<u>K</u>,A,B,C)**
- **Informally: we say that the data is skewed if one server holds much more data that the average**
- **E.g. we hash-partition on A, and some value of A occurs many times**
- **Then the server holding that value will be skewed**

PARALLEL EXECUTION OF RA OPERATORS: PARTITIONED HASH-JOIN

Data: **R(<u>K1</u>, A, B), S(<u>K2</u>, B, C)**

Query: **R(<u>K1</u>, A, B) ⋈ S(<u>K2</u>, B, C)**

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



Data: R(<u>K1</u>,A, B), S(<u>K2</u>, B, C)

Query: R(<u>K1</u>,A,B) $\bowtie$ S(<u>K2</u>,B,C)

PARALLEL JOIN ILLUSTRATION

Partition

M1

R1

K1	B
1	20
2	50

S1

K2	B
101	50
102	50

M2

R2

K1	B
3	20
4	20

S2

K2	B
201	20
202	50

Shuffle on B

Local Join

M1

R1’ $\bowtie$ S1’

K1	B
1	20
3	20
4	20

K2	B
201	20

M2

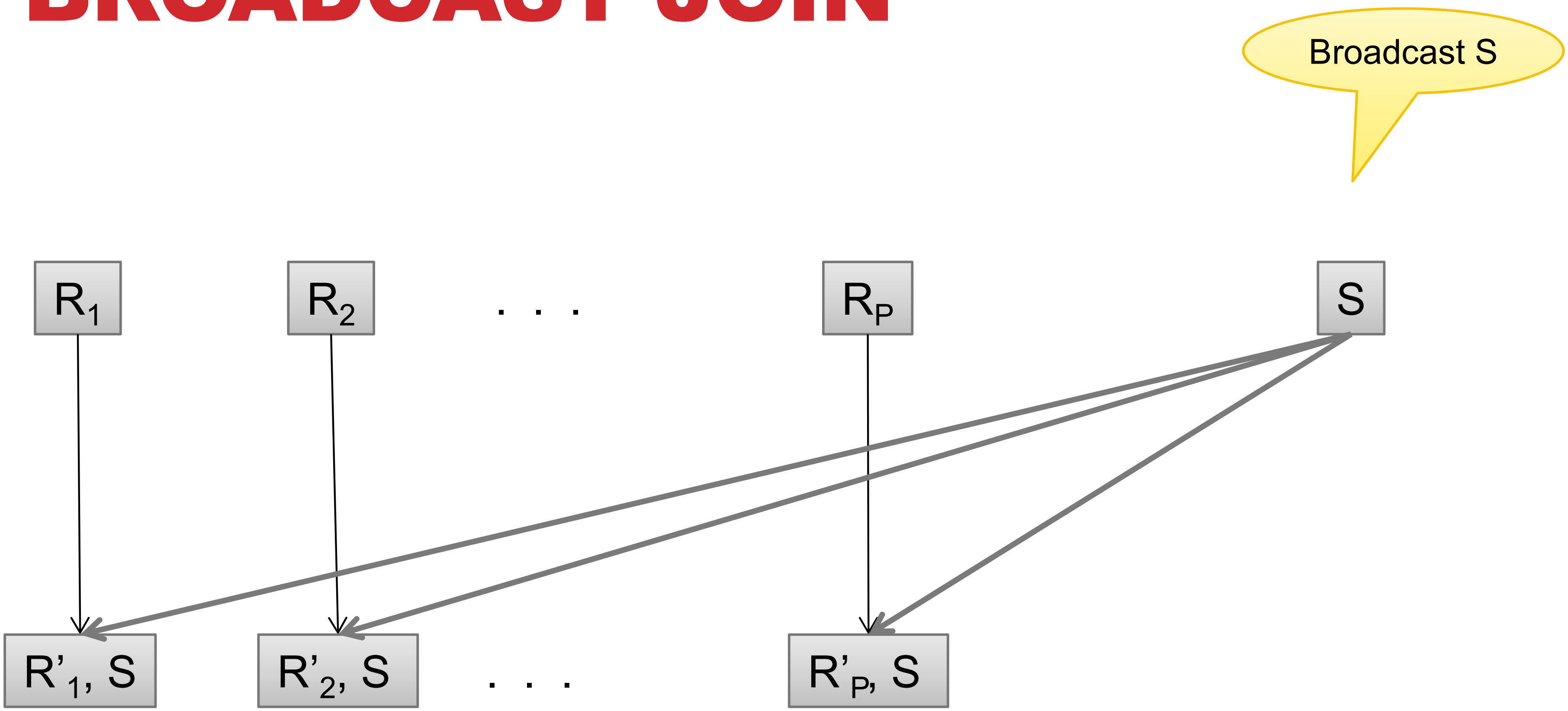
R2’ $\bowtie$ S2’

K1	B
2	50

K2	B
101	50
102	50
202	50

Data: R(A, B), S(C, D)
Query: R(A,B) $\bowtie_{B=C}$ S(C,D)

BROADCAST JOIN



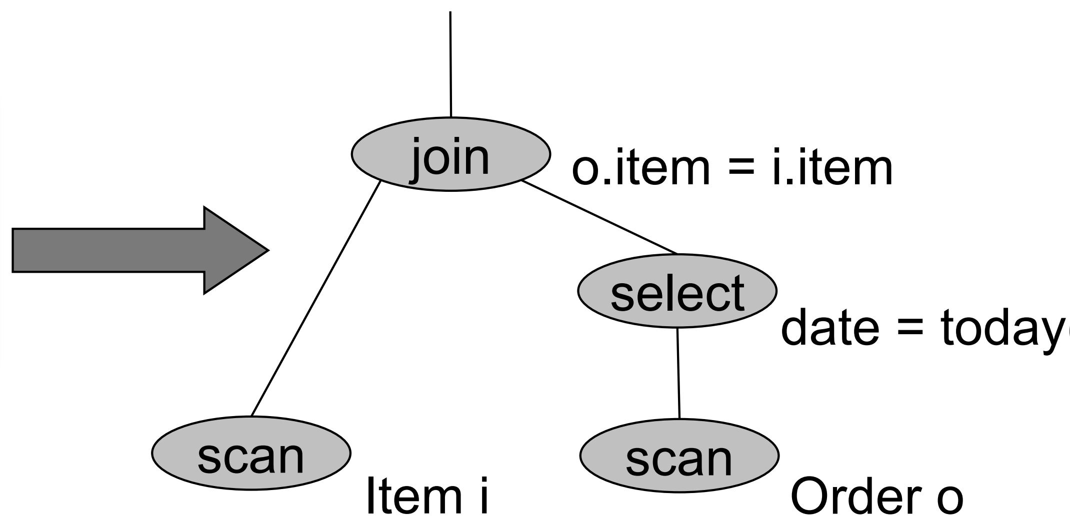
Why would you want to do this?

Order(oid, item, date), Line(item, …)

EXAMPLE PARALLEL QUERY PLAN

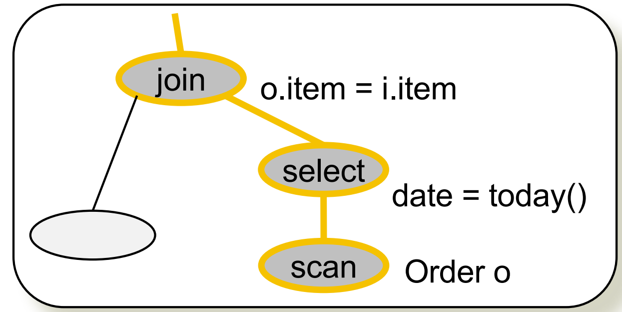
Find all orders from today, along with the items ordered

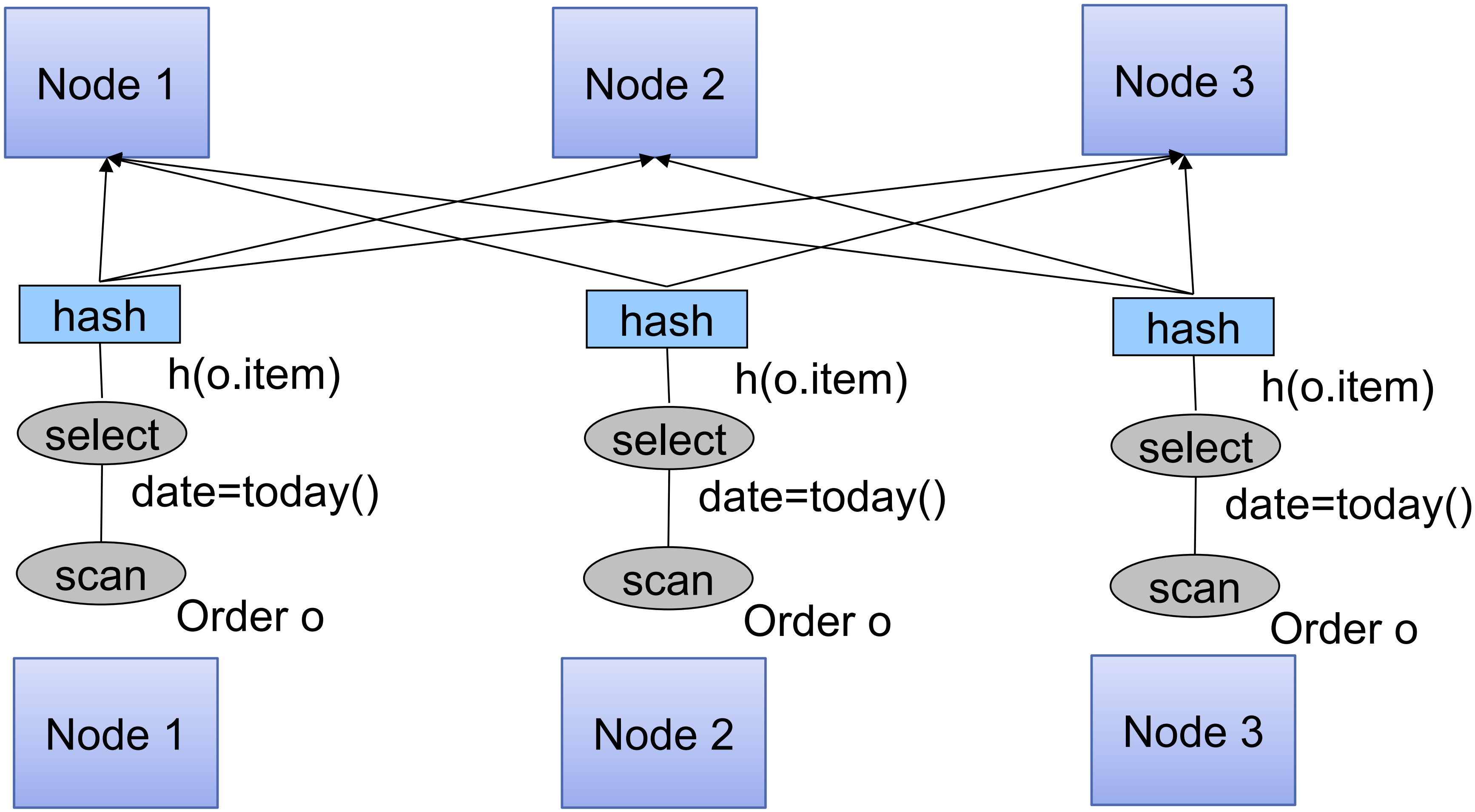
```
SELECT *
  FROM Order o, Line i
 WHERE o.item = i.item
   AND o.date = today()
```



Order(oid, item, date), Line(item, …)

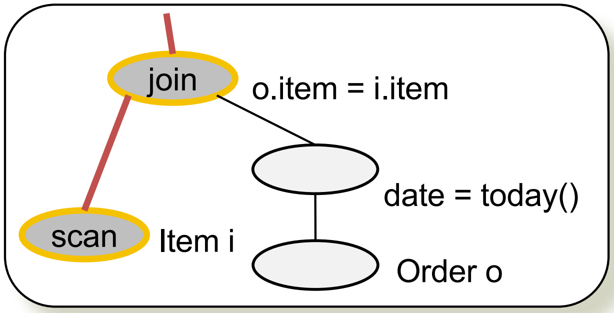
PARALLEL QUERY PLAN

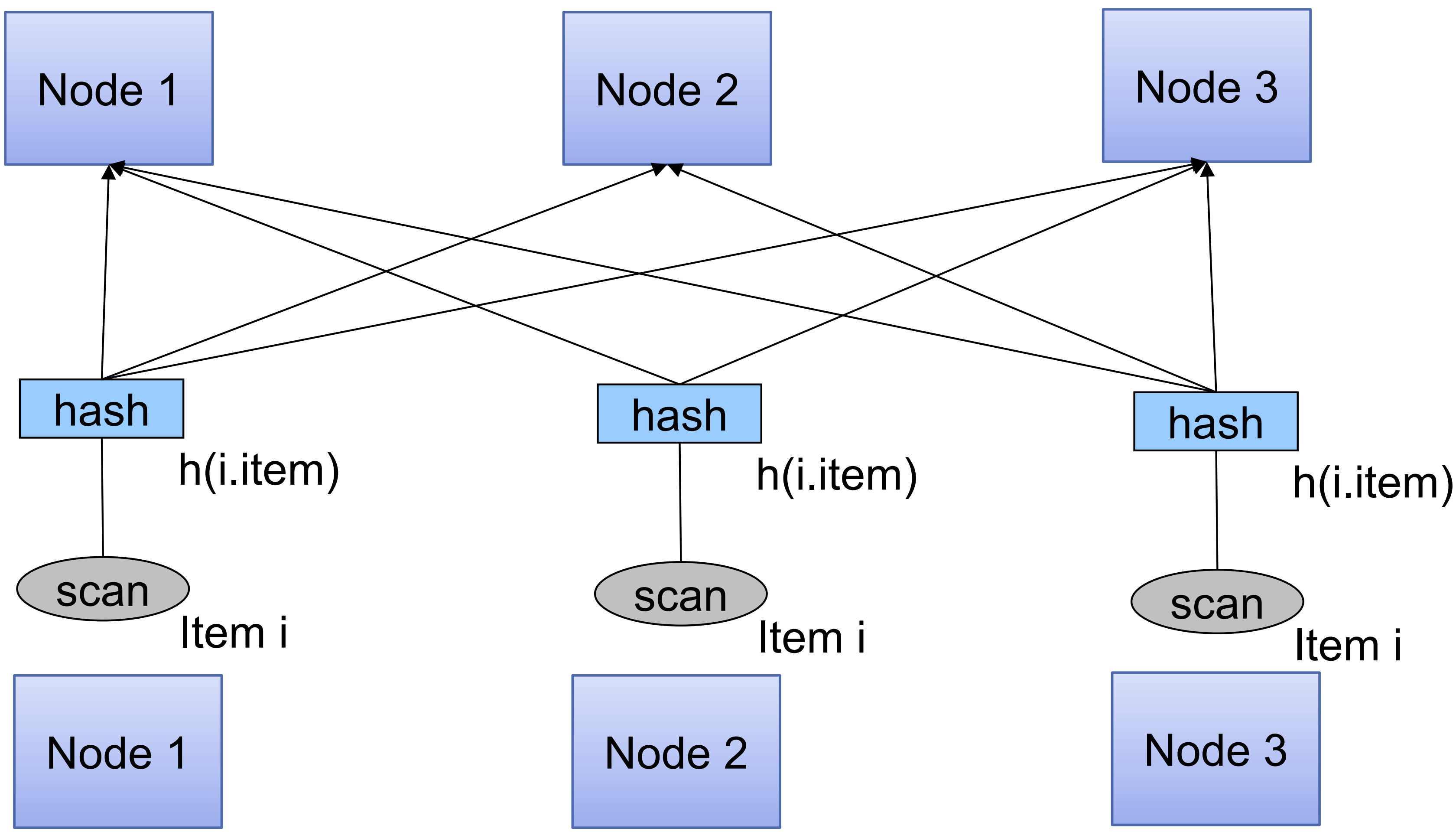




Order(oid, item, date), Line(item, …)

PARALLEL QUERY PLAN





Order(oid, item, date), Line(item, …)

EXAMPLE PARALLEL QUERY PLAN