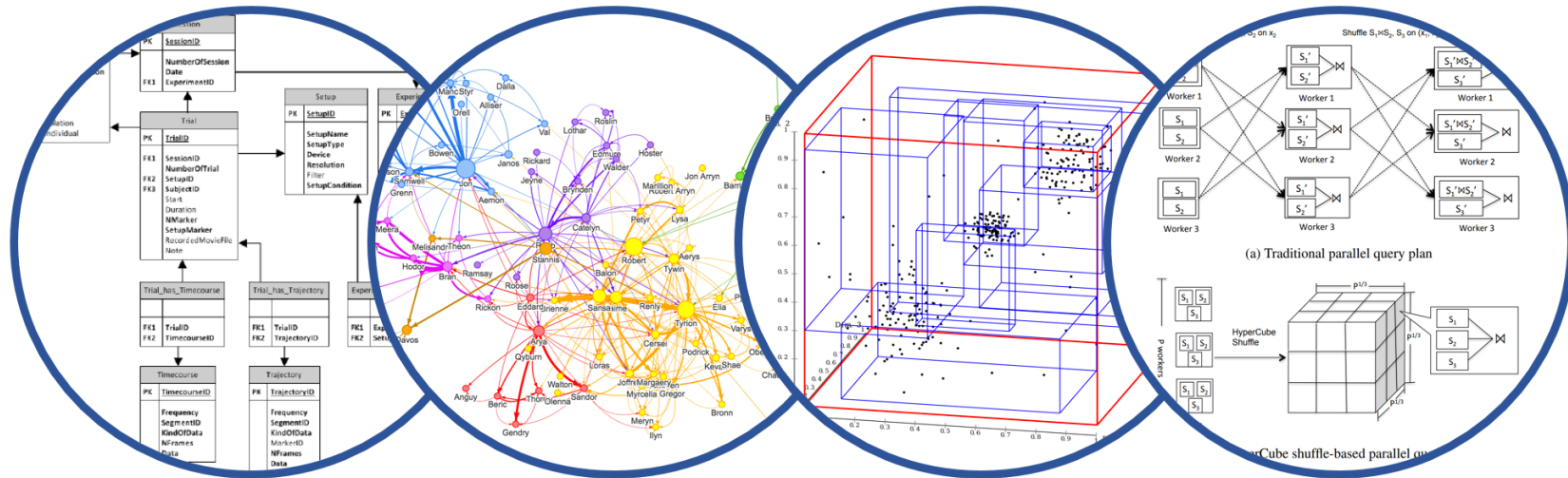




Database System Internals

Intro to Parallel DBMSs

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

What We Have Already Learned

- Phase 1: Query Execution
 - Data Storage and Indexing
 - Buffer management
 - Query evaluation and operator algorithms
 - Query optimization
- Phase 2: Transaction Processing
 - Concurrency control: pessimistic and optimistic
 - Transaction recovery: undo, redo, and undo/redo
- Phase 3: Parallel Processing & Distributed Transactions

Where We Are Headed Next

- **Scaling the execution of a query**
 - Parallel DBMS
 - MapReduce
 - Spark
- **Scaling transactions**
 - Distributed transactions
 - Replication

DATA & AI LANDSCAPE 2019

INFRASTRUCTURE

The diagram illustrates the evolution of data processing technologies, categorized into three main areas:

- HADOOP ON-PREMISE:** Includes logos for Cloudera, Hadoop (Hortonworks), MAPR, Pivotal, IBM InfoSphere, and jethro.
- HADOOP IN THE CLOUD:** Includes logos for AWS, Microsoft Azure, Google Cloud, SAP Cloud Platform, IBM Intelligent Businesslight, arm, Databricks, CAZENA, and Doble.
- STREAMING / IN-MEMORY:** Includes logos for Amazon Kinesis, Google Cloud Dataflow, databricks, SAP Cloud Platform, ORACLE, confluent, strim, haxealcast, GridGain, GIGASPACE, Watarro, PASTDATA, and kx.

NoSQL DATABASES

- Google Cloud
- aws
- ORACLE
- Microsoft Azure
- mongoDB
- MarkLogic
- Couchbase
- DATASAT
- redpanda
- KEPNER
- AzangoDB
- SCYLLA

NewSQL DATABASES

- SAP
- Plurix
- nuodb
- Microsoft Azure
- MEMSQL
- Infuxdata
- Cockroach LABS
- Microsoft Azure
- Voltdb
- Splice
- InfraGraph
- perdata
- Microsoft Azure

GRAPH DBs

- me4u
- Amazon Neptune
- IBM
- ORACLE
- Microsoft Azure
- InfraGraph
- Organice

MPP DBs / TERADATA

- IBM Data Warehouse
- Microsoft Azure
- Qobion
- Kognito
- Exasol
- Microsoft Azure
- Infoworks

CLOUD EDW

- aws
- Google Cloud
- Microsoft Azure
- Pivotal
- Microsoft Azure
- Infoworks

SERVERLESS

- Microsoft Azure
- PULSAR
- nuclia
- Microsoft Azure

DATA TRANSFORMATION

- talend
- pentaho
- atmyer
- TRIFACTA
- streamsets
- StreamSets
- UNIFI

DATA INTEGRATION

SAP Data Services

- Informatica
- Microsoft
- TEALUM
- inlogic
- enigma
- data Cloudscape
- Segment
- ATTUNITY
- Alteryx
- ZALONI
- imput.io
- Infovision
- Fluxion
- OROWFLOW
- MATILLION

DATA GOVERNANCE

- Informatica
- IBM
- Microsoft
- Stylepoint
- collibra
- Alation
- dremio
- Alphamon
- HEXUS
- OKERA
- HANTAI
- data.world

MGMT / MONITORING

- AWS
- New Relic
- octrifio
- rubrik
- dynatrace
- APPRODYNAMICS
- WAVESYSTEM
- SignalFx
- octrifio
- splunk
- Moogsoft
- guardent
- univention
- Namurly
- liberica
- zontec
- octrifio
- HEALTH
- Verbaan

The diagram illustrates the relationship between various cloud services and their categories. The categories and their associated services are as follows:

- STORAGE**
 - Amazon S3
 - Google Cloud Storage
 - Microsoft Azure Storage
 - IBM Cloud Object Storage
 - Oracle Cloud Infrastructure
 - Alibaba Cloud OSS
 - WuXi AppTec
 - Microsoft OneDrive
 - Dropbox
 - Amazon Glacier
 - Google Cloud Storage
 - Microsoft Azure Storage
 - IBM Cloud Object Storage
 - Oracle Cloud Infrastructure
 - Alibaba Cloud OSS
 - WuXi AppTec
 - Microsoft OneDrive
 - Dropbox
 - Amazon Glacier
- CLOUD SVCS**
 - Amazon AWS
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
 - Alibaba Cloud
 - WuXi AppTec
 - Microsoft OneDrive
 - Dropbox
 - Amazon Glacier
 - Google Cloud Storage
 - Microsoft Azure Storage
 - IBM Cloud Object Storage
 - Oracle Cloud Infrastructure
 - Alibaba Cloud OSS
 - WuXi AppTec
 - Microsoft OneDrive
 - Dropbox
 - Amazon Glacier
- DATA GENERATION & LABELLING**
 - Amazon Mechanical Turk
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
 - Alibaba Cloud
 - WuXi AppTec
 - Microsoft OneDrive
 - Dropbox
 - Amazon Glacier
 - Google Cloud Storage
 - Microsoft Azure Storage
 - IBM Cloud Object Storage
 - Oracle Cloud Infrastructure
 - Alibaba Cloud OSS
 - WuXi AppTec
 - Microsoft OneDrive
 - Dropbox
 - Amazon Glacier
- AI OPS**
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
- GPU DBs & CLOUD**
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
- HARDWARE**
 - Google TPUs
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud
 - Alibaba Cloud
 - Google Cloud
 - Microsoft Azure
 - IBM Cloud
 - Oracle Cloud

CROSS-INFRASTRUCTURE/ANALYTICS

ANALYTICS & MACHINE INTELLIGENCE

The diagram is divided into two main sections by a vertical line. The left section is titled 'DATA ANALYST PLATFORMS' and lists several logos: Microsoft, pentaho, alteryx, Digital Reasoning, guavus, AYASDI, ATTIVO, Datameer, incorta, interana, MODO, ENDOR, and Kogniti. The right section is titled 'DATA SCIENCE PLATFORMS' and lists several logos: IBM, databricks, dataiku, DOMINO, rapidminer, TIBCO, sas, ANACONDA, and MathWorks.

BI PLATFORMS

- looker
- tableau
- crucible analytics
- aws
- domo
- arc42 data
- thoughtspot
- atscale
- aliqui
- gooddata
- information builders
- aliqui
- microstrategy
- keen io

VISUALIZATION

- tableau
- power bi
- sap
- luma
- google cloud
- celonis
- zepl
- graphix
- chariot
- plioty
- flexcube tech

MACHINE LEARNING

- accure
- machine learner
- amazon sage maker
- google cloud
- vertex ai
- h2o
- data robot
- gamalio
- visenze
- element
- deepepsilon
- ai

COMPUTER VISION

- Microsoft Azure
- Amazon Rekognition
- Clarifai
- IBM
- EVER AI
- deepomatic
- nuance
- twentyone
- ubiquitous
- ACOE
- trax
- datarobot
- synthesis

HORIZONTAL AI

- Watson
- Cortana
- Face
- 80
- sentient
- Voyager Labs
- Amazon Transcribe
- Affective
- Petuum
- Numenta
- Narologic
- Curious AI
- OSARO
- IBM

SPEECH & NLP

- Google Cloud
- Twitter
- Amazon Alexa
- Amazon Transcribe
- narrative science
- semantic machine
- Mobvoi
- PRIMER
- SoundHound Inc.
- BlueMatrix
- IBM
- Uniphore
- Prophet

A collage of logos for various companies, organized into four columns:

- SEARCH:** elasticsearch, algolia, covéo, Lucidworks, ATTIVOQ, swifttype, EXOGENE, alpharange, MAANA, omni-us, SINEQUA.
- LOG ANALYTICS:** ORACLE, splunk, sumologic, solarwinds, TIMELINE, kibana, Logz.io.
- SOCIAL ANALYTICS:** Hootsuite, sprinklr, NETBASE, symphony, tracx, simple reach, bitly, SimilarWeb.
- WEB / MOBILE / COMMERCE ANALYTICS:** Google Analytics, mixpanel, AMPITUDE, Airtable, RESCUE, SIGOPT, granify.

OPEN SOURCE

The diagram illustrates a comprehensive ecosystem of data science and machine learning tools, organized into 12 functional categories:

- FRAMEWORKS:** Includes Databricks, Spark, Flink, YARN, Tez, Mesos, Kubernetes, Docker, CDAP, Hadoop, Red Hat, and HeliX.
- QUERY / DATA FLOW:** Includes Spark, SQL, Presto, DDL, SLAM DATA, GraphQL, and Flink.
- DATA ACCESS & DATABASES:** Includes Cassandra, MongoDB, Redis, Cockroach LABS, Druid, and SciDB.
- ORCHESTRATION & MGMT:** Includes Talend, Apache Airflow, and Mesos.
- STREAMING & MESSAGING:** Includes Spark, Flink, Beam, Kafka, Storm, and Apache RocketMQ.
- STAT TOOLS & LANGUAGES:** Includes Python, R, Scala, Jupyter, Studio, SciPy, and Julia.
- AI OPS & INFRA:** Includes MFlow, Kubeflow, and DC.
- AI / MACHINE LEARNING / DEEP LEARNING:** Includes TensorFlow, Keras, PyTorch, Theano, Caffe, MXNet, Dims, FeatureFu, VEs, Chainer, Microsoft, ONNX, PyTorch, and PyTorch.
- SEARCH:** Includes Elasticsearch and Solr.
- LOGGING & MONITORING:** Includes Kibana, SENTRY, Logstash, Prometheus, Fluentbit, Fluentd, Grafana, and Vector.
- VISUALIZATION:** Includes matplotlib, TensorBoard, seaborn, and Tableau.
- COLLABORATION:** Includes BeakerX, Jupyter, and Anaconda.
- SECURITY:** Includes Apache Ranger, KNOX, Sentry, and Accumulo.

DATA SOURCES & APIs

HEALTH

Apple, VALIDIC, practicefusion, fitbit, GARMIN, HUMAN APP, kinsa, MIMIC

IOT

GE Digital, UPTAKE, thingworx, helium, samsara, rastreamento

FINANCIAL & ECONOMIC DATA

Bloomberg, THOMSON REUTERS, DOW JONES, S&P CAPITAL IQ, CB Insights, PLAID, Fitch MEASURE, INVESTMENT FLOOD, THE WALL STREET JOURNAL, estimate, PREMIERE, Quandl, Alpha Alpha, StockTwits, xignite, Thinknum, earnest, predata

AIR / SPACE / SEA

Orbital Insight, planet, SATCATCH, AEROBOTICS, spire, kespri, INFLUENCER, UNDERSTORY, telluslabs, WINDWARD, DroneDeploy, MarineTraffic, LOGI DIGITAL, IRIS PERCS

PEOPLE / ENTITIES

acxiom, experian, EPILSON, InsideView, Crimson Hexagon, BASIS, Quantcast, SAFEGRAPH

LOCATION INTELLIGENCE

FOURSQUARE, Mapbox, sense360, privacy boxes, HEXAGON, PlaceIQ, esri, factual, CARTA, Mapillary, OpenStreetMap, cuebiq, Radar, OpenStreetMap

OTHER

DATA.GOV, IBM GENET, Labcorp, CRUX, algrat.io

DATA RESOURCES

DATA SERVICES

- OPERA
- UCL
- DATA SCIENCE
- fractal
- kaggle
- DataKind
- EXL
- INNOCPUXUS

INCUBATORS & SCHOOLS

- PLURALSIGHT
- GA
- galvanize
- DataCamp
- DataElite
- INSIGHT
- The Data Incubator
- METIS

RESEARCH

- OpenAI
- facebook research
- MIRI
- MLA
- VECTOR INSTITUTE
- CSAIL
- DFK
- AI2
- ALLEN INSTITUTE FOR ARTIFICIAL INTELLIGENCE

APPLICATIONS – ENTERPRISE

SALES

- CH@RUS
- INSIDE.SALES.com
- conversica
- clari
- aviso
- tactai
- troops
- fusers/machines
- Clearbit

MARKETING - B2B

- RADIUS
- EVERSTING
- Lattice
- HINTIGO
- sense
- tubular
- EN NGAGIO
- KNOTCH
- mip
- App Annie

MARKETING - B2C

- zeta
- blomio
- reach
- SendGrid
- brage
- action
- BLUECORE
- CONTENTIQUE
- amptium
- merpote
- Amparo
- SPARKTEL
- QUANTIFIED
- Simon
- Stylife
- PERASO
- remesh

CUSTOMER EXPERIENCE / SERVICE

- qualtrics
- MEDALLIA
- SurveyMonkey
- Utest Testing
- CLARABRIE
- zendesk
- Customer
- helpdesk
- INTERAC
- liveperson
- Gainsight
- pendo
- HEAP
- Amplitude
- Watson Assistant
- Dialogflow
- DigitalGems
- ASAPP
- ada
- automat
- ahnhit
- CarDesk
- recloni
- adform
- iframa

ENTERPRISE PRODUCTIVITY

- slack
- ORACLE
- GURU
- luminator
- DIFFBOT
- klars
- talia
- castro

HUMAN CAPITAL

- Hiya
- symetrics
- hiQ
- GHOSTER
- mya
- Aillyo
- Wade&Tendy
- Stella
- Clerken
- entelo
- SALTWORKS
- uncommon
- bionomy
- QNT

LEGAL

- Ravel
- Evolv
- disco
- Kira
- JUDICATA
- BREVIA
- IRONCLAD
- PROMOTION
- ROSS
- castext

REGTECH & COMPLIANCE

- digitix
- TERRIAN
- Comply Advantage

FINANCE

- nplan
- Zwora
- SANANA
- TRADESHIP
- SCALEFACTOR
- bukeeper
- pilot

BACK OFFICE AUTOMATION & RPA

- UiPath
- blueprints
- VIDADO
- Workfusion
- worlato
- Redwood
- Catalytic
- NETWORKS
- automation
- KRYON
- AKVPII

SECURITY

- TANIUM
- CYCLANE
- zscaler
- StackPath
- Illumio
- CODE42
- CipherCloud
- DARKTRACE
- ANOMALI
- Trend Micro
- VECTRA
- pinDrops
- exabeam
- SINEGY
- SentinelOne
- SecurityScorecard
- SOCCURE
- CodeSecure
- bitglass
- Blueloton
- Recorded Future
- feedzai
- Cyber
- BITSIGHT
- spionagegroup
- Insider Cyber
- FORTER
- riskcrypt
- JA S K
- ARAT
- BLUECHECKPOINT
- Semantic
- ORION
- XENIOUS
- SHIELD A
- Amnoria

PARTNERSHIPS

- EUROPEAN
- INFORM
- DATA REPUBLIC

APPLICATIONS – INDUSTRY

[illegible]

How to Scale the DBMS?

- Can easily replicate the web servers and the application servers
- We cannot so easily replicate the database servers, because the database is unique
- We need to design ways to **scale up the DBMS**

Building Our Parallel DBMS

Data model?

**Relational
(SimpleDB!)**

Building Our Parallel DBMS

Data model?

Relational
(SimpleDB!)

Scaleup goal?

Scaling Transactions Per Second

- **OLTP: Transactions per second**
“Online Transaction Processing”
- Amazon
- Facebook
- Twitter
- ... your favorite Internet application...
- Goal is to increase transaction throughput
- We will get back to this next week

Scaling Single Query Response Time

- OLAP: Query response time
“Online Analytical Processing”
- Entire parallel system answers one query
- Goal is to improve query runtime
- Use case is analysis of massive datasets

Volume alone is not an issue

- Relational databases *do* parallelize easily; techniques available from the 80's
 - Data partitioning
 - Parallel query processing
- SQL is *embarrassingly parallel*
 - We will learn how to do this!

New **workloads** are an issue

- Big volumes, small analytics
 - OLAP queries: join + group-by + aggregate
 - Can be handled by today's RDBMSs
- Big volumes, big analytics
 - More complex Machine Learning, e.g. click prediction, topic modeling, SVM, k-means
 - Requires innovation – Active research area

Building Our Parallel DBMS

Data model?

Relational

Scaleup goal?

OLAP

Building Our Parallel DBMS

Data model?

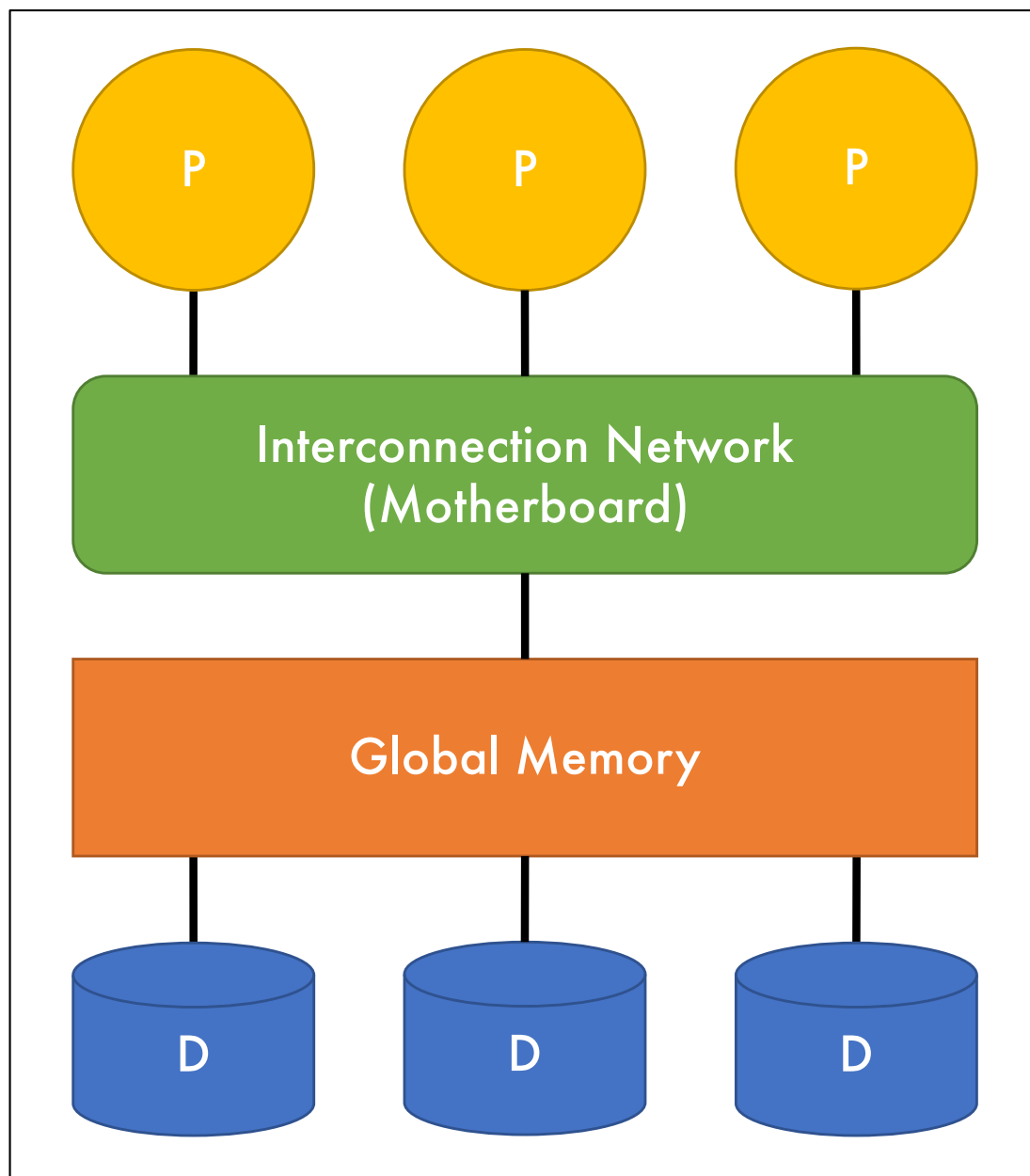
Relational

Scaleup goal?

OLAP

Architecture?

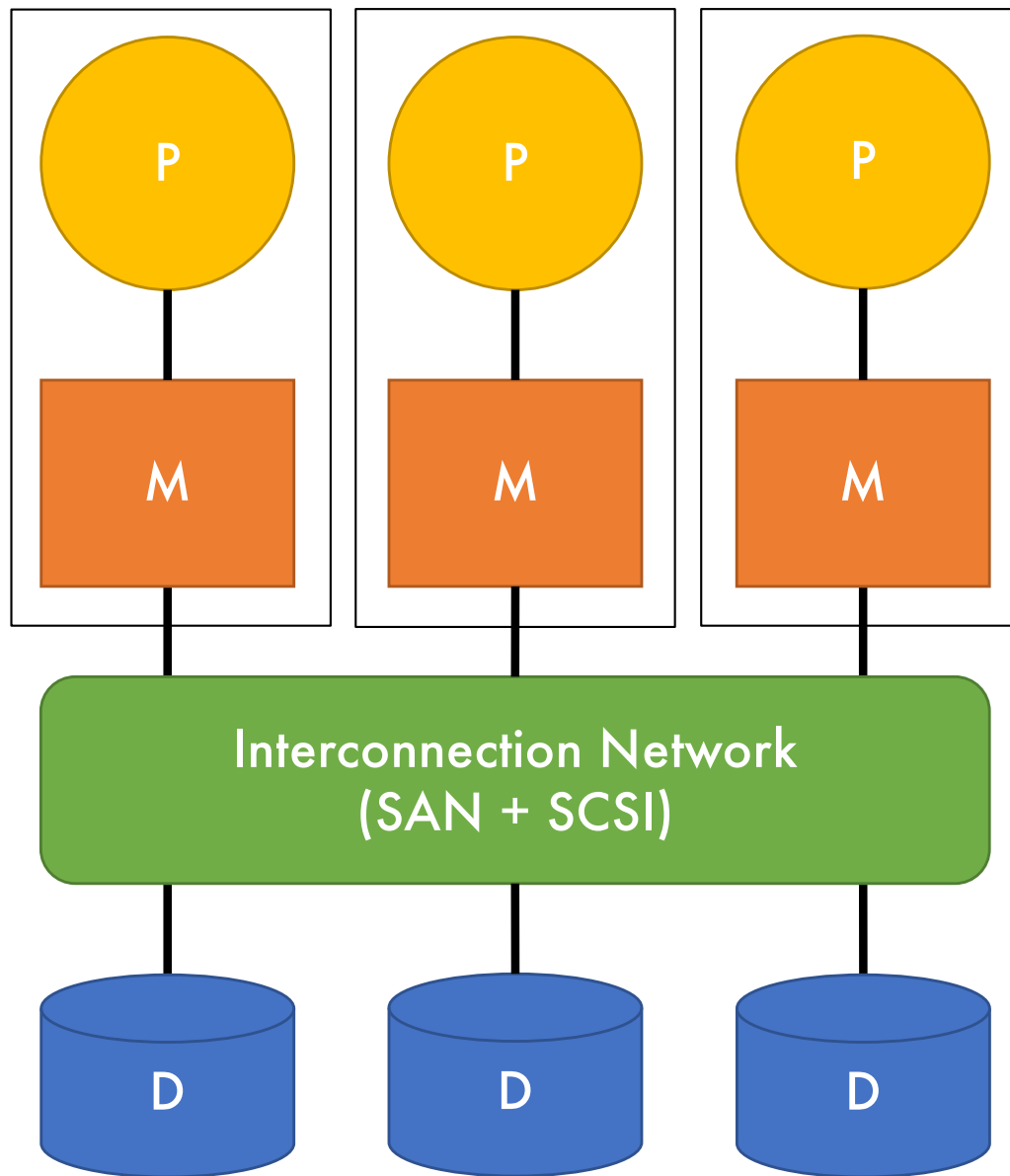
Shared-Memory Architecture



- Shared main memory and disks
- Your laptop or desktop uses this architecture
- **Expensive to scale**
- **Easiest to implement on**



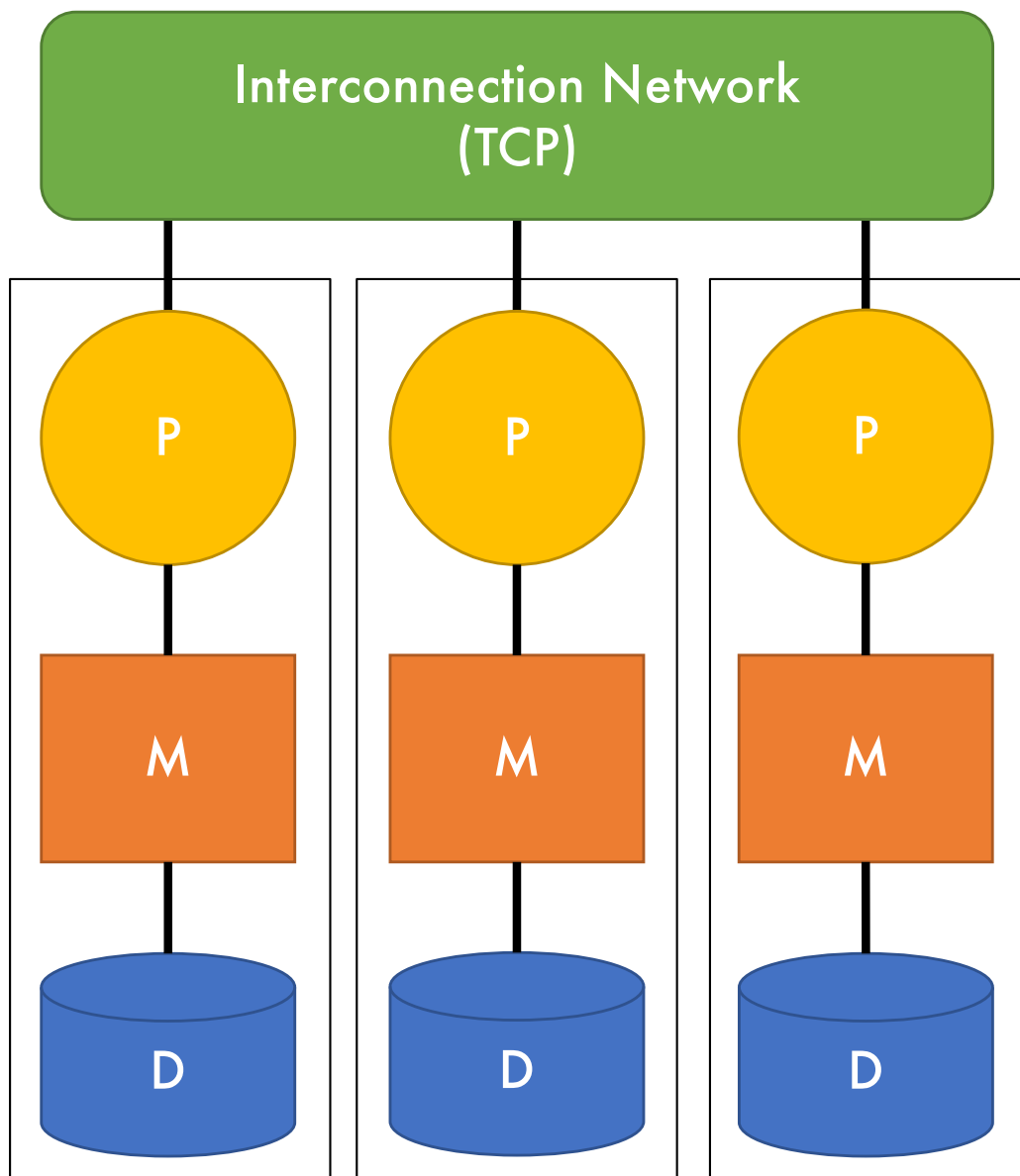
Shared-Disk Architecture



- Only shared disks
- No contention for memory and high availability
- Typically 1-10 machines

ORACLE[®]
DATABASE

Shared-Nothing Architecture



- Uses cheap, commodity hardware
- No contention for memory and high availability
- Theoretically can **scale infinitely**
- Hardest to implement on

teradata.

APACHE
Spark[™]

MySQL[™] Cluster

Building Our Parallel DBMS

Data model?

Relational

Scaleup goal?

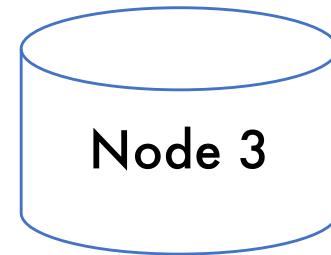
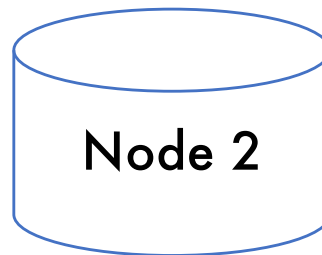
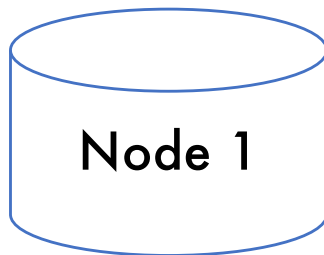
OLAP

Architecture?

Shared-Nothing

Shared-Nothing Execution Basics

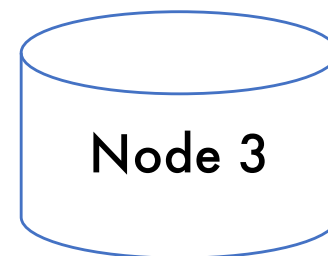
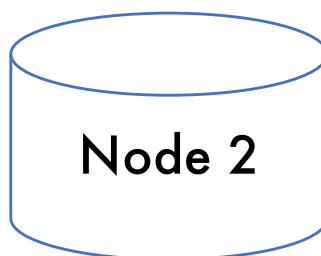
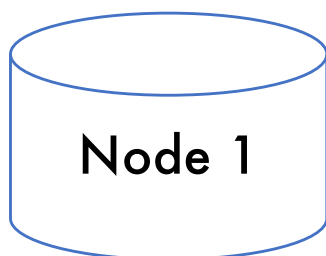
- Multiple DBMS instances (= processes) also called “nodes” execute on machines in a cluster
 - One node plays role of the coordinator
 - Other nodes play role of workers
- Workers execute queries
 - Typically **all workers execute the same plan**
 - Workers can execute multiple queries at the same time



Shared-Nothing Database

We will assume a system that consists of multiple commodity machines on a common network

New problem: **Where does the data go?**

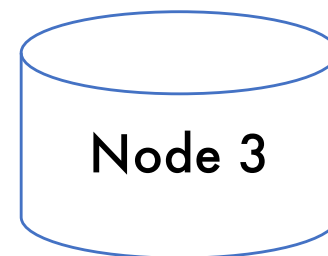
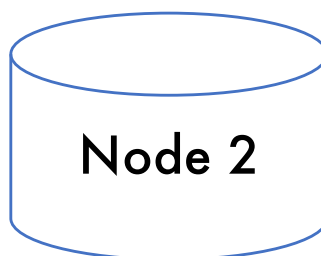
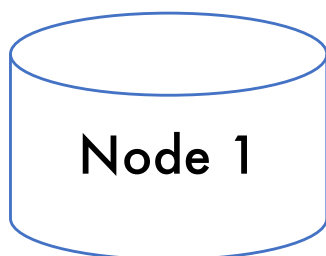


Shared-Nothing Database

We will assume a system that consists of multiple commodity machines on a common network

New problem: **Where does the data go?**

The answer will influence our execution techniques

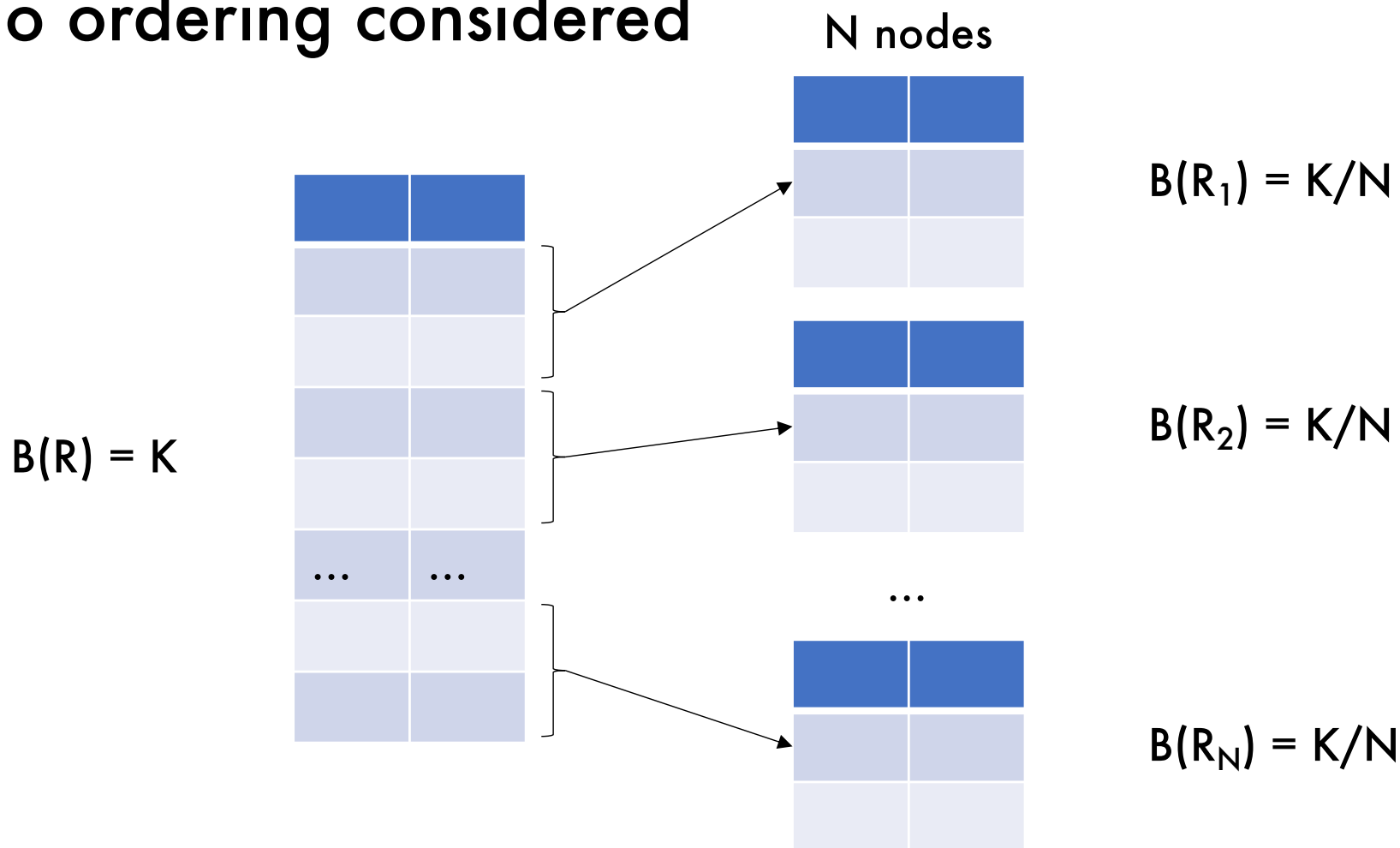


Option 1: Unpartitioned Table

- Entire table on just one node in the system
- Will bottleneck any query we need to run in parallel
- We choose partitioning scheme to divide rows among machines

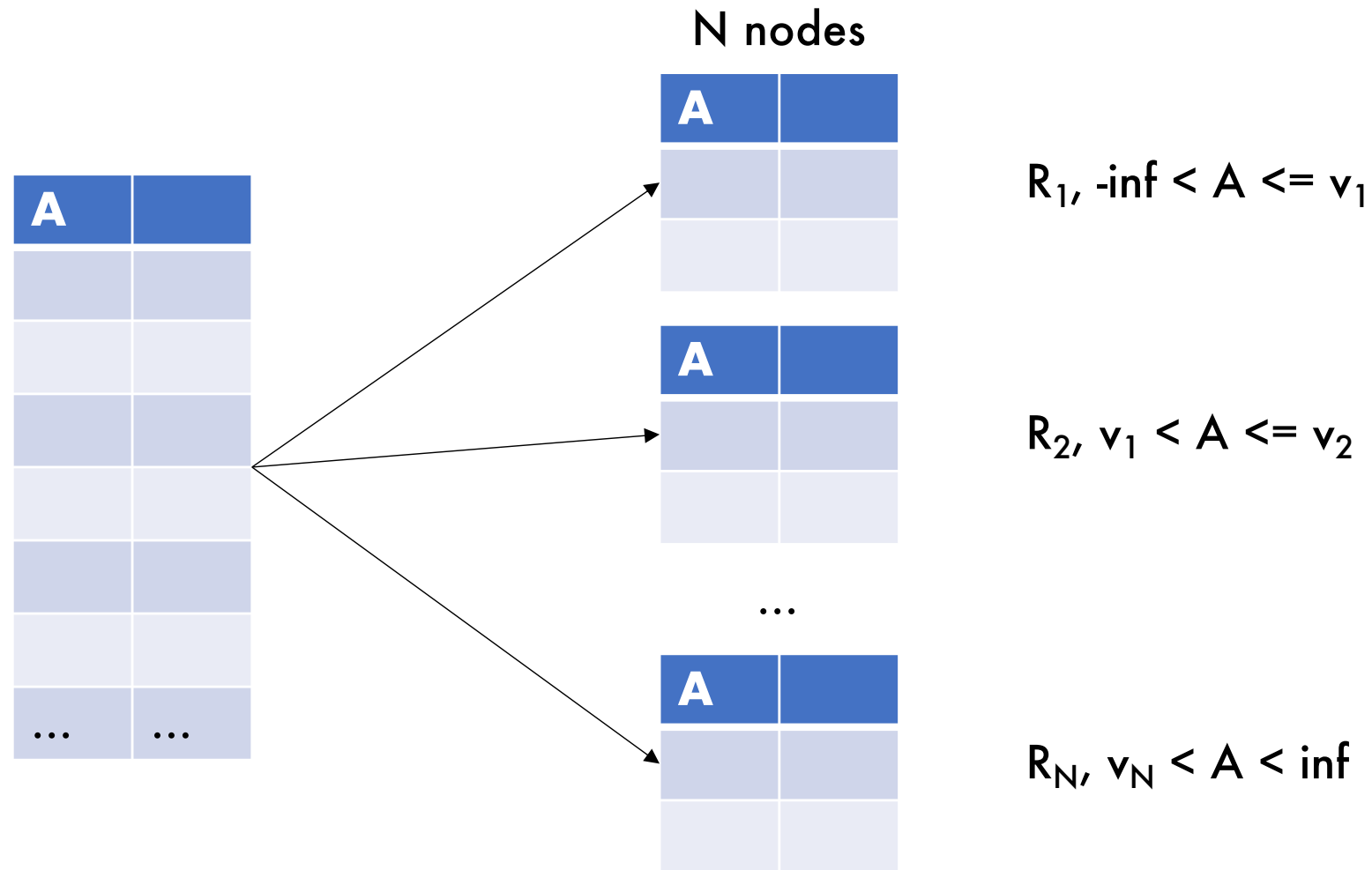
Option 2: Block Partitioning

Tuples are horizontally (row) partitioned by raw size with no ordering considered



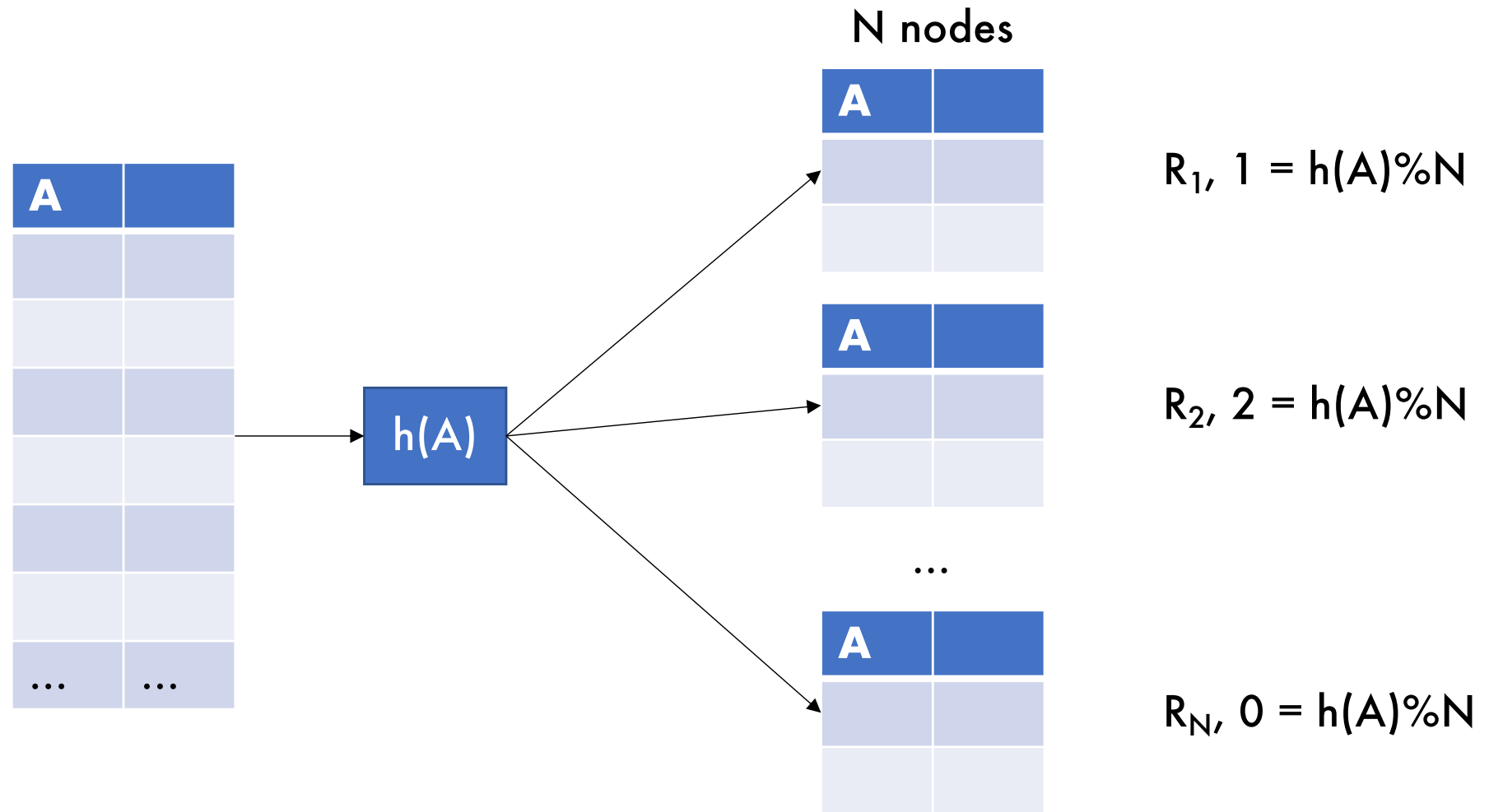
Option 3: Range Partitioning

Node contains tuples in chosen attribute ranges



Option 4: Hash Partitioning

Node contains tuples with chosen attribute hashes



Skew: The Justin Bieber Effect

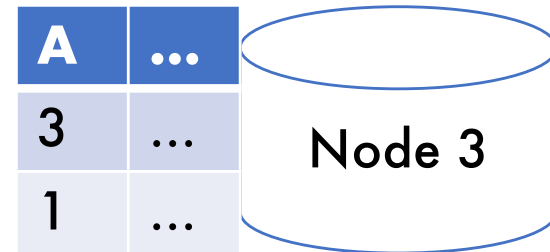
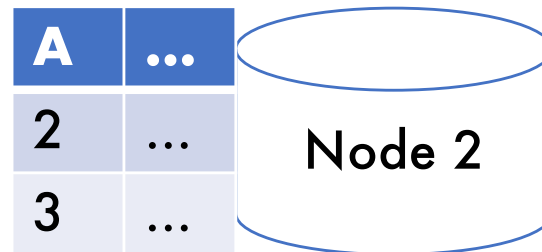
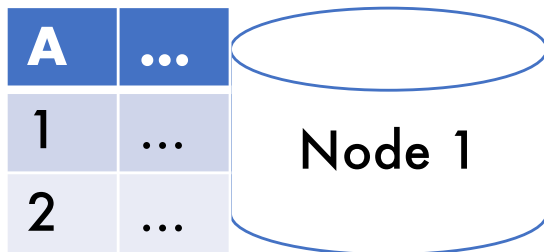
- Hashing data to nodes is very good when the attribute chosen better approximates a uniform distribution
- Keep in mind: Certain nodes will become bottlenecks if a poorly chosen attribute is hashed

Parallel Selection

Assume:

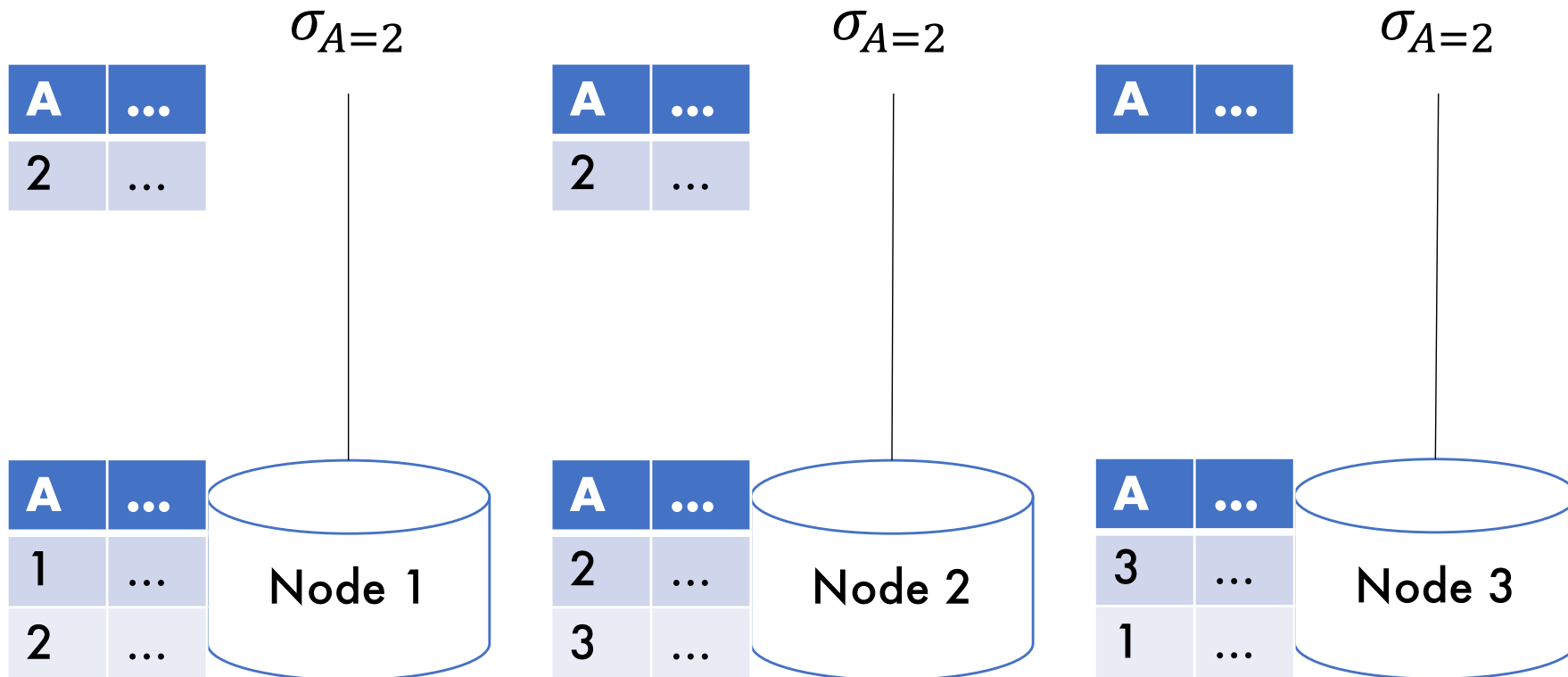
R is block partitioned

```
SELECT *  
FROM R  
WHERE A = 2
```



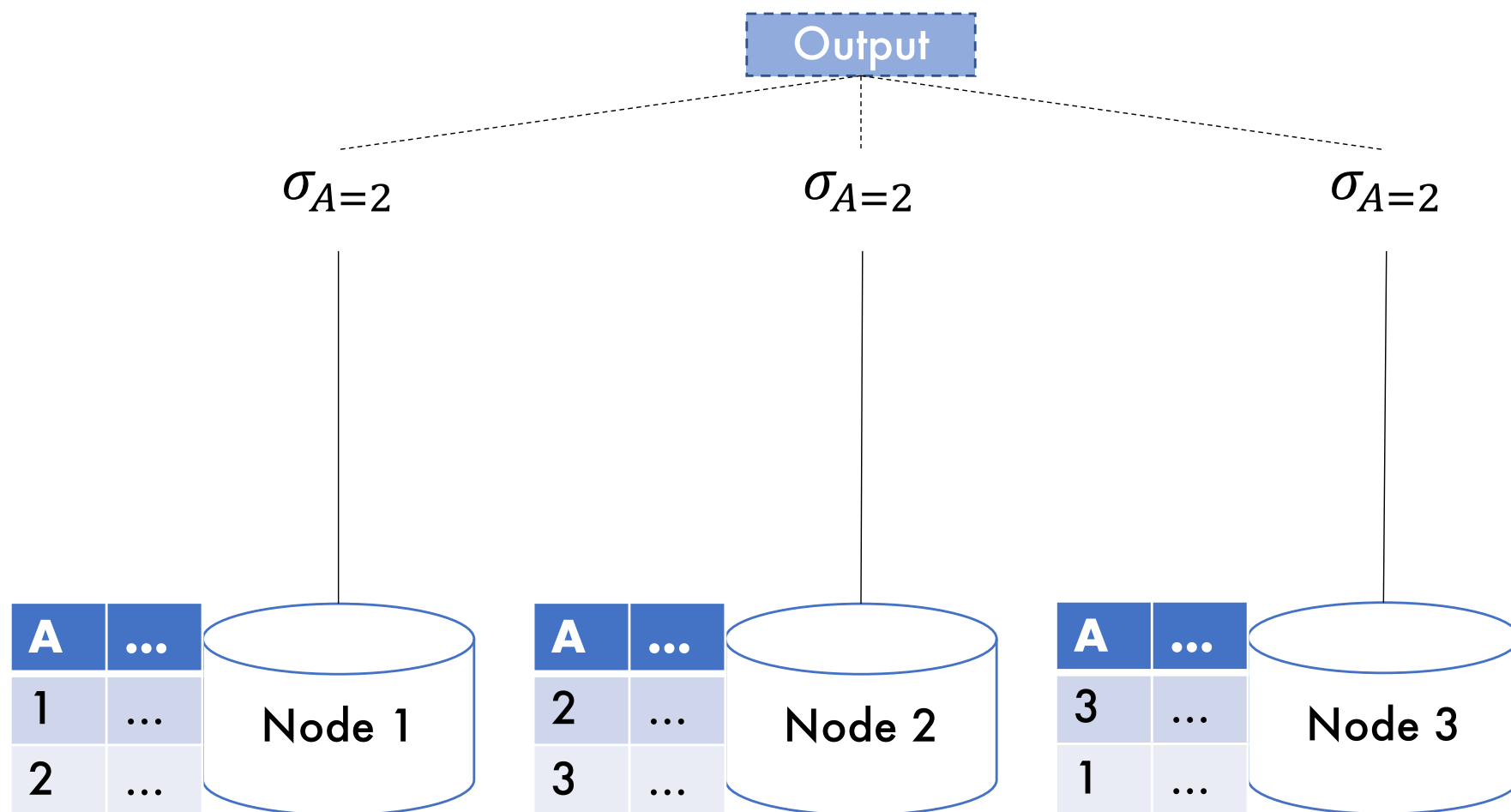
Parallel Selection

```
SELECT *  
FROM R  
WHERE A = 2
```



Implicit Union

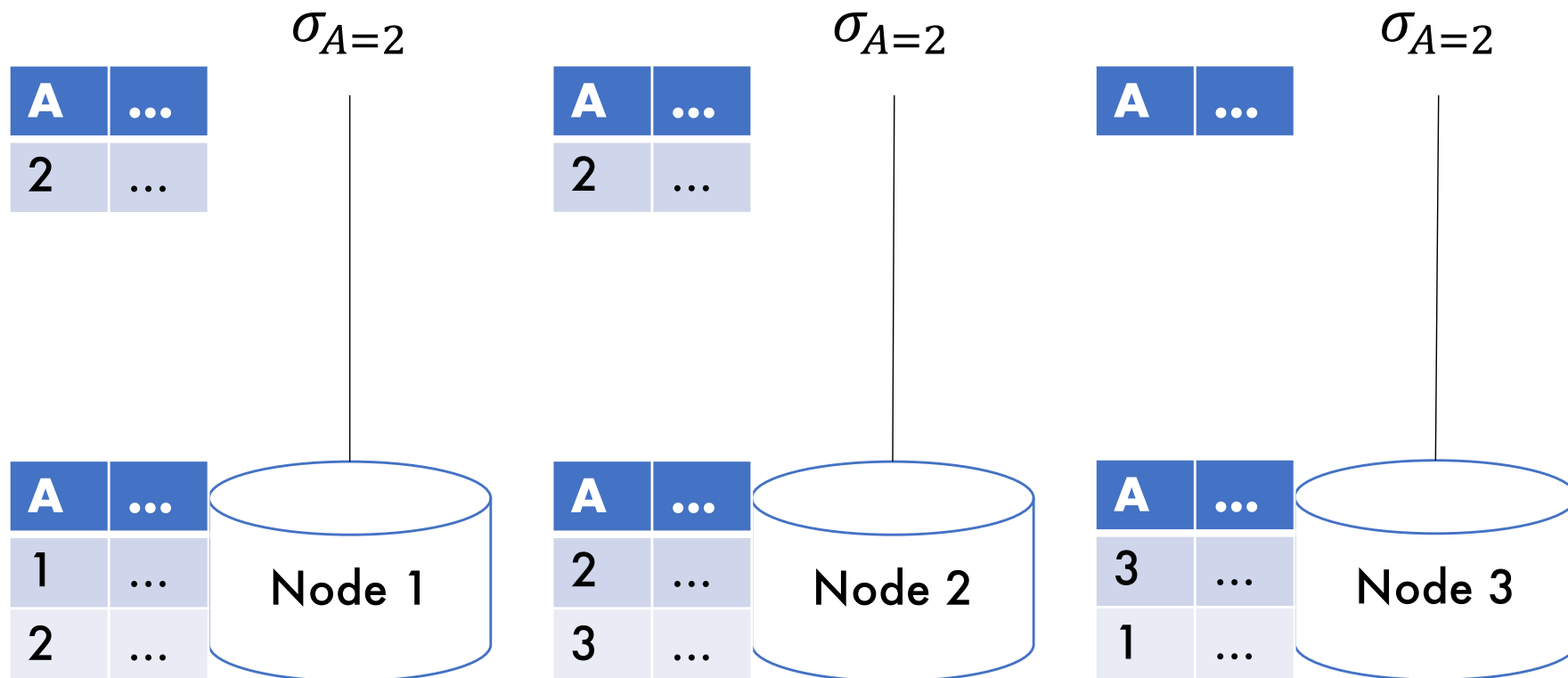
Parallel query plans implicitly union at the end



Parallel Selection

Data-parallel!

```
SELECT *  
FROM R  
WHERE A = 2
```



Parallel Selection

Compute $\sigma_{A=v}(R)$, or $\sigma_{v1 < A < v2}(R)$

- On a conventional database: cost = **B(R)**

Q: What is the cost on each node for a database with N nodes ?

A:

Parallel Selection

Compute $\sigma_{A=v}(R)$, or $\sigma_{v1 < A < v2}(R)$

- On a conventional database: cost = **B(R)**

Q: What is the cost on each node for a database with N nodes ?

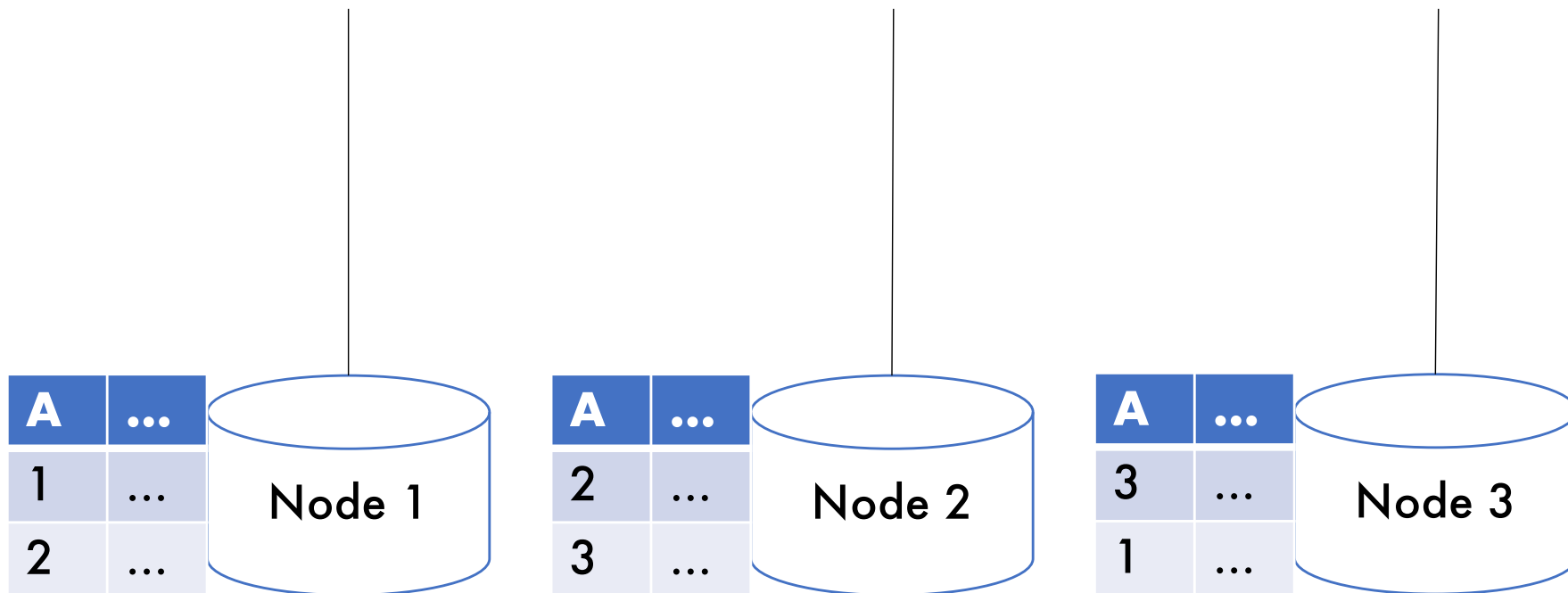
A: $B(R) / N$ block reads on each node

Parallel Selection

What if this query
is not data-parallel?

Assume:
R is block partitioned

```
SELECT *  
FROM R  
.....
```



Partitioned Aggregation

Assume:

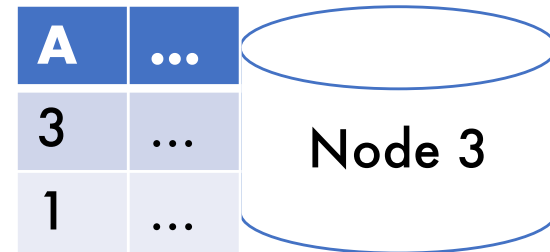
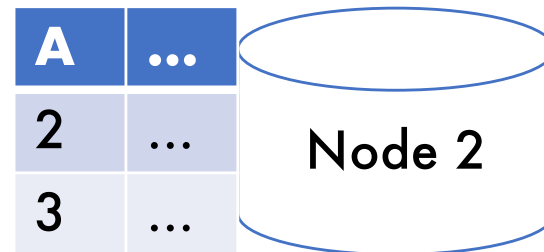
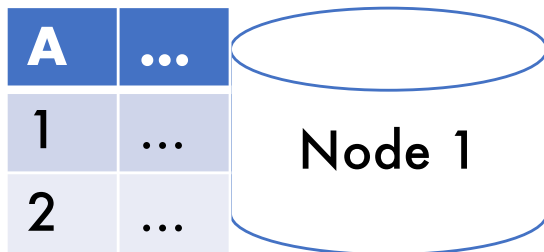
R is block partitioned

```
SELECT *  
FROM R  
GROUP BY R.A
```

$\gamma_{R.A}$

$\gamma_{R.A}$

$\gamma_{R.A}$



Partitioned Aggregation

Assume:
R is block partitioned

```
SELECT *  
FROM R  
GROUP BY R.A
```

A	...
1	...
1	...

$\gamma_{R.A}$

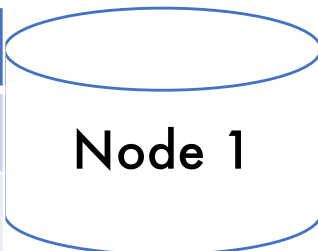
A	...
2	...
2	...

$\gamma_{R.A}$

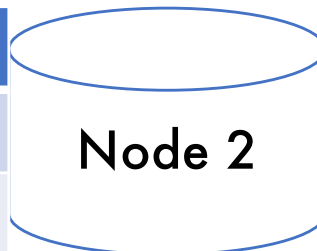
A	...
3	...
3	...

$\gamma_{R.A}$

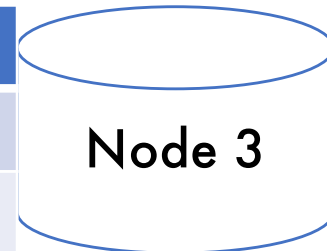
A	...
1	...
2	...



A	...
2	...
3	...



A	...
3	...
1	...



Partitioned Aggregation

1. Hash shuffle tuples

Assume:
R is block partitioned

```
SELECT *  
FROM R  
GROUP BY R.A
```

A	...
1	...
1	...

$\gamma_{R.A}$

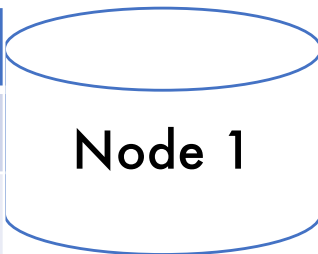
A	...
2	...
2	...

$\gamma_{R.A}$

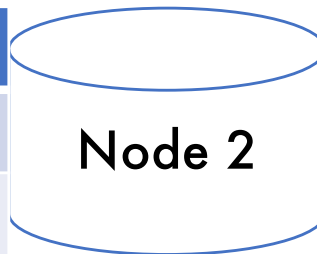
A	...
3	...
3	...

$\gamma_{R.A}$

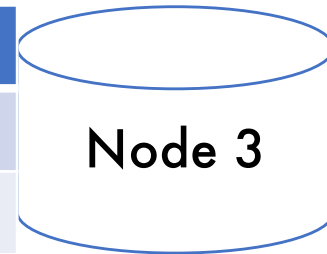
A	...
1	...
2	...



A	...
2	...
3	...



A	...
3	...
1	...

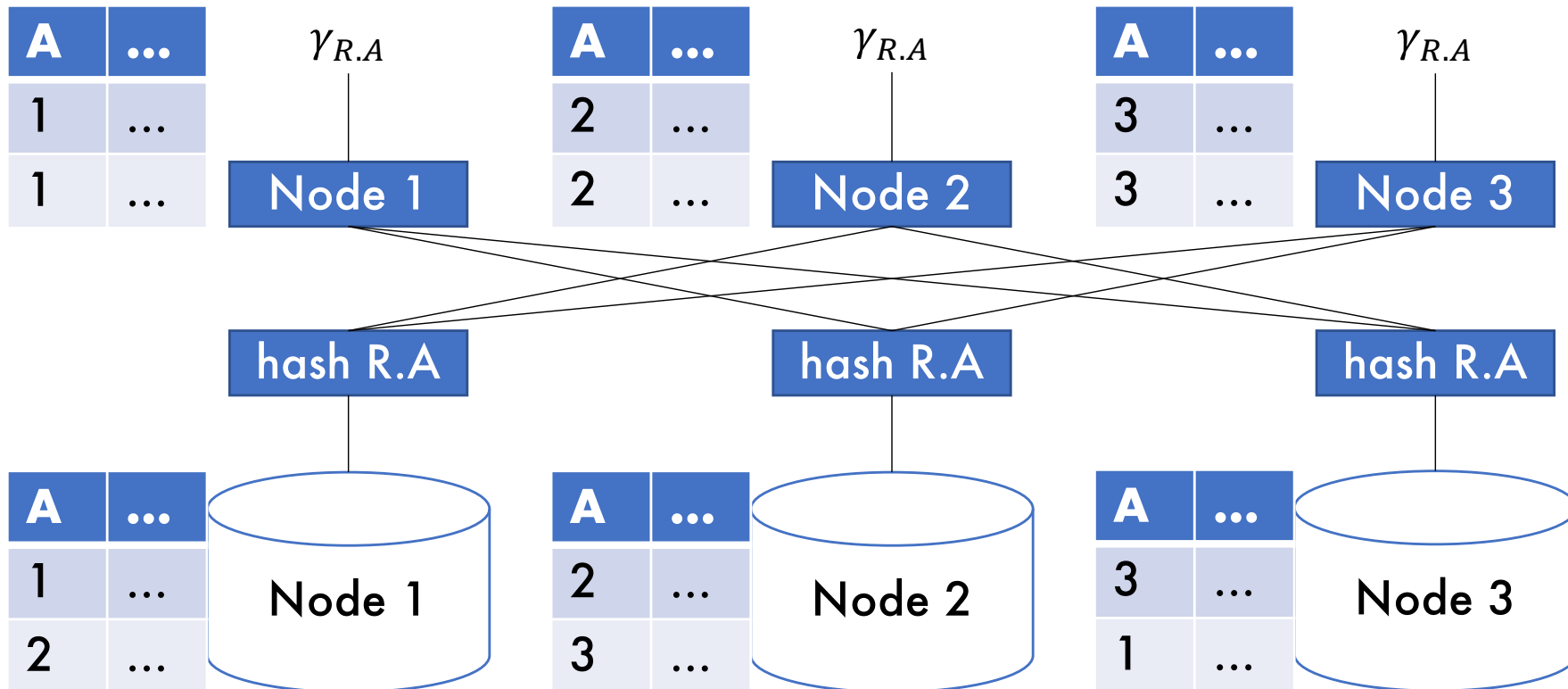


Partitioned Aggregation

1. Hash shuffle tuples

Assume:
R is block partitioned

```
SELECT *  
FROM R  
GROUP BY R.A
```

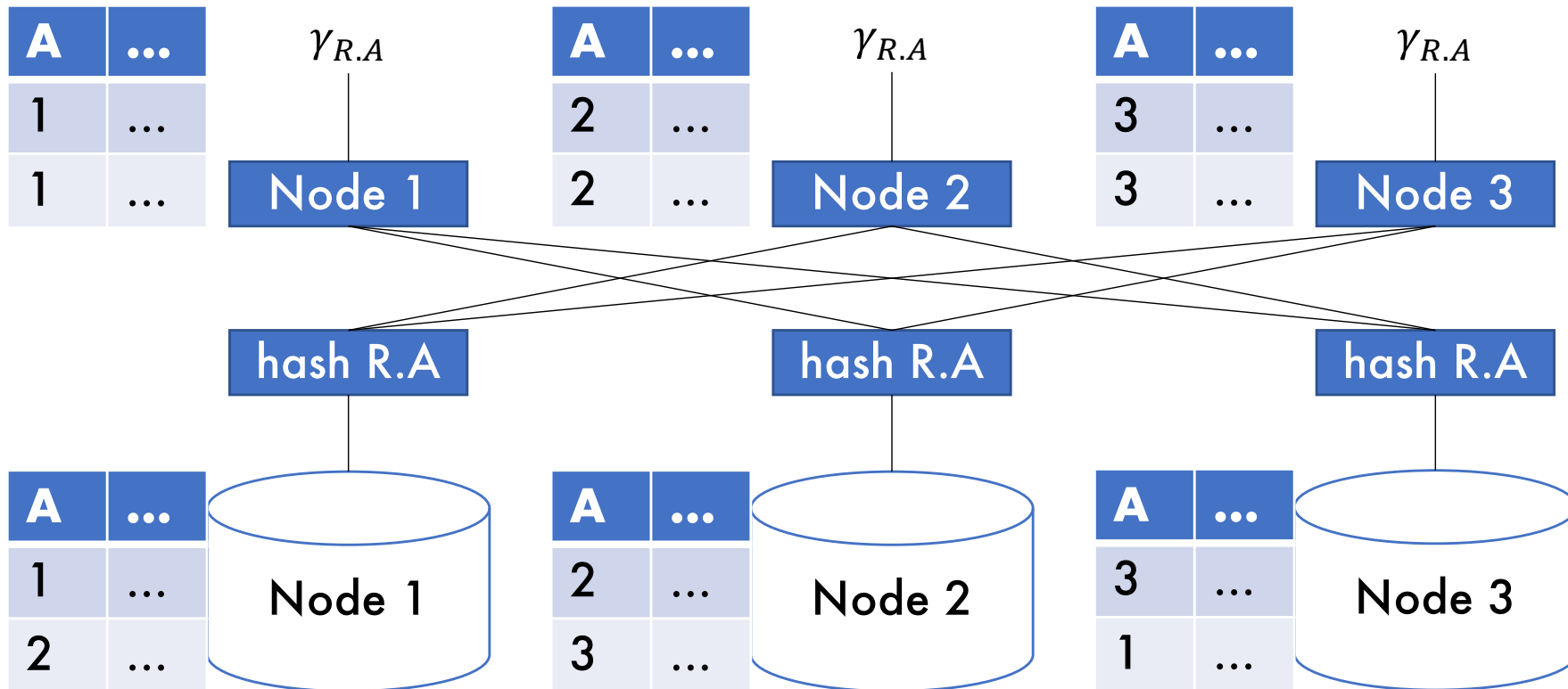


Partitioned Aggregation

1. Hash shuffle tuples
2. Local aggregation

Assume:
R is block partitioned

```
SELECT *  
FROM R  
GROUP BY R.A
```

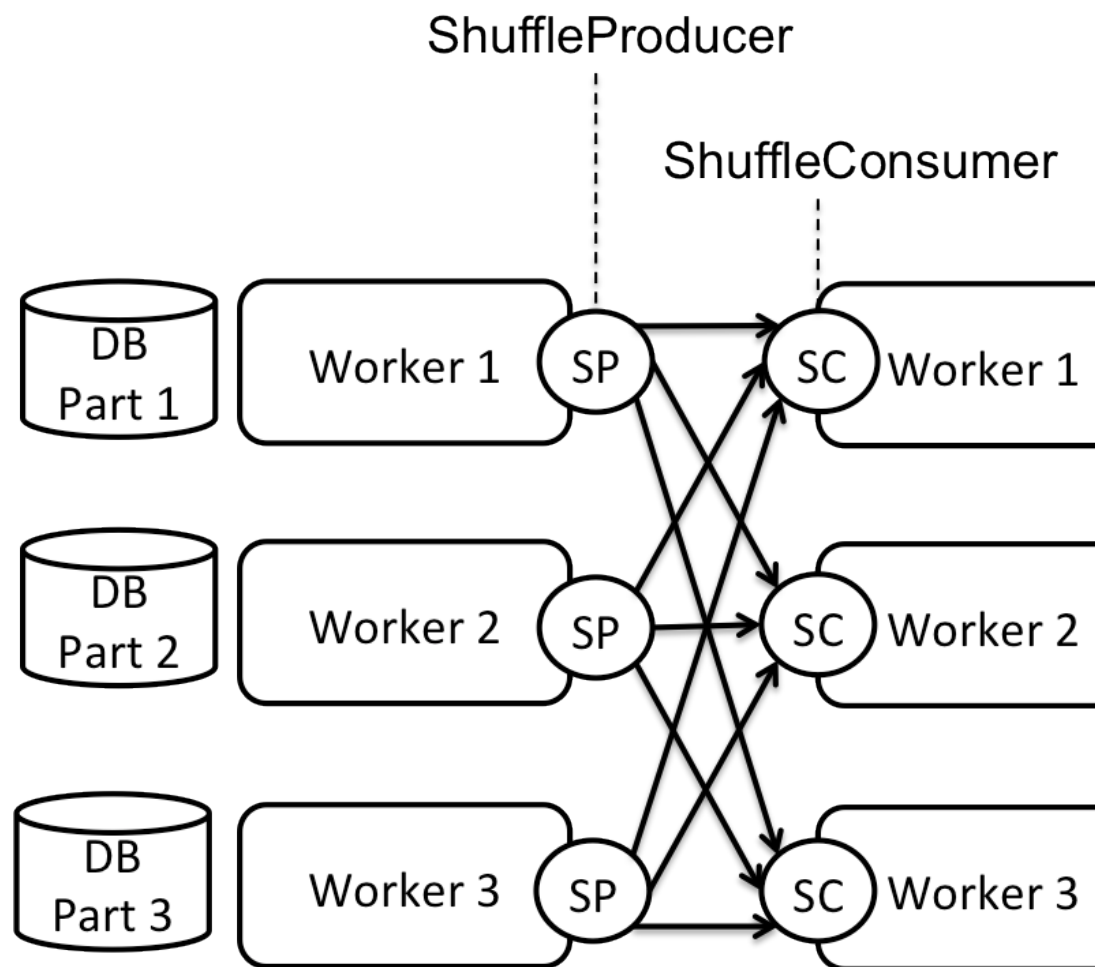


Parallel Query Evaluation

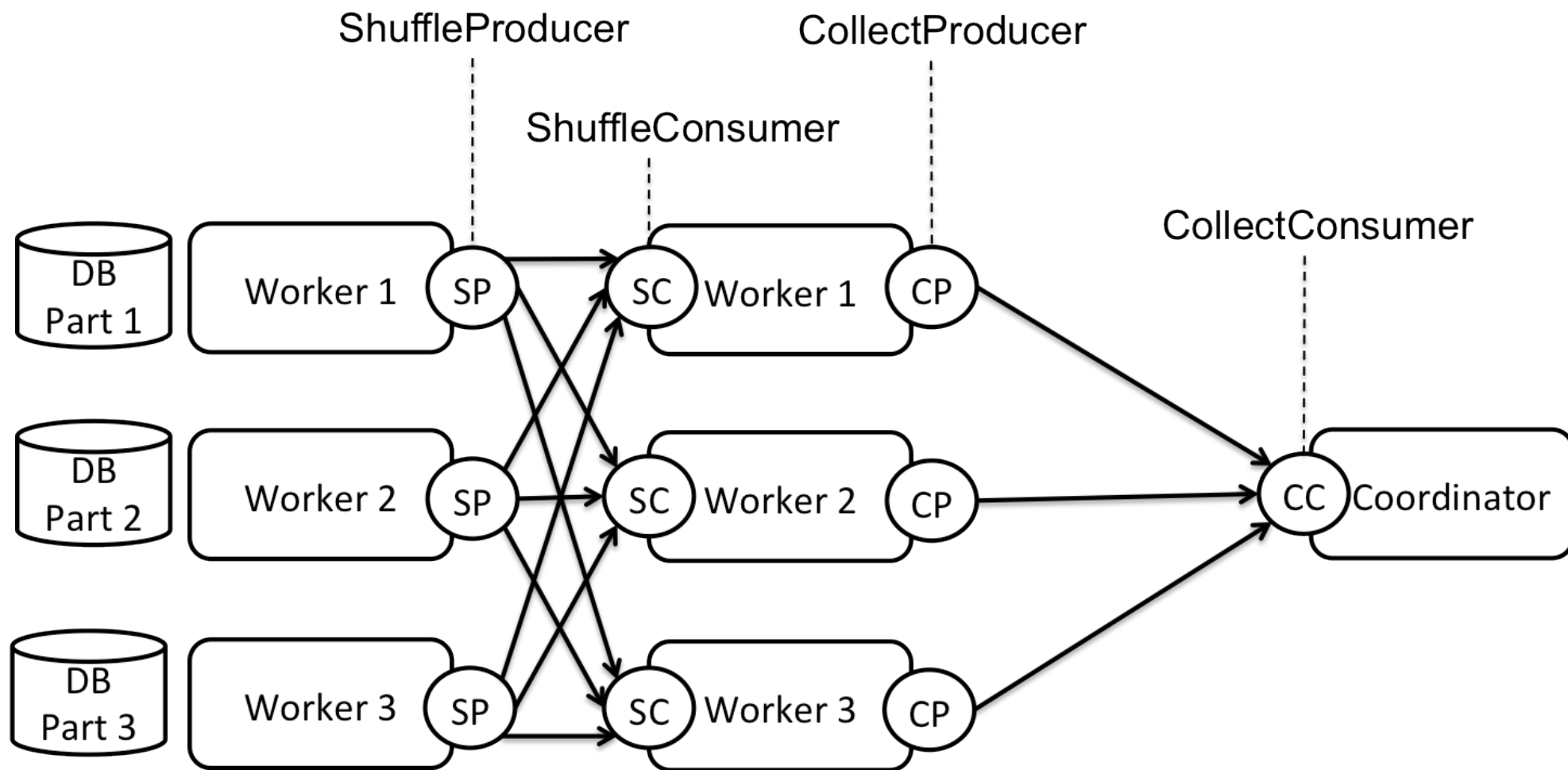
New operator: **Shuffle**

- Serves to re-shuffle data between processes
 - Handles data routing, buffering, and flow control
- Two parts: **ShuffleProducer** and **ShuffleConsumer**
- Producer:
 - Pulls data from child 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

Parallel Query Execution



Parallel Query Execution



Partitioned Hash Equijoin Algorithm

1. Hash shuffle tuples on join attributes

Assume:

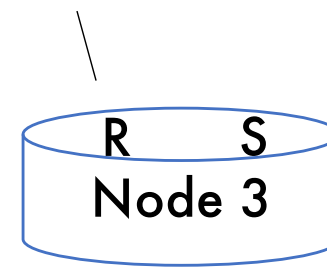
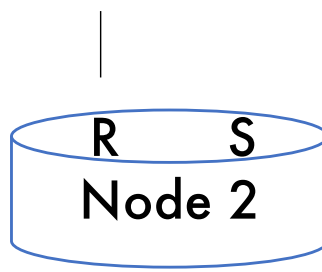
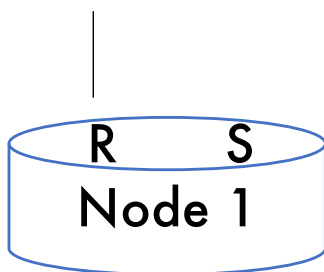
R and S are block partitioned

```
SELECT *  
FROM R, S  
WHERE R.A = S.A
```

$\bowtie_{R.A=S.A}$

$\bowtie_{R.A=S.A}$

$\bowtie_{R.A=S.A}$



Partitioned Hash Equijoin Algorithm

1. Hash shuffle tuples on join attributes

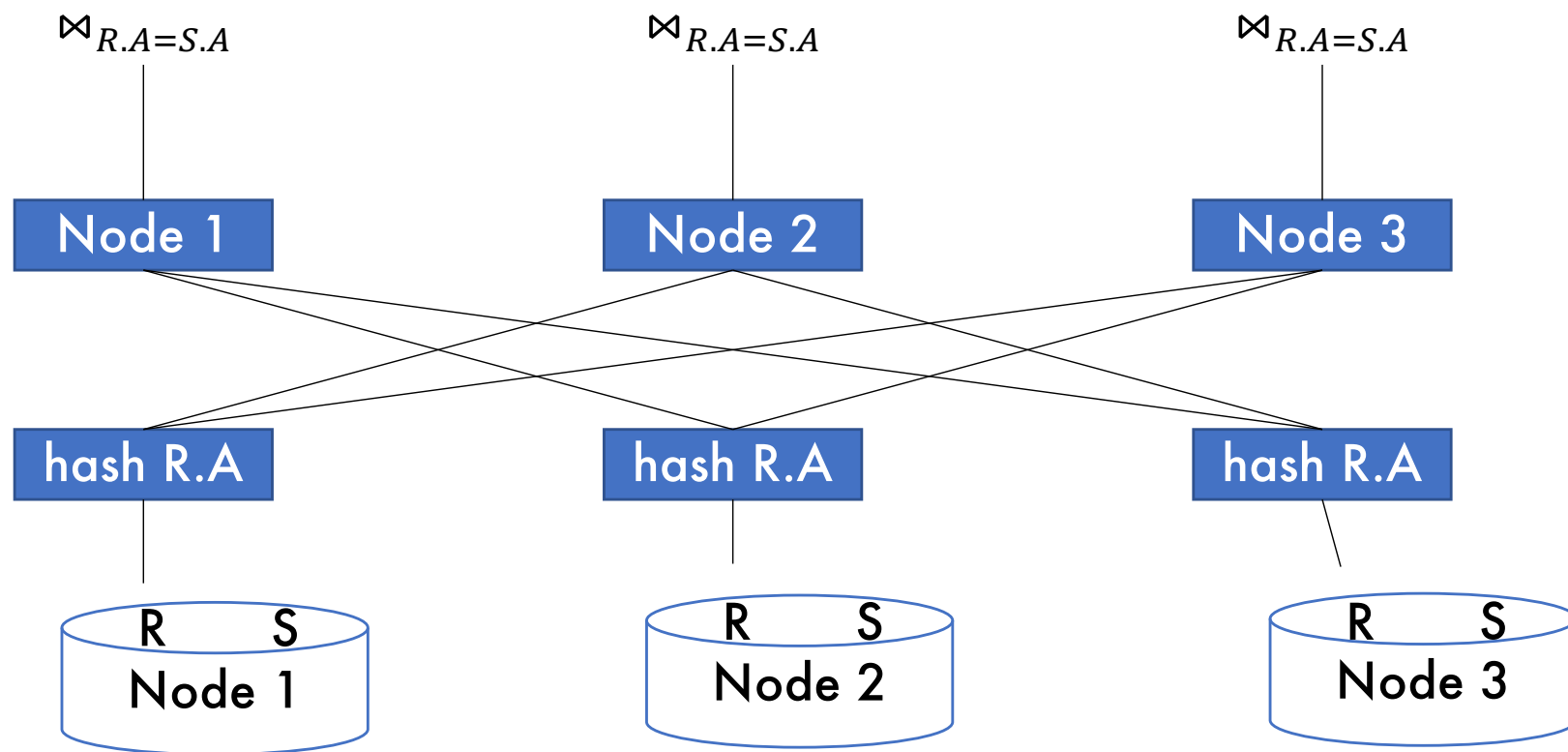
Assume:

R and S are block partitioned

SELECT *

FROM R, S

WHERE R.A = S.A



Partitioned Hash Equijoin Algorithm

1. Hash shuffle tuples on join attributes

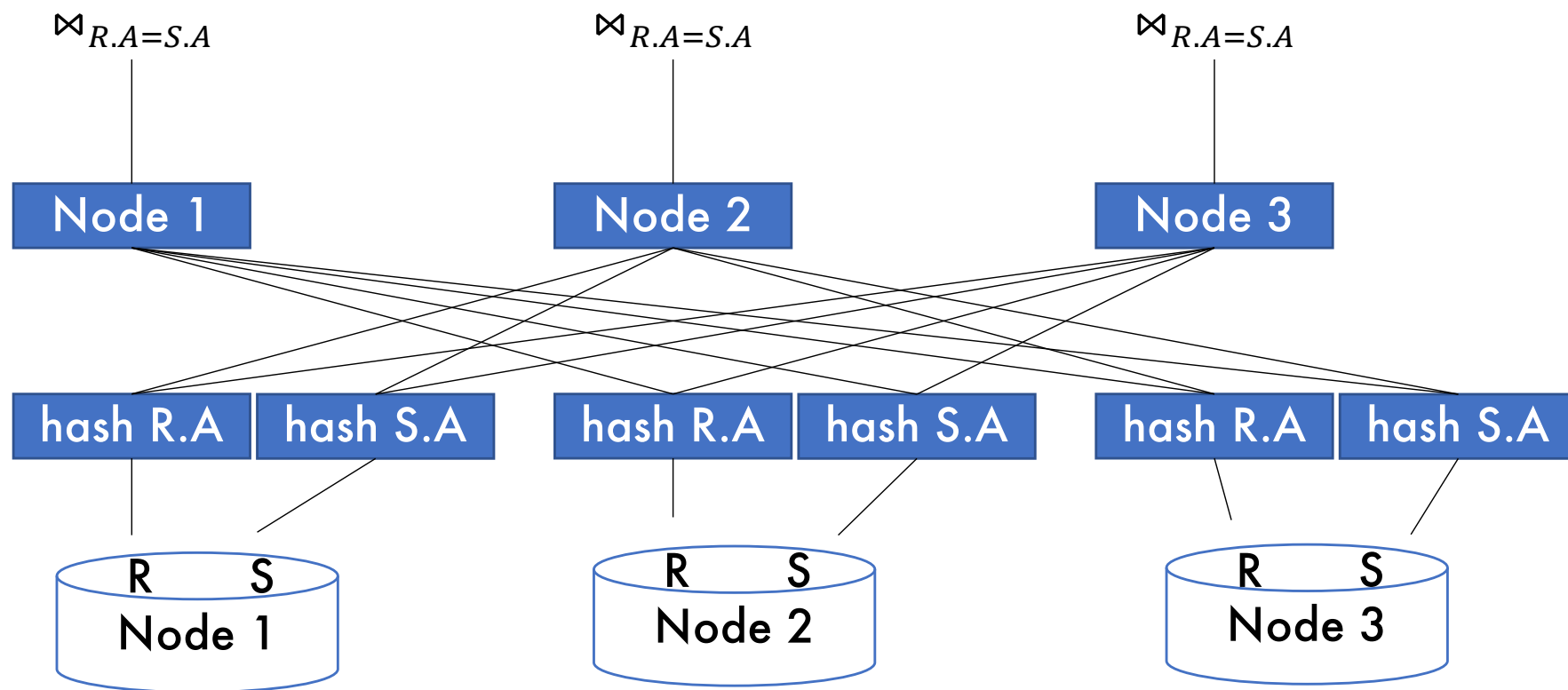
Assume:

R and S are block partitioned

SELECT *

FROM R, S

WHERE R.A = S.A



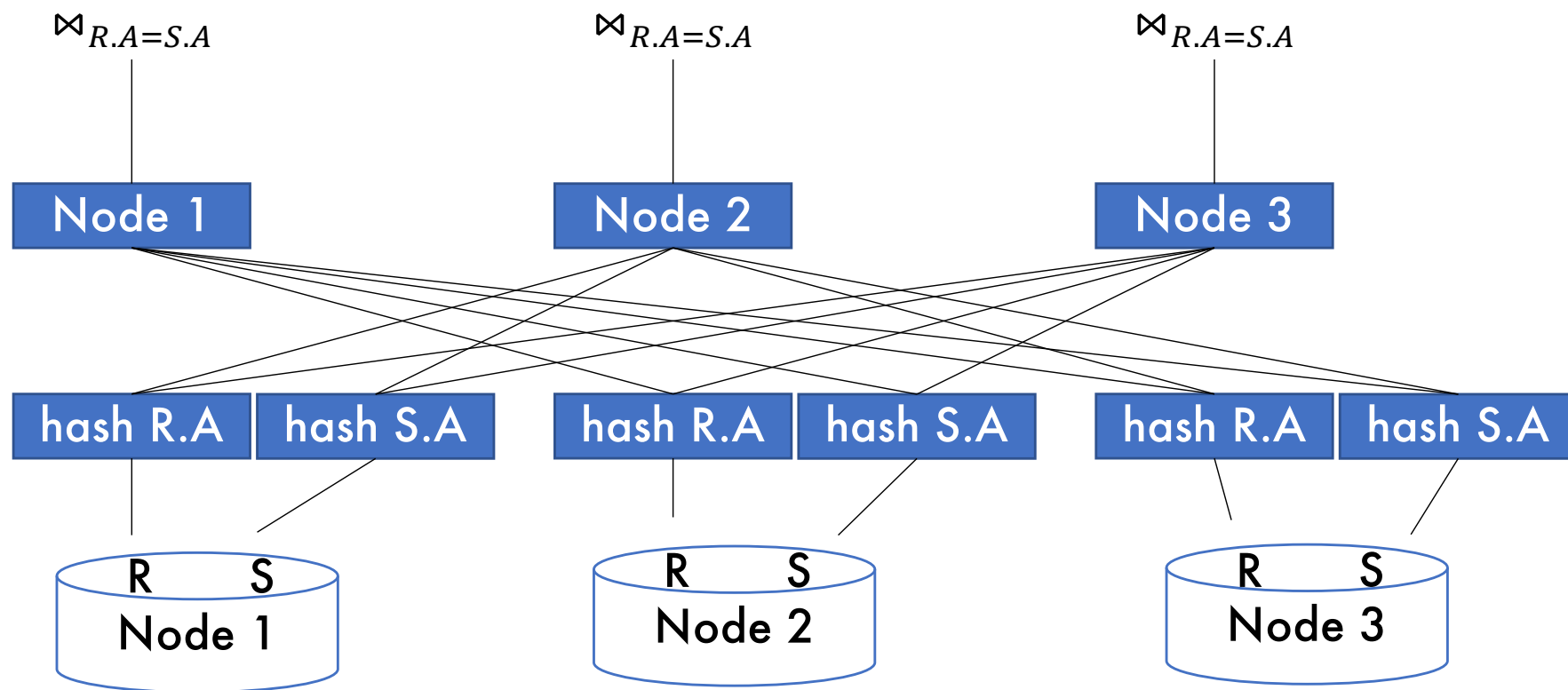
Partitioned Hash Equijoin Algorithm

1. Hash shuffle tuples on join attributes
2. Local join

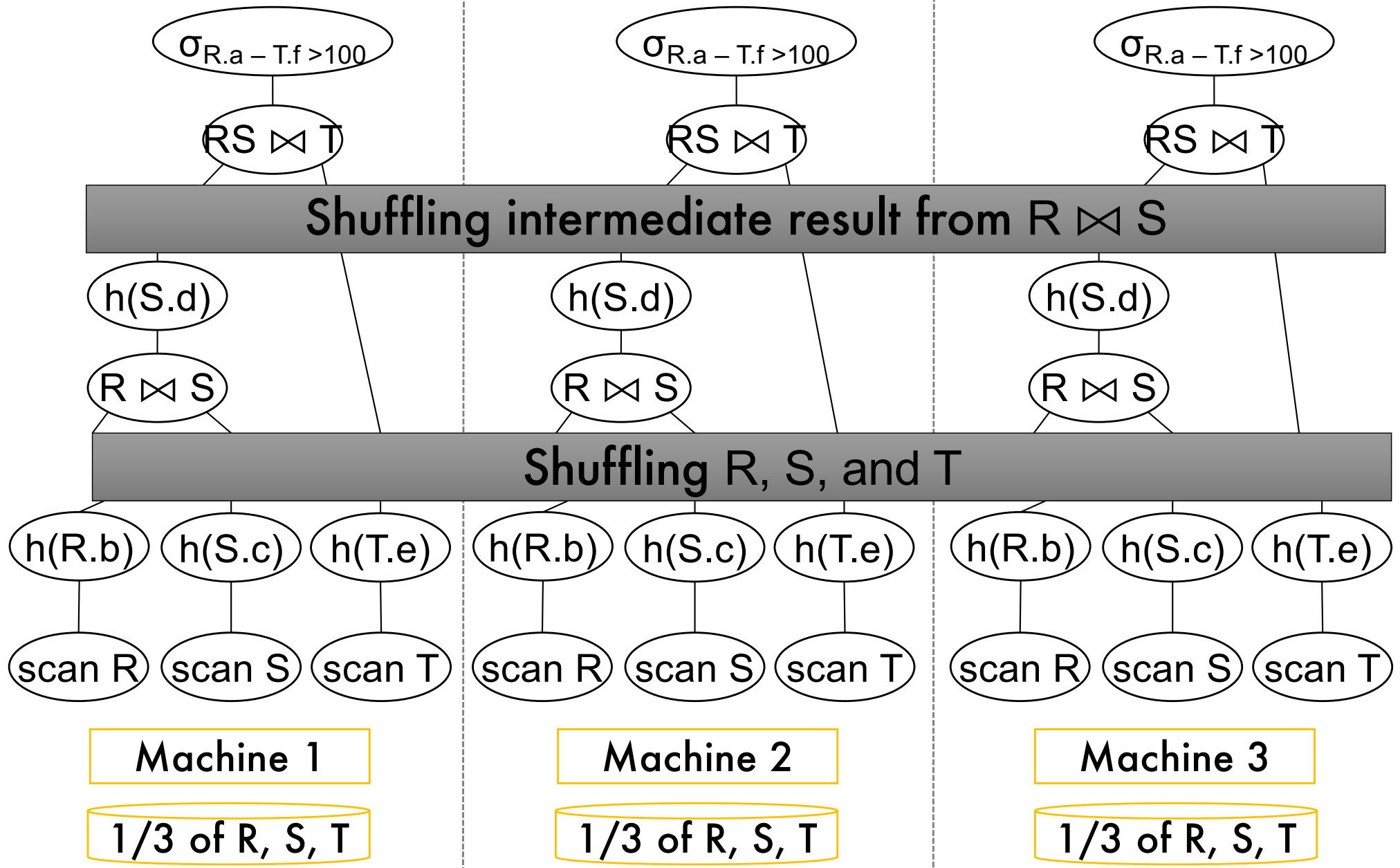
Assume:

R and S are block partitioned

```
SELECT *  
FROM R, S  
WHERE R.A = S.A
```



Multiple Shuffles



Summary

- With one new operator, we've made SimpleDB an OLAP-ready parallel DBMS!
- Next lecture:
 - Skew handling
 - Algorithm refinements

Speedup and Scaleup

- Consider:
 - Query: $\gamma_{A, \text{sum}(C)}(R)$
 - 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: $\gamma_{A, \text{sum}(C)}(R)$
 - 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 $1/2$ as many chunks)
- If we double both P and the size of R , what is the new running time?

Speedup and Scaleup

- Consider:
 - Query: $\gamma_{A, \text{sum}(C)}(R)$
 - 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 $1/2$ 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)

Basic Parallel GroupBy

Can we do better?

- Sum?
- Count?
- Avg?
- Max?
- Median?

Basic Parallel GroupBy

Can we do better?

- Sum?
- Count?
- Avg?
- Max?
- Median?

Distributive	Algebraic	Holistic
$\text{sum}(a_1 + a_2 + \dots + a_9) =$ $\text{sum}(\text{sum}(a_1 + a_2 + a_3) +$ $\text{sum}(a_4 + a_5 + a_6) +$ $\text{sum}(a_7 + a_8 + a_9))$	$\text{avg}(B) =$ $\text{sum}(B) / \text{count}(B)$	$\text{median}(B)$

Basic Parallel GroupBy

Can we do better?

- Sum?
- Count?
- Avg?
- Max?
- Median?

Distributive	Algebraic	Holistic
$\text{sum}(a_1 + a_2 + \dots + a_9) = \text{sum}(\text{sum}(a_1 + a_2 + a_3) + \text{sum}(a_4 + a_5 + a_6) + \text{sum}(a_7 + a_8 + a_9))$	$\text{avg}(B) = \text{sum}(B) / \text{count}(B)$	$\text{median}(B)$

YES

- Compute partial aggregates before shuffling

Basic Parallel GroupBy

Can we do better?

- Sum?
- Count?
- Avg?
- Max?
- Median?

Distributive	Algebraic	Holistic
$\text{sum}(a_1 + a_2 + \dots + a_9) =$ $\text{sum}(\text{sum}(a_1 + a_2 + a_3) +$ $\text{sum}(a_4 + a_5 + a_6) +$ $\text{sum}(a_7 + a_8 + a_9))$	$\text{avg}(B) =$ $\text{sum}(B) / \text{count}(B)$	$\text{median}(B)$

YES

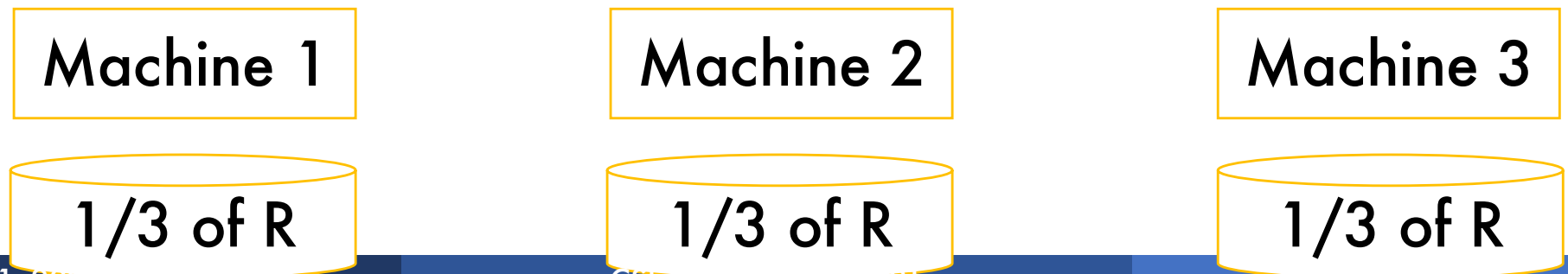
- Compute partial aggregates before shuffling

MapReduce implements this as “Combiners”

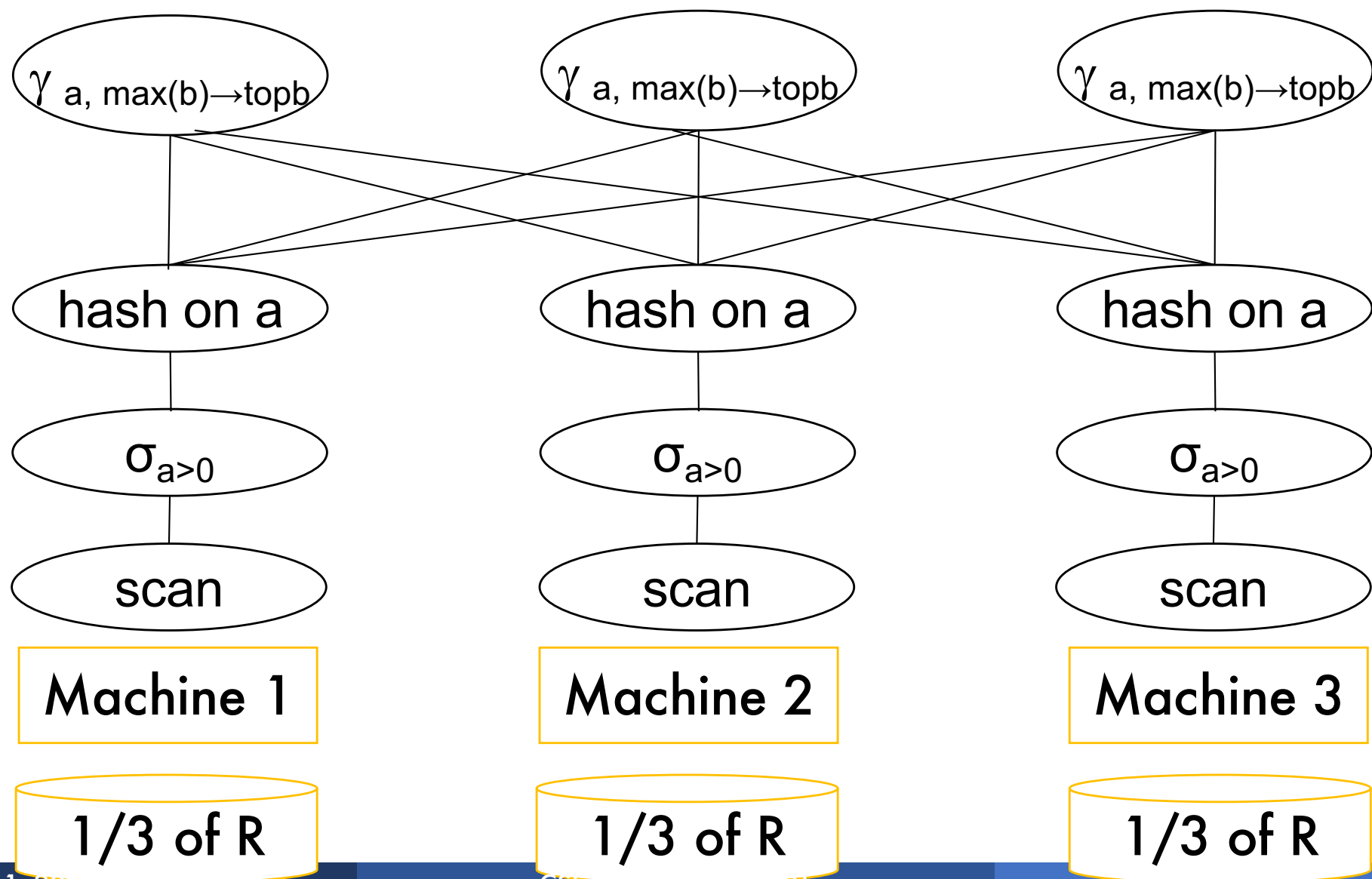
Exercise (www.draw.io is fast!)

Example Query with Group By

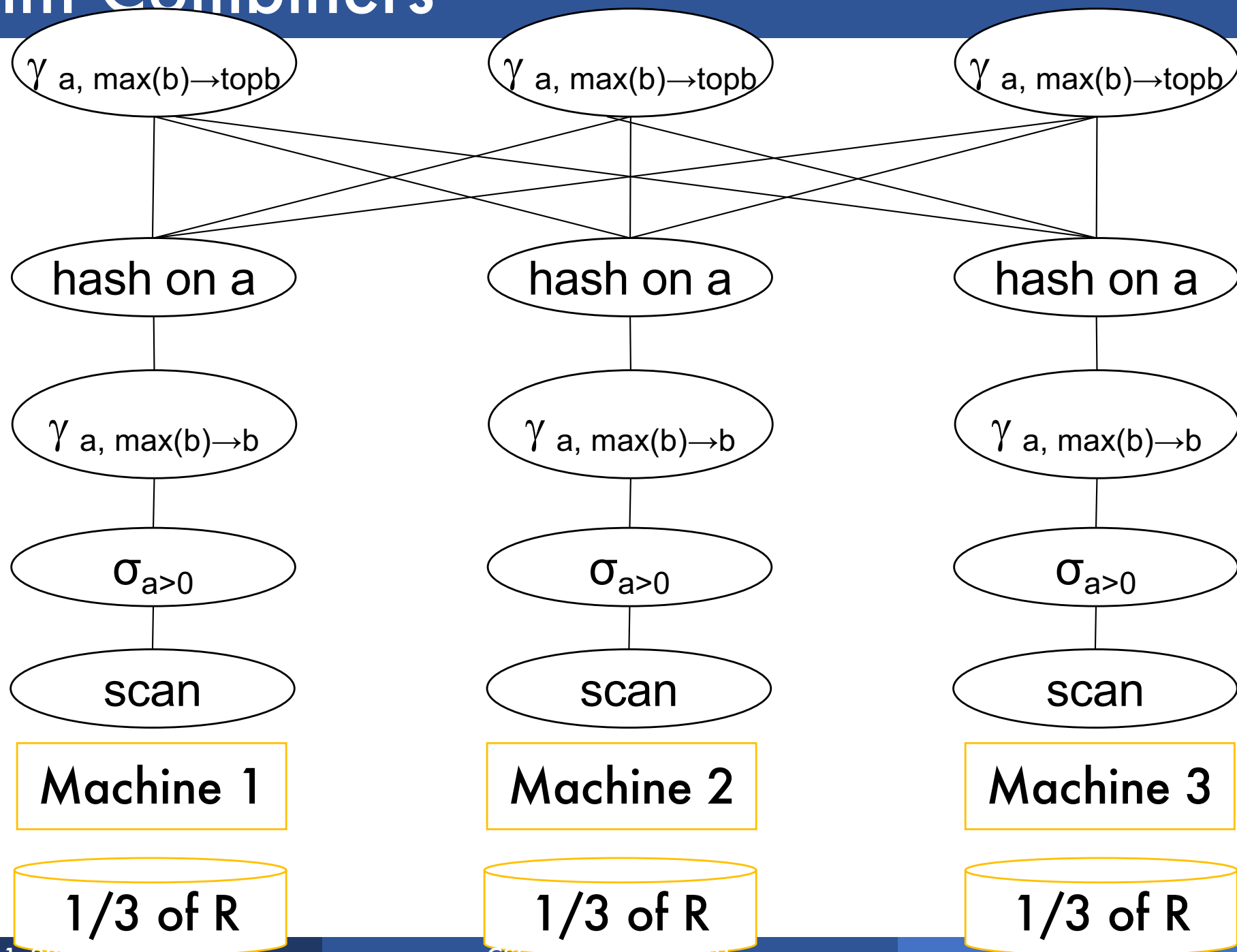
```
SELECT a, max(b) as topb  
FROM R WHERE a > 0  
GROUP BY a
```



Without Combiners



With Combiners



Parallel Join: $R \bowtie_{A=B} S$

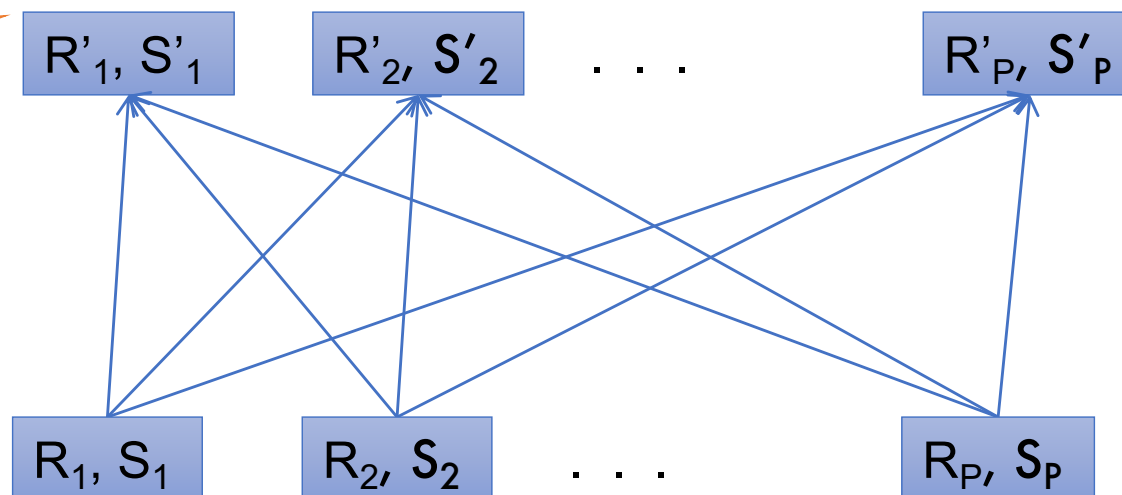
- **Data:** $R(\underline{K1}, A, C), S(\underline{K2}, B, D)$
- **Query:** $R(\underline{K1}, A, C) \bowtie S(\underline{K2}, B, D)$

Parallel Join: $R \bowtie_{A=B} S$

- **Data:** $R(\underline{K1}, A, C), S(\underline{K2}, B, D)$
- **Query:** $R(\underline{K1}, A, C) \bowtie S(\underline{K2}, B, D)$

Each server computes the join locally

Reshuffle R on R.A and S on S.B



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

Parallel Join: $R \bowtie_{A=B} S$

▪ Step 1

- Every server holding any chunk of R partitions its chunk using a hash function $h(t.A) \bmod P$
- Every server holding any chunk of S partitions its chunk using a hash function $h(t.B) \bmod P$

▪ Step 2:

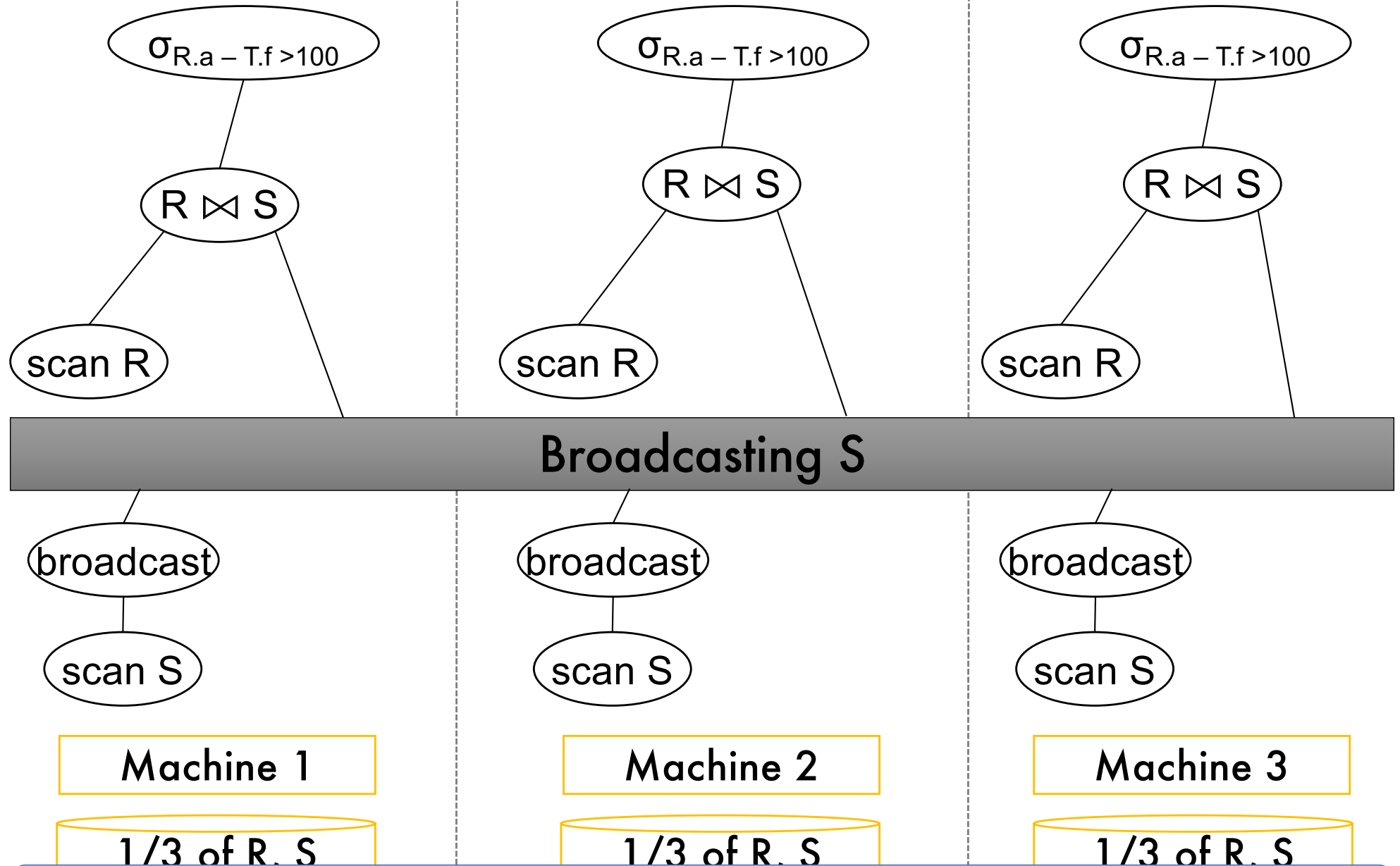
- Each server computes the join of its local fragment of R with its local fragment of S

Optimization for Small Relations

When joining R and S

- If $|R| \gg |S|$
 - Leave R where it is
 - Replicate entire S relation across nodes
- Also called a **small join** or a **broadcast join**

Broadcast Join Example



Can save huge network costs!

Justin Biebers Re-visited

Skew:

- Some partitions get more **input** tuples than others

Reasons:

- Range-partition instead of hash
 - Some values are very popular:
 - Heavy hitters values; e.g. 'Justin Bieber'
 - Selection before join with different selectivities
-
- Some partitions generate more **output** tuples than others

Some Skew Handling Techniques

If using range partition:

- Ensure each range gets same number of tuples
- E.g.: $\{1, 1, 1, 2, 3, 4, 5, 6\} \rightarrow [1,2]$ and $[3,6]$
- Eq-depth v.s. eq-width histograms

Some Skew Handling Techniques

Create more partitions than nodes

- And be smart about scheduling the partitions
 - E.g. One node **ONLY** does Justin Biebers
- Note: MapReduce uses this technique

Some Skew Handling Techniques

Use subset-replicate (a.k.a. “skewedJoin”)

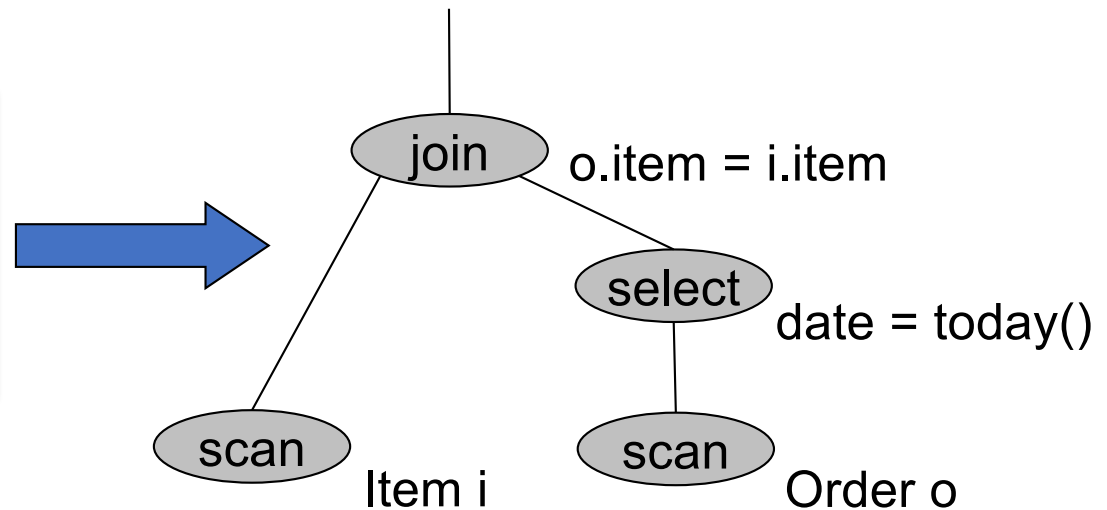
- Given $R \bowtie_{A=B} S$
- Given a heavy hitter value $R.A = 'v'$ (i.e. $'v'$ occurs very many times in R)
- Partition R tuples with value $'v'$ across all nodes e.g. block-partition, or hash on other attributes
- Replicate S tuples with value $'v'$ to all nodes
- R = the build relation
- S = the probe relation

Example: Teradata – Query Execution

Order(oid, item, date), Line(item, ...)

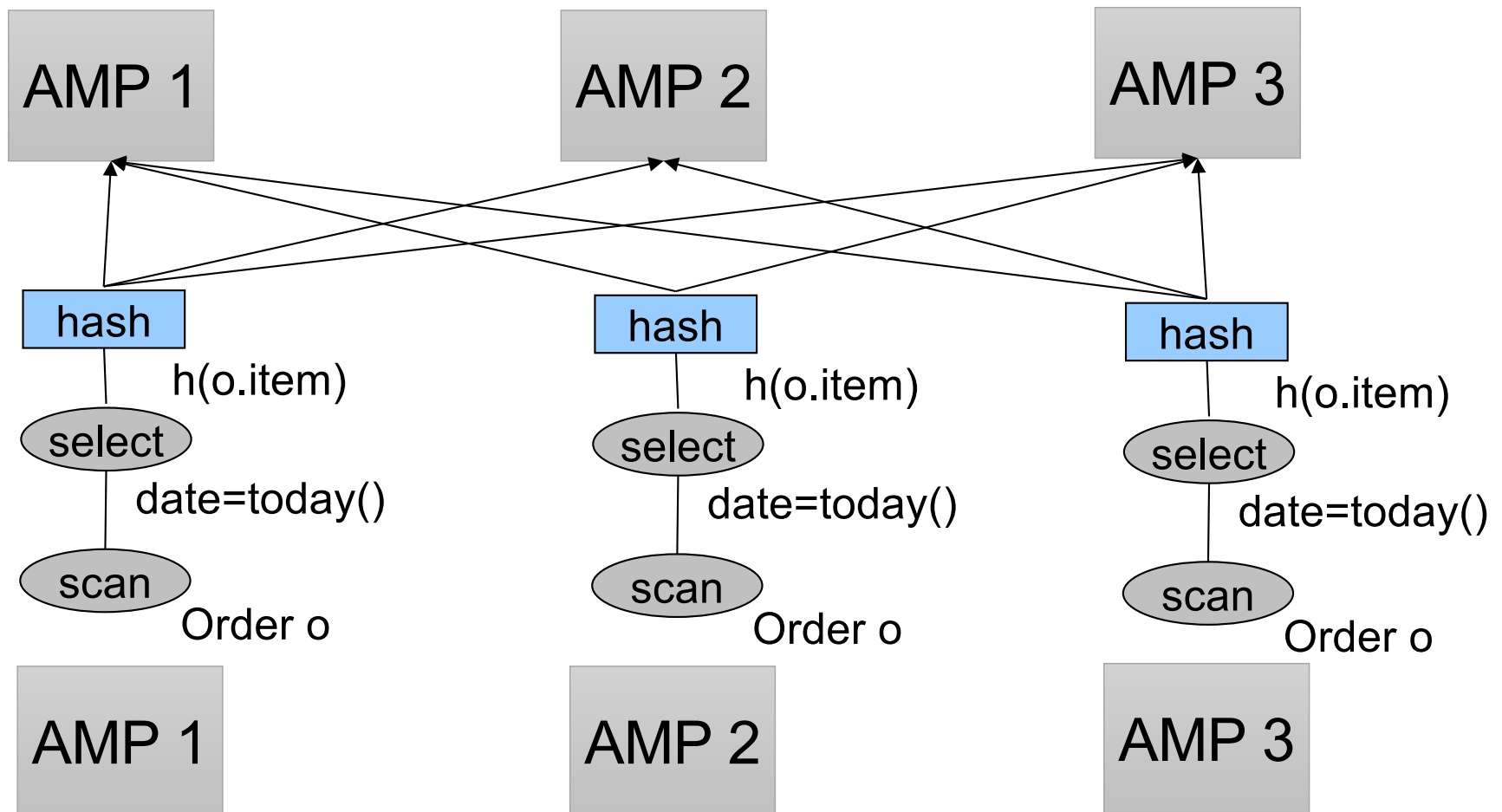
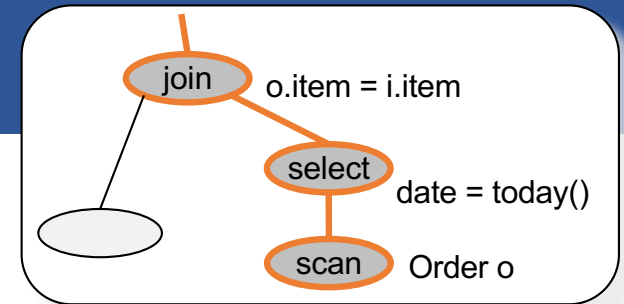
Find all orders from today, along with the items ordered

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



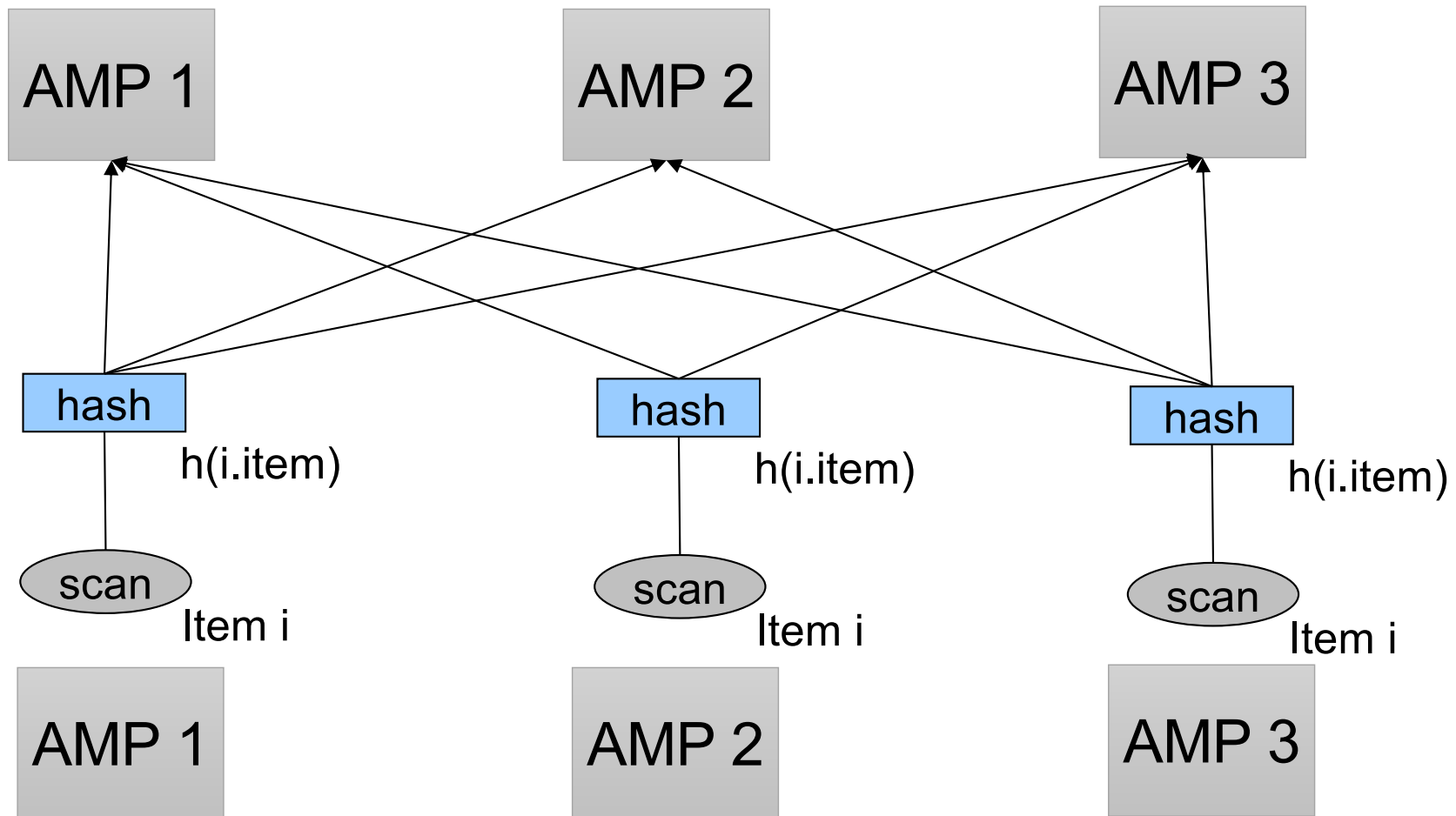
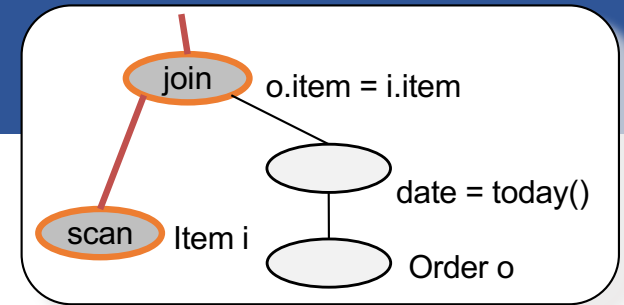
Query Execution

Order(oid, item, date), Line(item, ...)



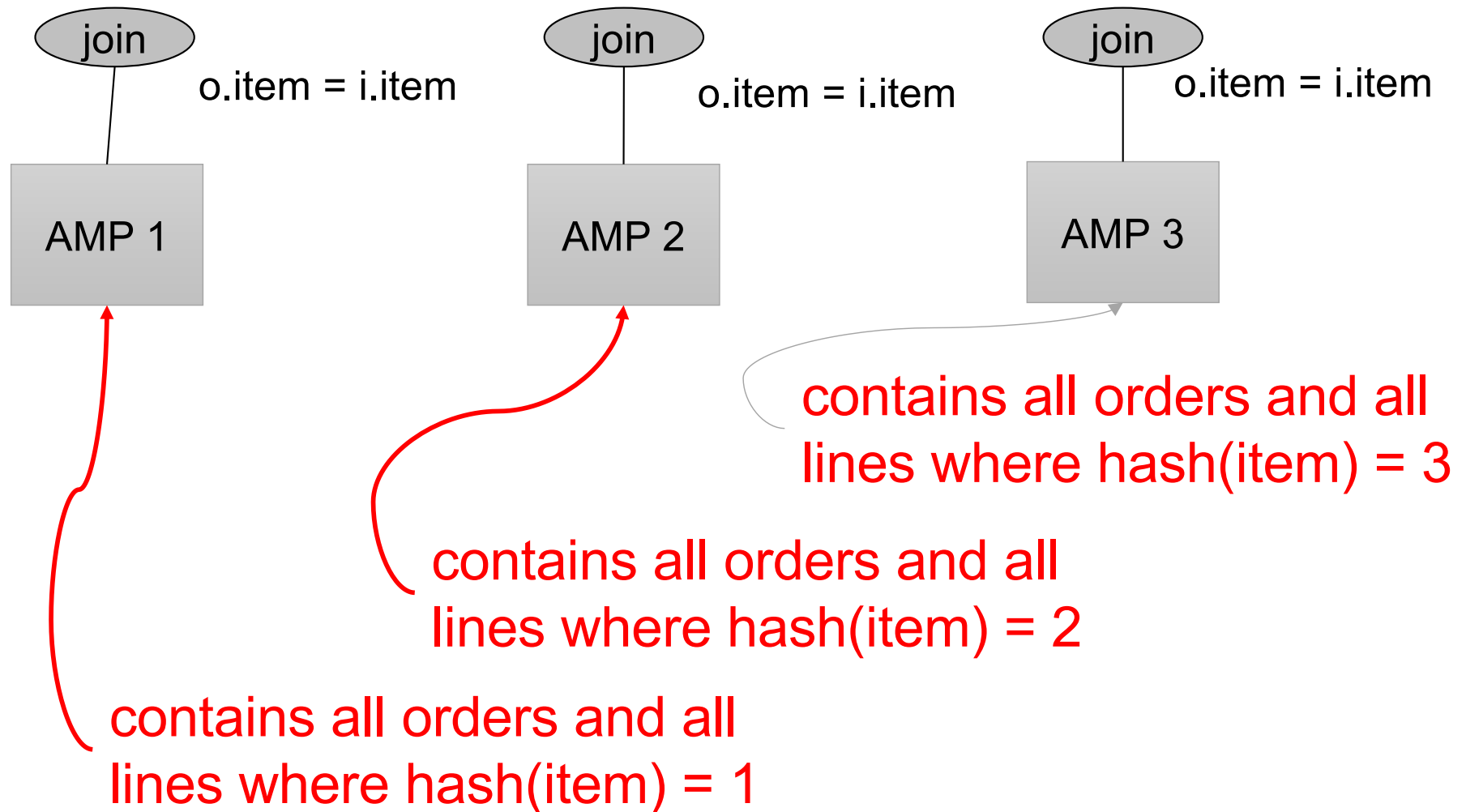
Query Execution

Order(oid, item, date), Line(item, ...)



Query Execution

Order(oid, item, date), Line(item, ...)



Example 2

SELECT *
FROM R, S, T
WHERE R.b = S.c AND S.d = T.e AND (R.a - T.f) > 100

Machine 1

1/3 of R, S, T

Machine 2

1/3 of R, S, T

Machine 3

1/3 of R, S, T

