Optimizing Hive Joins

Overview

Understand how joins are implemented in Hive

Optimize joins which involve large tables

Use semi-joins in place of IN/EXISTS subqueries

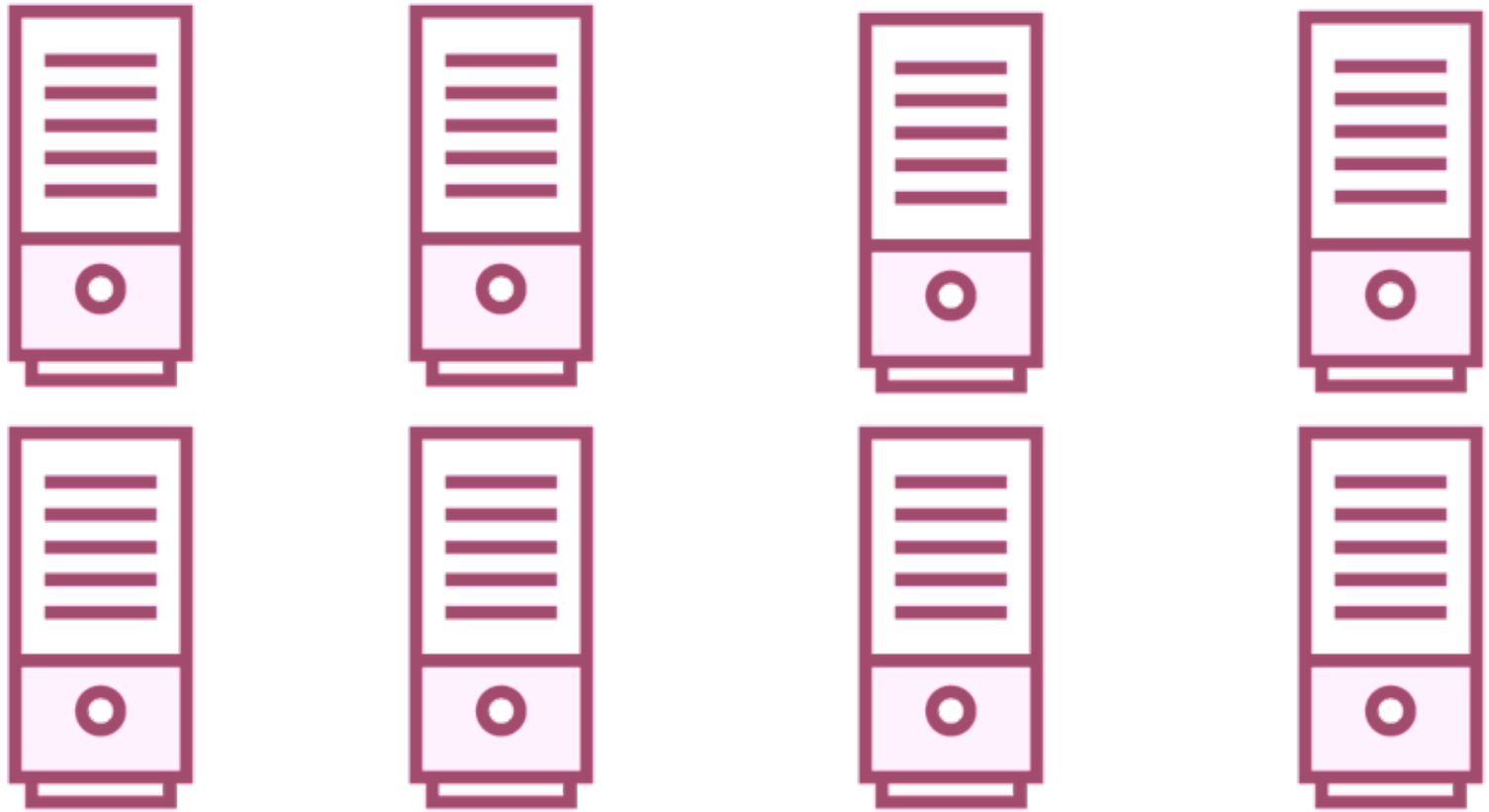
Optimize map-only joins

Join Operations as MapReduce Jobs

Join Operations

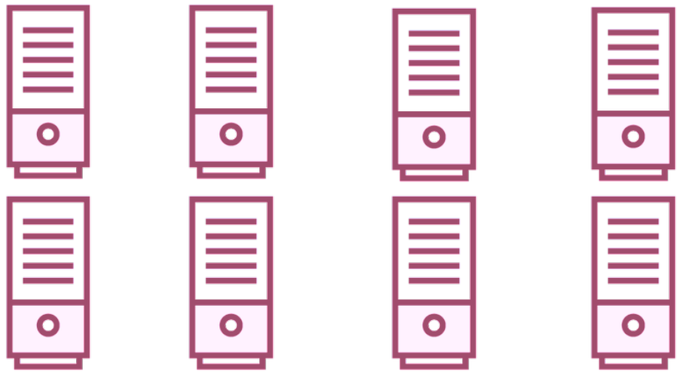
Join operations are MapReduce jobs under the hood

MapReduce



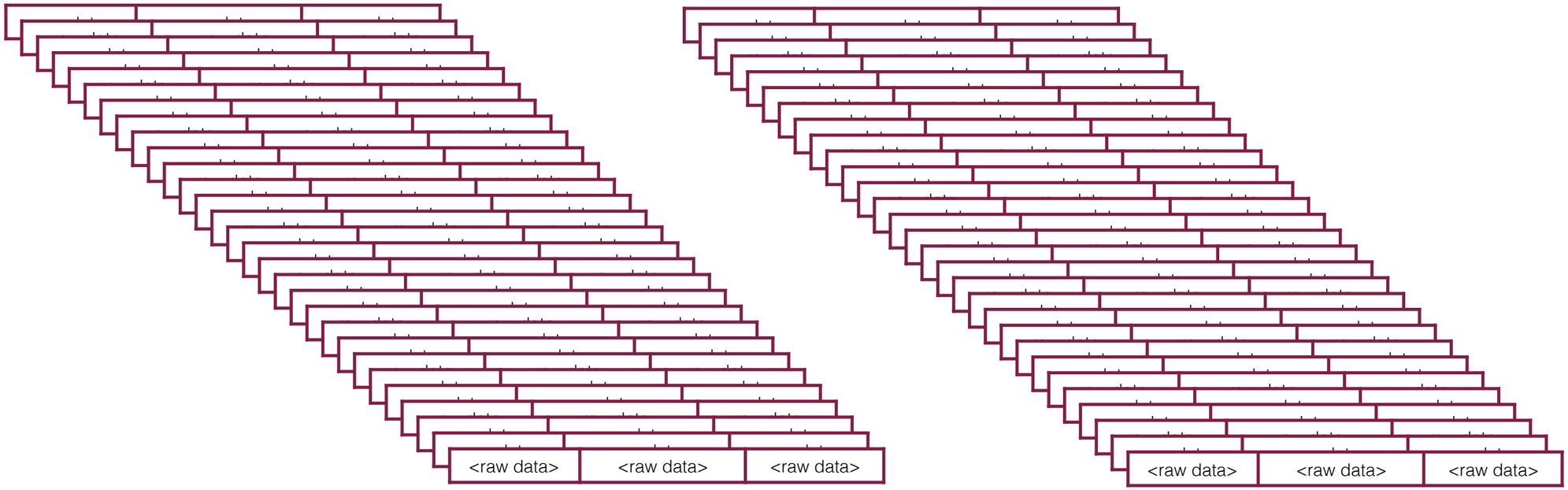
A programming paradigm which runs on a distributed system

MapReduce



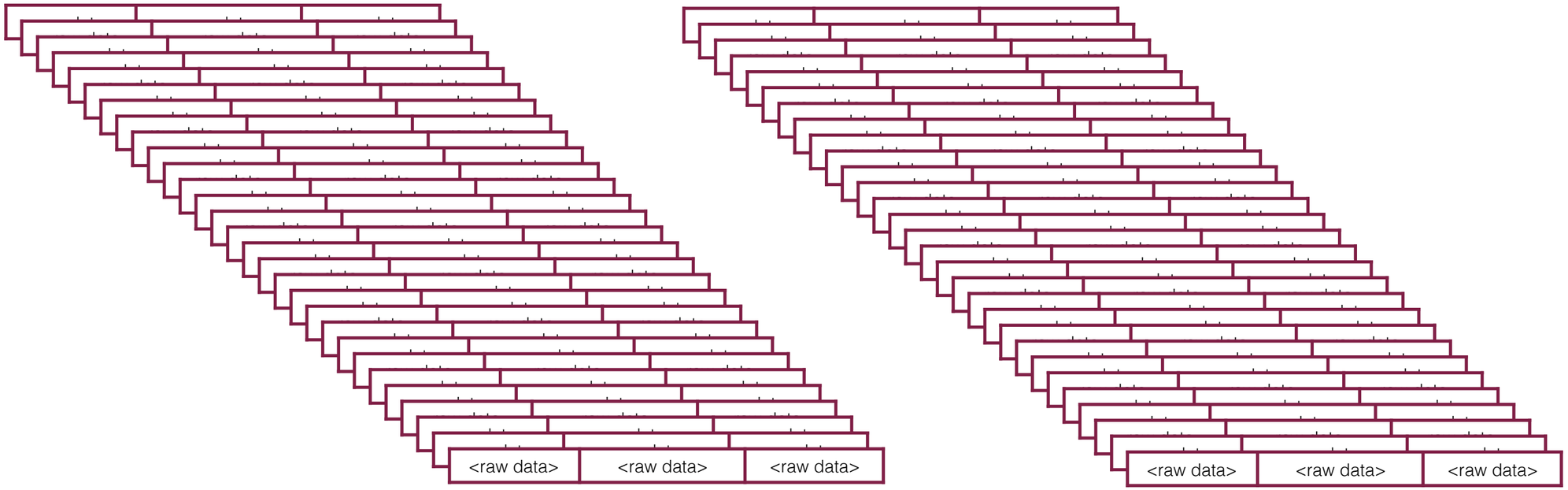
Takes advantage of the inherent parallelism in data processing

MapReduce



Modern systems generate millions of records of raw data

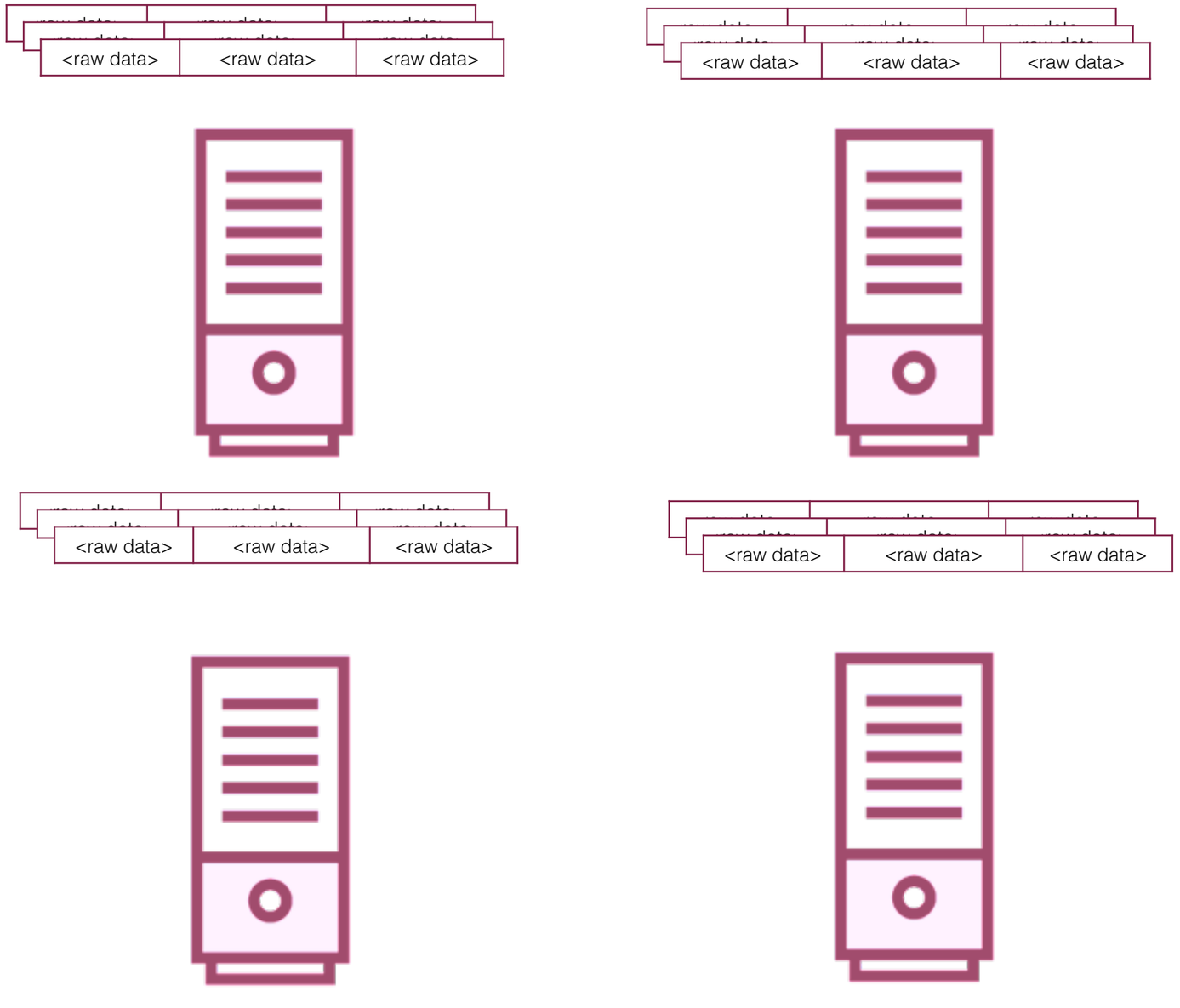
MapReduce



A task of this scale is processed in two stages

`map` `reduce`

map


<raw data>
<raw data>
<raw data>
<raw data>
<raw data>
<raw data>
<raw data>
<raw data>
<raw data>
<raw data>
<raw data>
<raw data>

reduce

<raw data>	<raw data>	<raw data>
<raw data>	<raw data>	<raw data>
<raw data>	<raw data>	<raw data>
<raw data>	<raw data>	<raw data>

map

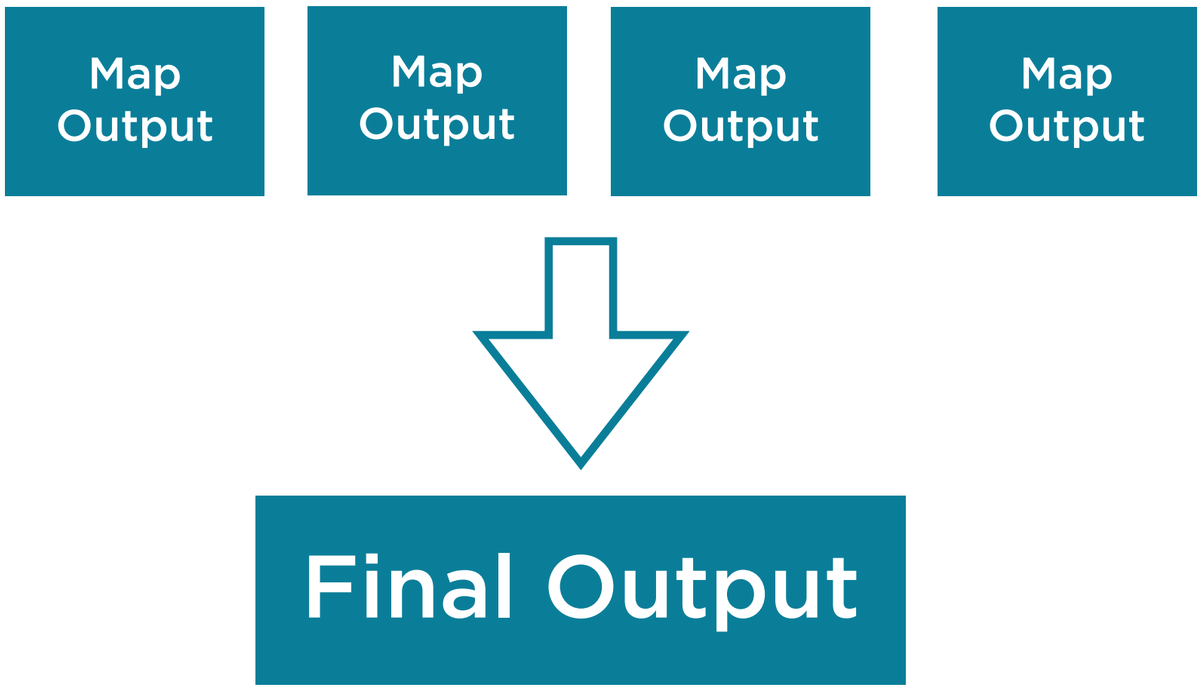
An operation performed in parallel, on small portions of the dataset

map

reduce

An operation to combine the results of the map step

reduce



map A step that can be performed in parallel

reduce A step to combine the intermediate results

Join Columns and MapReduce Jobs

Join Columns

Join combines records from two or more tables on the same column value

Join Columns

Trades

Symbol	Open	High	Low	Close	Day
GOOG	820	840	818	829	1-1-2017

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

```
select * from Names join Trades
on Names.Symbol = Trades.Symbol
```

Join Columns

Trades

Symbol	Open	High	Low	Close	Day
GOOG	820	840	818	829	1-1-2017

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

One join column = one MapReduce job

Join Columns

Trades

Symbol	Open	High	Low	Close	Day
GOOG	820	840	818	829	1-1-2017

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

```
select * from Names join Trades
on (Names.Symbol = Trades.Symbol)
join Revenues on (Names.Symbol = Revenues.Symbol)
```

Join Columns

Trades

Symbol	Open	High	Low	Close	Day
GOOG	820	840	818	829	1-1-2017

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

One join column = one MapReduce job

Join Columns

Trades

Symbol	Open	High	Low	Close	Day
GOOG	820	840	818	829	1-1-2017

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

```
select * from Names join Trades
on (Names.Symbol = Trades.Symbol)
join Revenues on (Names.Name = Revenues.Name)
```

Join Columns

Trades

Symbol	Open	High	Low	Close	Day
GOOG	820	840	818	829	1-1-2017

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

```
select * from Names join Trades
on (Names.Symbol = Trades.Symbol)
join Revenues on (Names.Name = Revenues.Name)
```

Join Columns

Trades

Symbol	Open	High	Low	Close	Day
GOOG	820	840	818	829	1-1-2017

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

Two join columns = two MapReduce jobs

For faster queries...

Minimize the number of MapReduce jobs run

Demo

Join operations on 3 tables, with different columns in the join clause

Join Operations and Table Sizes

Size of Tables

Trades 500GB

Symbol	Open	High	Low	Close
GOOG	820	840	818	829

Names 10MB

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues 100MB

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

```
select * from Names join Trades
on (Names.Symbol = Trades.Symbol)
join Revenues on (Names.Symbol = Revenues.Symbol)
```

Size of Tables

Trades 500GB

Symbol	Open	High	Low	Close
GOOG	820	840	818	829

Names 10MB

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues 100MB

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

```
select * from Names join Trades
on (Names.Symbol = Trades.Symbol)
join Revenues on (Names.Symbol = Revenues.Symbol)
```

Size of Tables

Trades 500GB

Symbol	Open	High	Low	Close
GOOG	820	840	818	829

Names 10MB

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues 100MB

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

Names 10MB

Trades 500GB

Revenues 100MB

Size of Tables

Names	Trades	Revenues
10MB	500GB	100MB

All tables, except for the last are held in memory

Size of Tables

Names
10MB

Trades
500GB

Revenues
100MB

The last table is streamed from disk to the job

Size of Tables

Names

10MB

Trades

500GB

Revenues

100MB

Inefficient to keep large tables in memory

Size of Tables

Names

10MB

Revenues

100MB

Trades

500GB

Re-order the tables in the join so the largest table is at the end

Size of Tables

Trades 500GB

Symbol	Open	High	Low	Close
GOOG	820	840	818	829

Names 10MB

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues 100MB

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

```
select * from Names join Revenues
on (Names.Symbol = Revenues.Symbol)
join Trades on (Names.Symbol = Trades.Symbol)
```

Size of Tables

Trades 500GB

Symbol	Open	High	Low	Close
GOOG	820	840	818	829

Names 10MB

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Revenues 100MB

Symbol	Name	Revenue
GOOG	Google	90B
AAPL	Apple	215B
MSFT	Microsoft	85B

```
select * from Names join Revenues
on (Names.Symbol = Revenues.Symbol)
join Trades on (Names.Symbol = Trades.Symbol)
```

For faster queries...

Specify the **largest** table
at the very **end**

```
select /*+ streamtable(Trades) */
Names.Symbol, Trades.High, Revenues.Revenue
from Names join Trades
on (Names.Symbol = Trades.Symbol)
join Revenues on (Names.Symbol = Revenues.Symbol)
```

The Streamtable Keyword

Specify which table to stream in a join operation

Stream the largest table, do not hold it in memory

Join Optimizations with Bucketing and Partitioning

Bucketing

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
1	iPhone	599
2	Doll	35
3	Shoes	33
4	Jeans	69
5	Skates	123
6	Make Up	99
7	Book	24
8	Belt	20

Join Orders and Products to get the names of the products users have bought

Bucketing

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
7	iPhone	599
8	Doll	35
3	Shoes	33
1	Jeans	69
6	Skates	123
5	Make Up	99
4	Book	24
2	Belt	20

Product ID is the join column

Bucketing

Orders

ID	Product ID	Quantity	Amount
o1	**4**	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
7	iPhone	599
8	Doll	35
3	Shoes	33
1	Jeans	69
6	Skates	123
5	Make Up	99
4	**Book**	**24**
2	Belt	20

Need to scan the entire dataset to find the corresponding row

Bucketing

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
6	Skates	123
3	Shoes	33

ID	Name	Cost
7	iPhone	599
1	Jeans	69
4	Book	24

ID	Name	Cost
2	Belt	20
8	Doll	35
5	Make Up	99

Bucket the Products table on the ID column

Bucketing

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
6	Skates	123
3	Shoes	33

ID	Name	Cost
7	iPhone	599
1	Jeans	69
4	**Book**	**24**

ID	Name	Cost
2	Belt	20
8	Doll	35
5	Make Up	99

Scan a much smaller dataset to access each row

Bucketing

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
6	Skates	123
3	Shoes	33

ID	Name	Cost
7	iPhone	599
1	Jeans	69
4	Book	24

ID	Name	Cost
2	Belt	20
8	Doll	35
5	Make Up	99

Faster joins

Partitioning

Join optimizations would work the same way

Reduce the dataset to scan to find the corresponding row

For faster queries...

Use **bucketing** or **partitioning**
on the join columns

Left Semi-joins Instead of Subqueries

Left Semi-join

Any join query that requests rows from the left row source based on the existence of rows in the right row source without including data from the right row source in the final result and without duplicating rows from the left row source is a logical left semi join.

http://sqlity.net/

Left Semi-join

Any join query that requests rows from the left row source based on the existence of rows in the right row source without including data from the right row source in the final result and without duplicating rows from the left row source is a logical left semi join.

http://sqlity.net/

Left Semi-join

Any join query that requests rows from the left row source based on the existence of rows in the right row source without including data from the right row source in the final result and without duplicating rows from the left row source is a logical left semi join.

http://sqlity.net/

Left Semi-join

Any join query that requests rows from the left row source based on the existence of rows in the right row source without including data from the right row source in the final result and without duplicating rows from the left row source is a logical left semi join.

http://sqlity.net/

Left Semi-join

Names

Symbol	Name
GOOG	Google
~~AAPL~~	~~Apple~~
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					
MSFT					

```
select names.symbol
from names left semi join trades
on
(names.symbol = trades.symbol);
```

Left Semi-join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					
MSFT					

Symbol	Name
GOOG	Google
MSFT	Microsoft

```
select symbol from names
where symbol in
(select symbol from trades);
```

The IN/NOT IN Keywords

Check for existence of a column value in the subquery

```
select symbol from names
where symbol in
(select symbol from trades);
```

The IN/NOT IN Keywords

Only one column can be selected in the subquery

```
select names.symbol
from names
where exists
(select trades.symbol from trades
where names.symbol = trades.symbol);
```

The EXISTS/NOT EXISTS Keywords

Check if even one result is returned from the subquery

```
select names.symbol
from names
where exists
(select trades.symbol from trades
where names.symbol = trades.symbol);
```

The EXISTS/NOT EXISTS Keywords

The subquery has to reference the parent query i.e. should be correlated

Left Semi-join

It is much more efficient to replace the IN/EXISTS subqueries with a left semi-join

```
select names.symbol
from names left semi join trades
on
(names.symbol = trades.symbol);
```

The Left Semi-join Operator

Choose records from the left table

```
select names.symbol
from names left semi join trades
on
(names.symbol = trades.symbol);
```

The Left Semi-join Operator

Choose records from the left table

```
select names.symbol
from names left semi join trades
on
(names.symbol = trades.symbol);
```

The Left Semi-join Operator

Choose records from the left table

Based on the existence of rows in the right table

```
select names.symbol
from names left semi join revenues
on
(names.symbol = revenues.symbol
and
names.name = revenues.name);
```

The Left Semi-join Operator

Allows specifying multiple columns in the join

Not allowed when we use the IN keyword

Left Semi-join vs. In/Exists

Left Semi-join

Only scans the right table till a match is found

Can specify matches on multiple columns

Bucketing or partitioning the right table can improve performance

In/Exists

Needs to the scan the entire subquery table

IN allows only one column to be selected in the subquery

No advantage of a bucketed or partitioned right table

For faster queries...

Use the **left semi-join** rather than in/exists keywords with subqueries

Demo

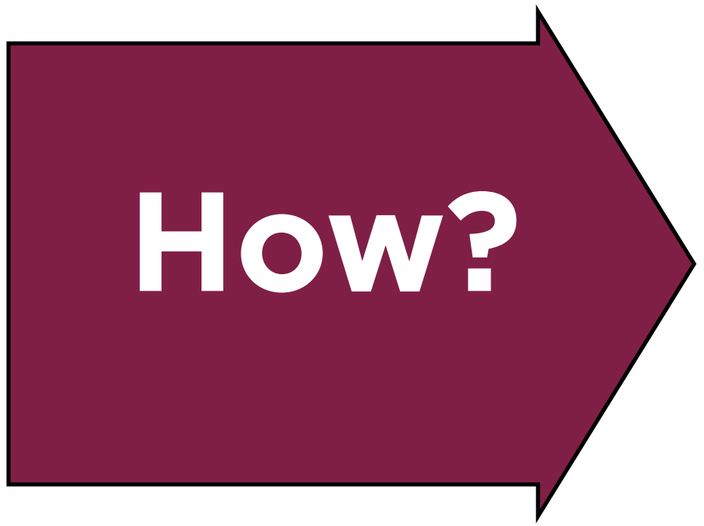
The left semi-join as opposed to the in/exists subqueries

The Anatomy of a MapReduce Program

Counting Word Frequencies

Consider a large text file

Twinkle twinkle little star
How I wonder what you are
Up above the world so high
Like a diamond in the sky
Twinkle twinkle little star
How I wonder what you are
.....



Word	Frequency
above	14
are	20
how	21
star	22
twinkle	32
...	..

MapReduce Flow

Twinkle twinkle little star

How I wonder what you are

Up above the world so high

Like a diamond in the sky

Twinkle twinkle little star

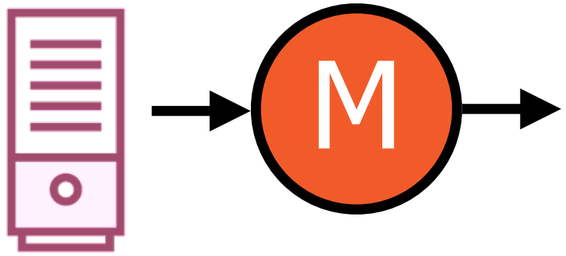
How I wonder what you are

Each partition is given to a different process i.e. to mappers

MapReduce Flow

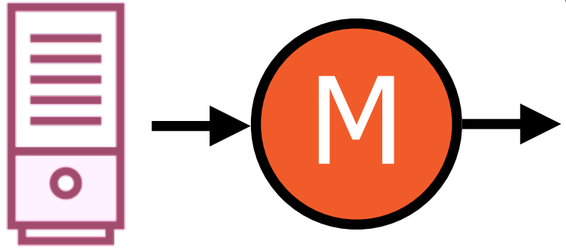
Twinkle twinkle little star

How I wonder what you are



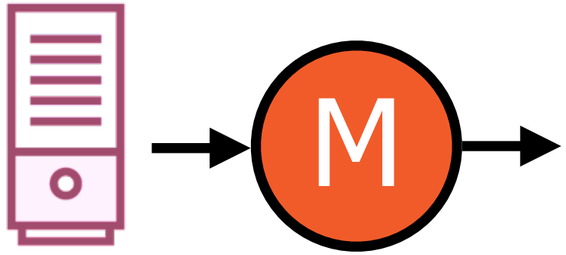
Up above the world so high

Like a diamond in the sky



Twinkle twinkle little star

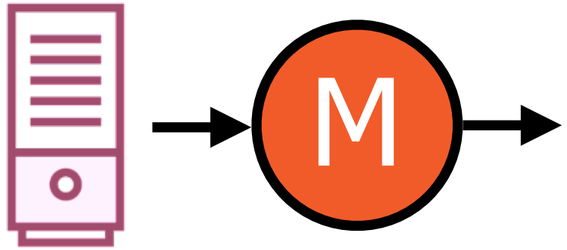
How I wonder what you are



Each mapper works in parallel

Map Flow

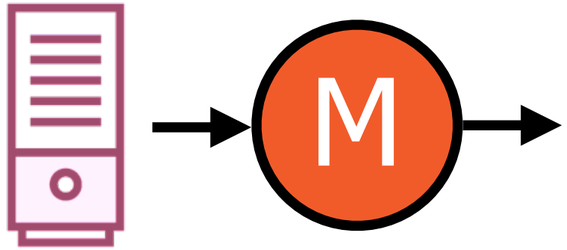
Twinkle twinkle little star
How I wonder what you are



Within each mapper, the rows are processed serially

Map Flow

Twinkle twinkle little star
How I wonder what you are

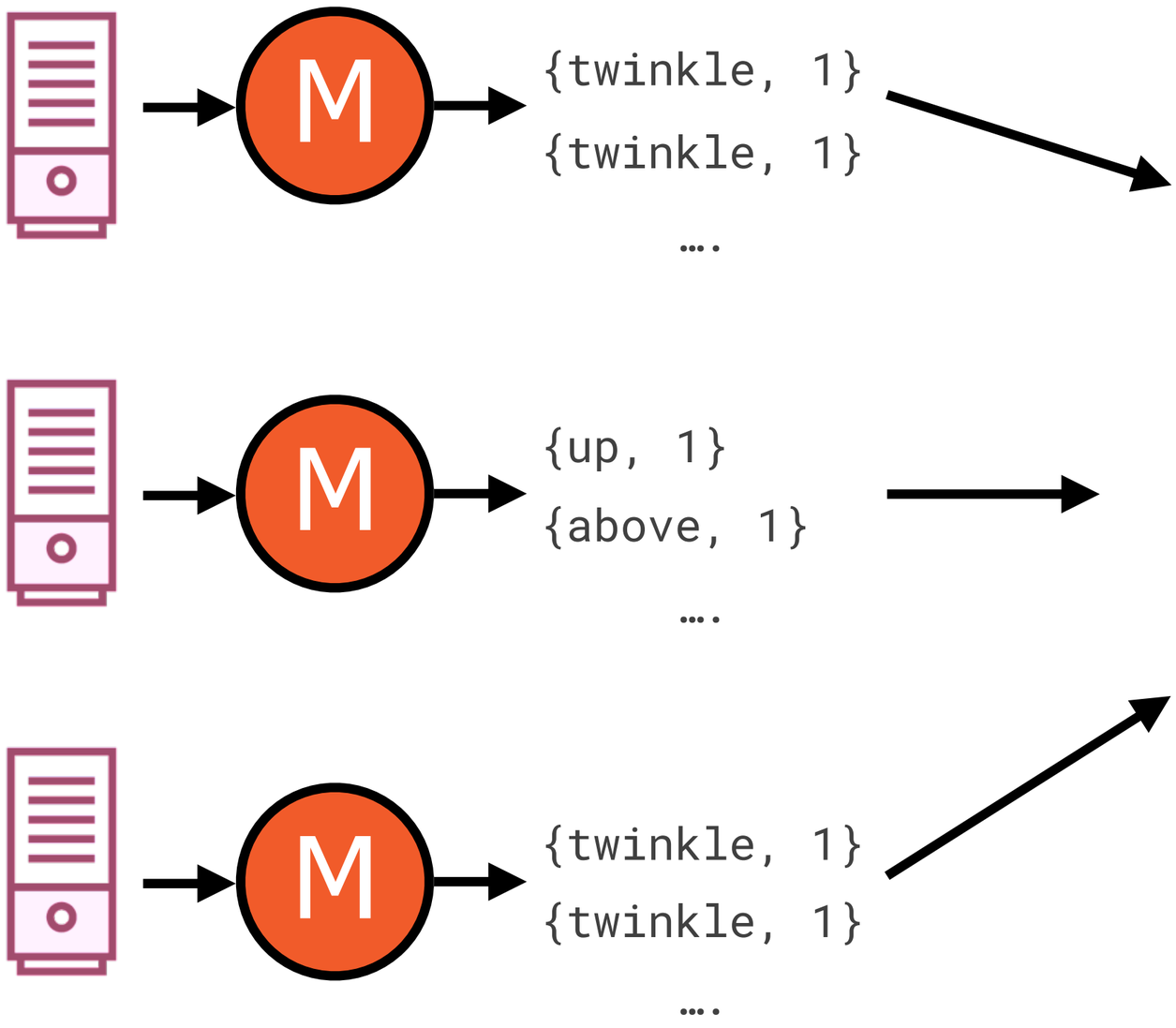


Word	# Count

{twinkle, 1}
{twinkle, 1}
{little, 1}
{star, 1}

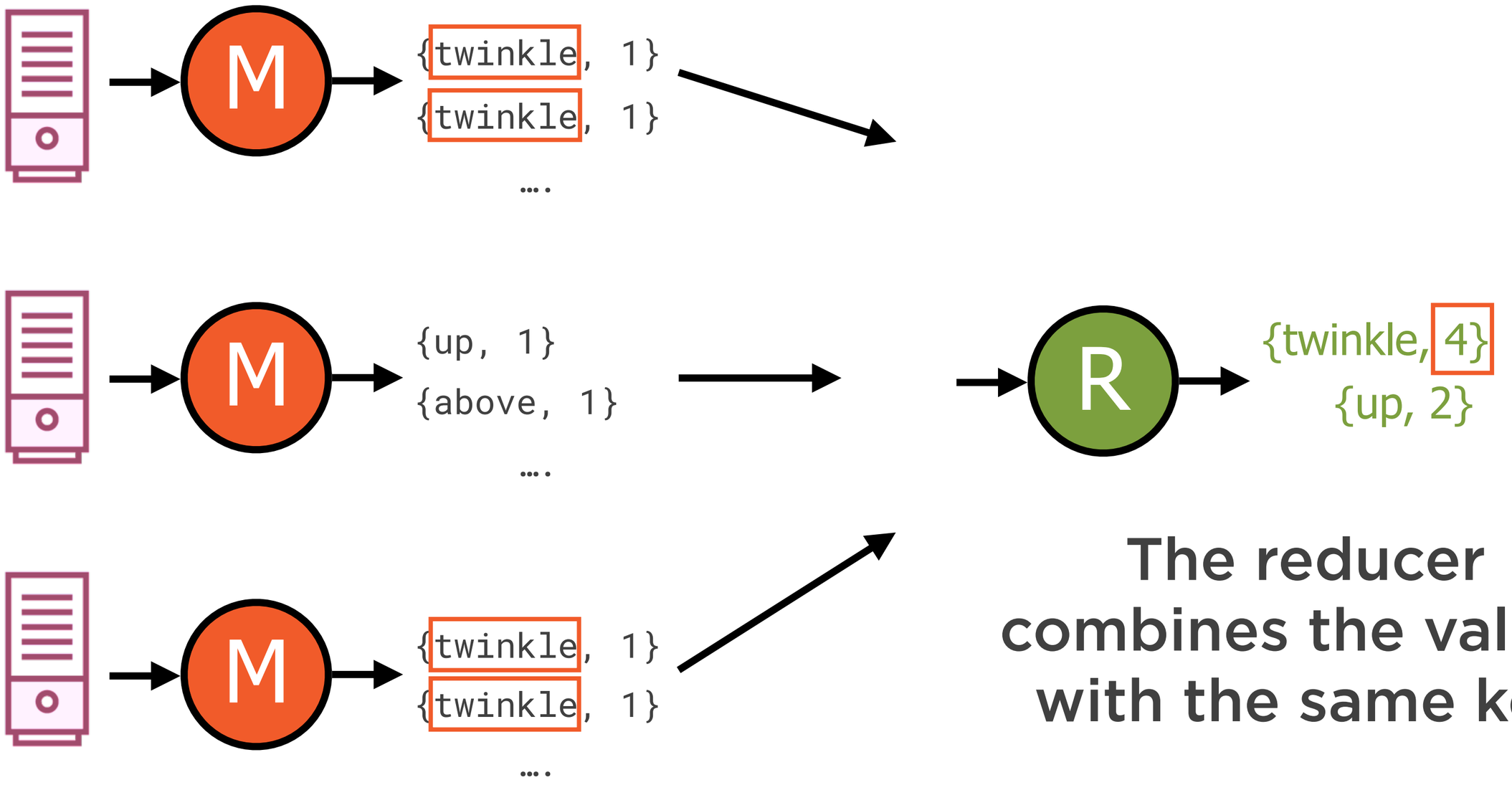
Each row emits {key, value} pairs

Reduce Flow



The results are passed on to another process i.e. a reducer

Reduce Flow



The reducer combines the values with the same key

Key Insight Behind MapReduce

Many data processing tasks can be expressed in this form

MapReduce

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
1	iPhone	599
2	Doll	35
3	Shoes	33
4	Jeans	69
5	Skates	123
6	Make Up	99
7	Book	24
8	Belt	20

```
select * from Orders join Products
on Orders.ProductID = Products.ID
```

MapReduce

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35

Products

ID	Name	Cost
1	iPhone	599
2	Doll	35

Product ID	ID, Quantity, Amount

ID	Name, Cost

The mapper operates on each row of the tables

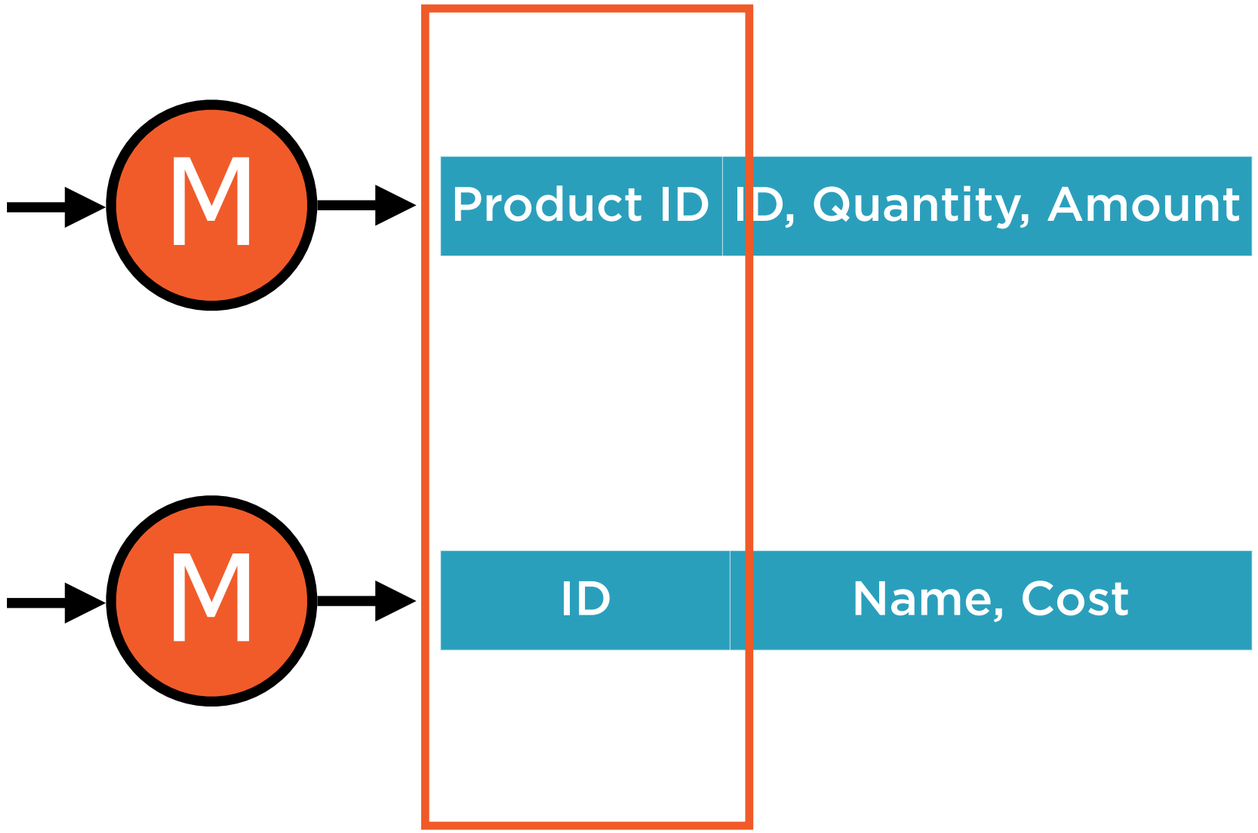
MapReduce

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35

Products

ID	Name	Cost
1	iPhone	599
2	Doll	35



The join column is the key

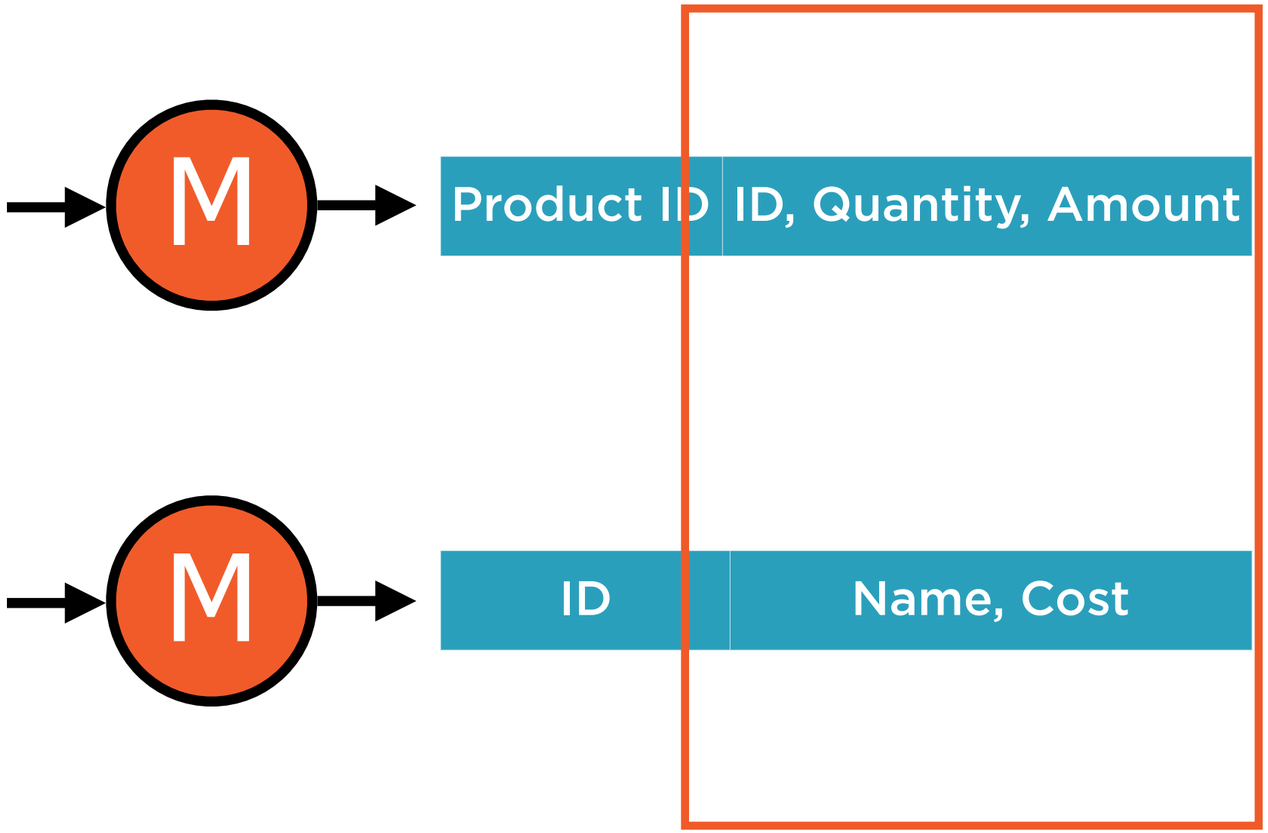
MapReduce

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35

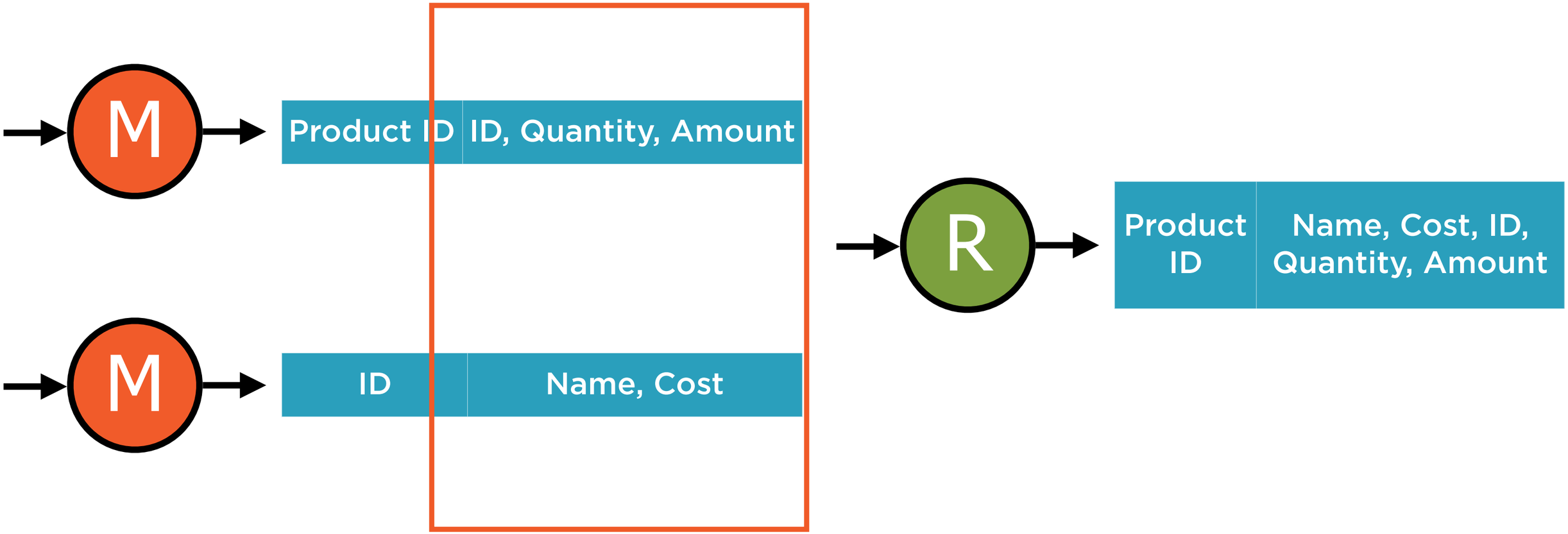
Products

ID	Name	Cost
1	iPhone	599
2	Doll	35



The remaining columns are values

MapReduce



The reducer combines all columns which have the same key

Joins as Map-only Operations

MapReduce operations have 2 phases of processing

Joins as Map-only Operations

Certain queries can be structured to have no reduce phase

Joins as Map-only Operations

Such joins are called map-side joins

More performant

Joins as Map-only Operations

Improves processing time

Reduces data transfer between machines in the cluster

Reduces operations such as shuffle and sort between map and reduce phases

Joins as Map-only Operations

We'd like joins to be map-side joins if possible

Conditions for Map-side Joins

Conditions for Map-side Joins

All tables, except one, are small enough to be held in memory

Tables are bucketed on the join columns and
Table1 Buckets = N * Table2 Buckets

Conditions for Map-side Joins

All tables, except one, are small enough to be held in memory

Tables are bucketed on the join columns and Table1 Buckets = N * Table2 Buckets

All Tables but One Are Small

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

Left table is the smaller table

Inner Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

Only rows which have a match in both the left and right table

Inner Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

```
select * from Names join Trades
on (Names.Symbol = Trades.Symbol)
```

Inner Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Names

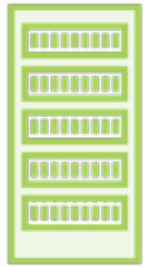
Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

A copy of the smaller table is stored in a hash table like structure

Inner Join

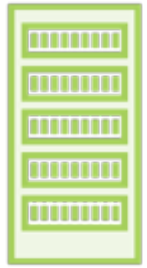
Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft



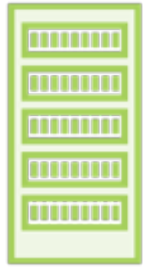
Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft



Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft



This hash table is copied to each mappers' local disk

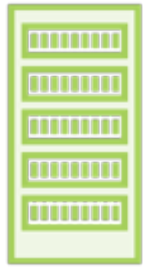
Inner Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

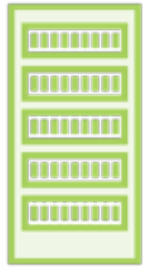


Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
AAPL					

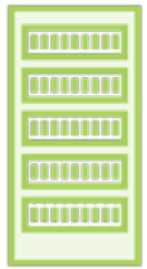


Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
MSFT					



Parts of the larger table are distributed to each mapper

Inner Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
GOOG

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
AAPL

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
MSFT

Mappers run on the entire Names and parts of the Trades table

Inner Join

Names

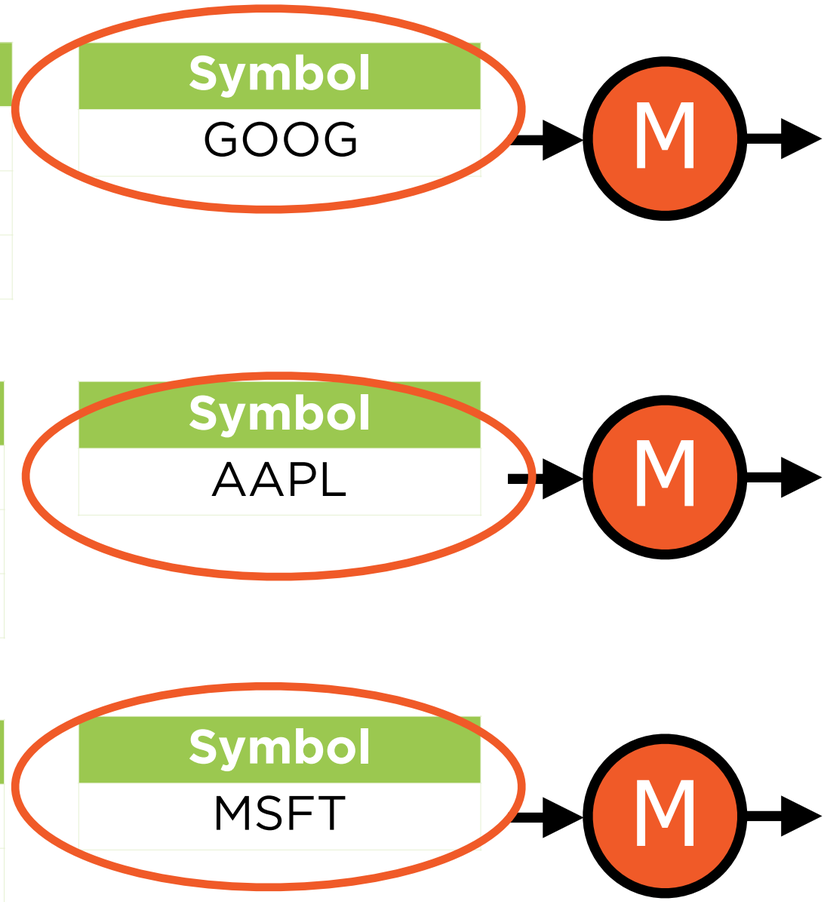
Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft



Combine with those rows which are available on the mapper

Inner Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

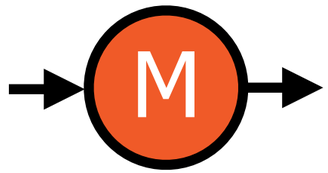
Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Names

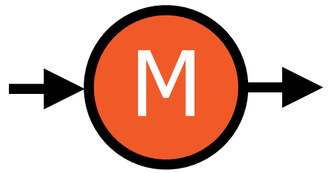
Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
GOOG



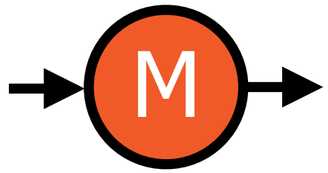
Symbol	Other Columns
GOOG	

Symbol
AAPL



Symbol	Other Columns
AAPL	

Symbol
MSFT



Symbol	Other Columns
MSFT	

The output of all the mappers forms the final output

Inner Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
GOOG

Symbol	Other Columns
GOOG	

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
AAPL

Symbol	Other Columns
AAPL	

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
MSFT

Symbol	Other Columns
MSFT	

No reducer needed

The left table is the smaller table

Inner Joins

Are possible as map-only joins

Left Outer Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

All rows from the left table are in the result

- with a matching row
- padded with nulls

Left Outer Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

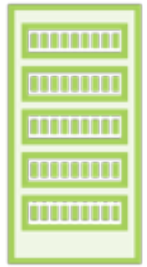
Symbol	Open	High	Low	Close	Day
GOOG					

```
select * from Names left outer join Trades
on (Names.Symbol = Trades.Symbol)
```

Left Outer Join

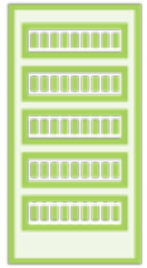
Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft



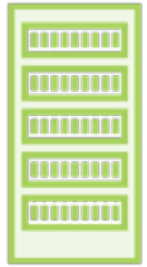
Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft



Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft



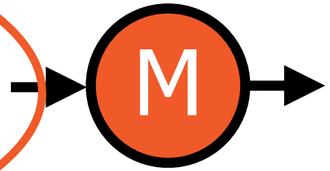
The smaller table is copied to each mappers's disk

Left Outer Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
GOOG



Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
AAPL

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
MSFT

Is the row not present in this chunk or not present in the entire table?

No way to tell

The left table is the smaller table

Inner Joins

~~**Left Outer Joins**~~

Are possible as map-only joins

Right Outer Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

All rows from the right table are in the result

- with a matching row
- padded with nulls

Right Outer Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

```
select * from Names right outer join Trades
on (Names.Symbol = Trades.Symbol)
```

Right Outer Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
GOOG

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
AAPL

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
MSFT

The entire left table is present to check for matches

The left table is the smaller table

Inner Joins

~~**Left Outer Joins**~~

Right Outer Joins

Are possible as map-only joins

Full Outer Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

All rows from the both tables are in the result

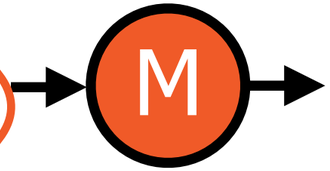
- with a matching row
- padded with nulls

Full Outer Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
GOOG



Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
AAPL

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Symbol
MSFT

Should a row be padded with nulls or does it have a match?

No way to tell

The left table is the smaller table

Inner Joins

~~**Left Outer Joins**~~

Right Outer Joins

~~**Full Outer Joins**~~

Are possible as map-only joins

Right Table Is Small

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

What is possible as a map-only join now is different

The right table is the smaller table

Inner Joins

Left Outer Joins

~~**Right Outer Joins**~~

~~**Full Outer Joins**~~

Are possible as map-only joins

Conditions for Map-side Joins

All tables, except one, are small enough to be held in memory

Tables are bucketed on the join columns and
Table1 Buckets = N * Table2 Buckets

Conditions for Map-side Joins

All tables, except one, are small enough to be held in memory

Tables are bucketed on the join columns and
Table1 Buckets = N * Table2 Buckets

Tables Bucketed on Join Columns

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

Join column = Symbol

Bucketed column = Symbol

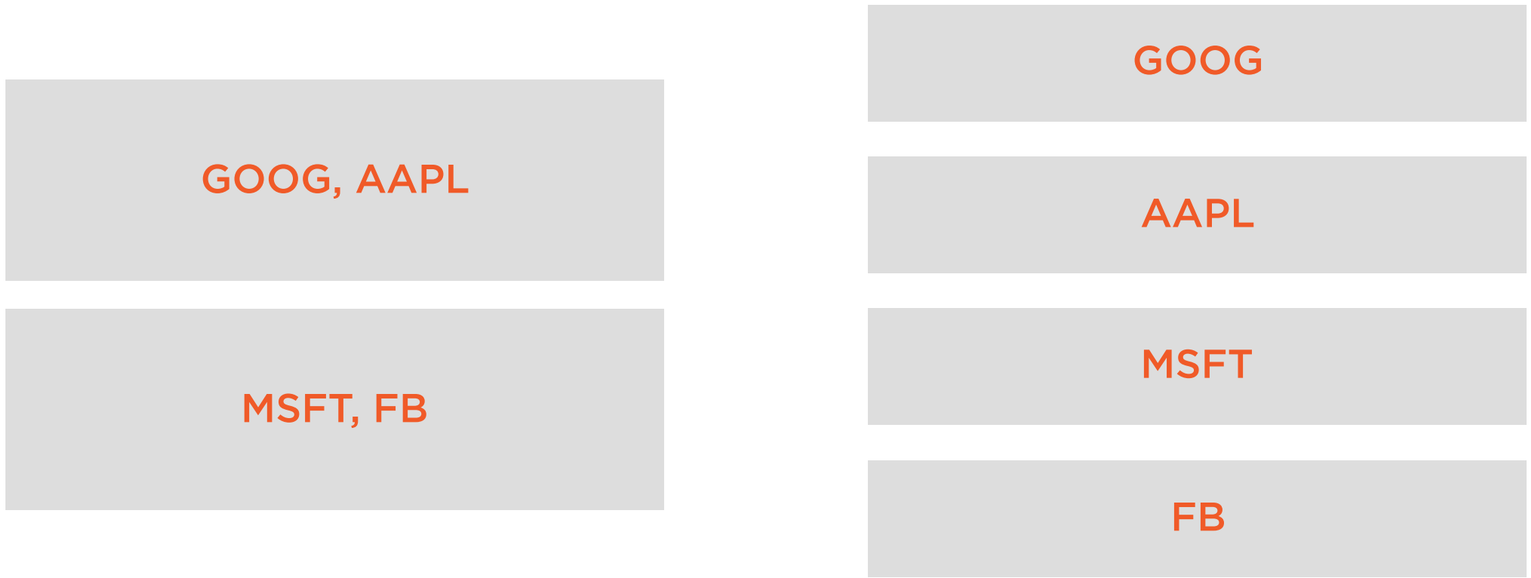
Tables Bucketed on Join Columns

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					



Tables Bucketed on Join Columns

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

Tables Bucketed on Join Columns

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					

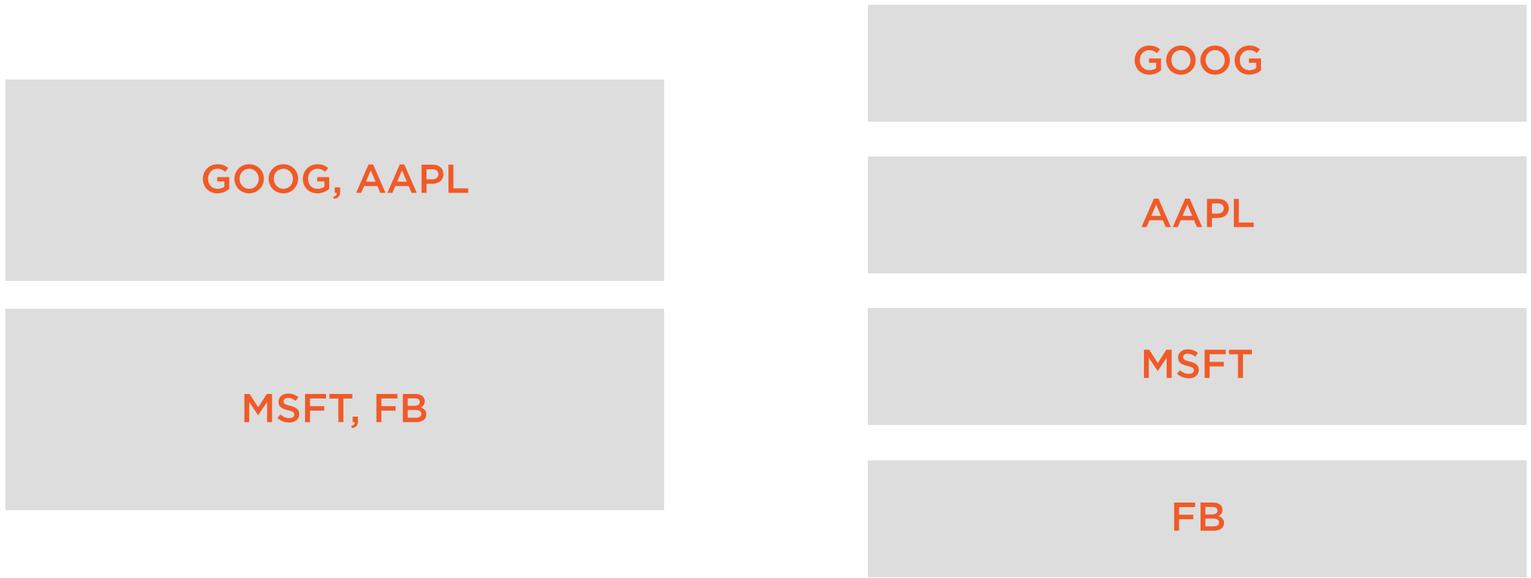
Bucket Map Join

Names

Symbol	Name
GOOG	Google
AAPL	Apple
MSFT	Microsoft

Trades

Symbol	Open	High	Low	Close	Day
GOOG					



```
set hive.optimize.bucketmapjoin = true;
```

Enable Bucket Map Join

This is not enabled by default in Hive

Bucket map joins are more efficient when tables are...

Bucketed on the join column

Sorted on the join column

Have the **same number** of buckets

Rows can be joined using
merge-sort

Can only be used for **equi-joins**

```
set hive.input.format =
org.apache.hadoop.hive.ql.io.BucketizedHiveInputFormat;

set hive.optimize.bucketmapjoin = true;

set hive.optimize.bucketmapjoin.sortedmerge = true;
```

Enable Bucket Map Join with Sort-merge

Format of the file that is read in from disk into Hive

```
set hive.input.format =
org.apache.hadoop.hive.ql.io.BucketizedHiveInputFormat;

set hive.optimize.bucketmapjoin = true;

set hive.optimize.bucketmapjoin.sortedmerge = true;
```

Enable Bucket Map Join with Sort-merge

Format of the file that is read in from disk into Hive

```
set hive.input.format =
org.apache.hadoop.hive.ql.io.BucketizedHiveInputFormat;

set hive.optimize.bucketmapjoin = true;

set hive.optimize.bucketmapjoin.sortedmerge = true;
```

Enable Bucket Map Join with Sort-merge

Format of the file that is read in from disk into Hive

```
select /*+ mapjoin(trades) */ names.symbol, high, day
from names join trades on
(names.symbol = trades.symbol)
```

Explicitly Perform a Map-join

Read the table specified completely on the mapper node

Summary

A deep understanding of how joins work in Hive

Faster joins on large tables

Optimized semi-joins in place of IN/ EXISTS subqueries

Understood under what conditions joins are map-only and how to optimize them