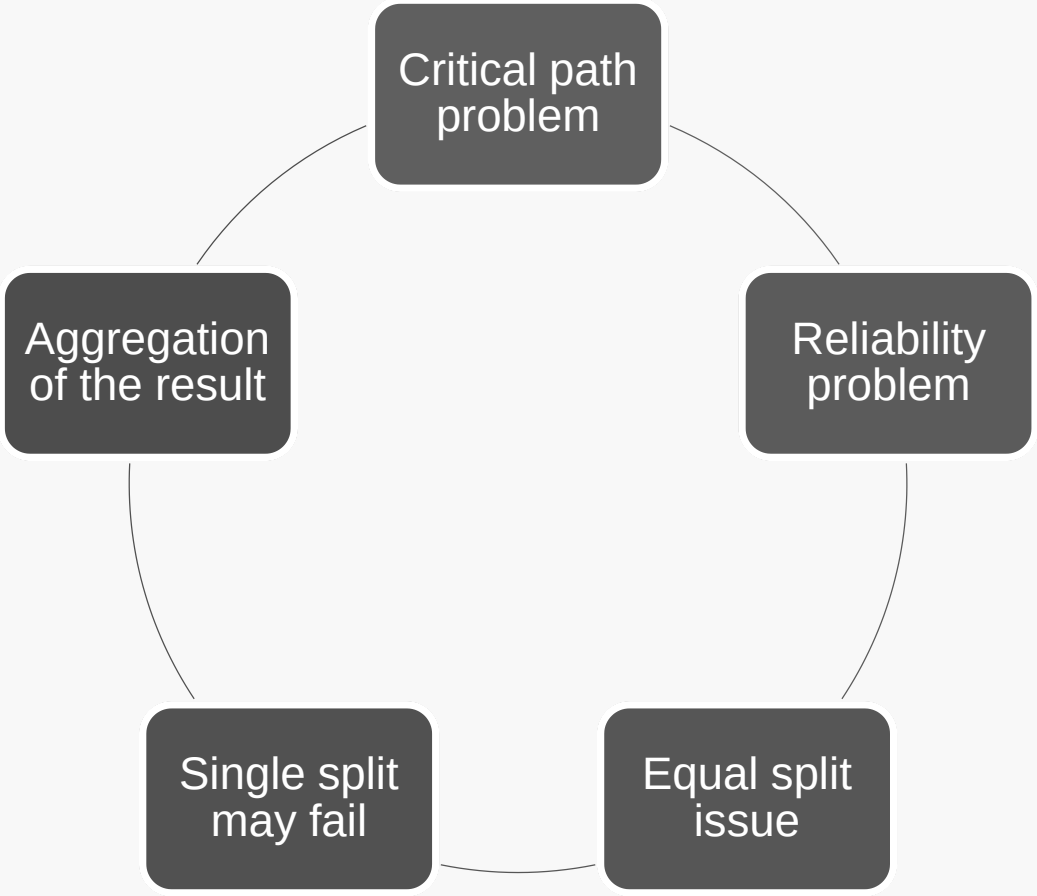
MAP-REDUCE

Submitted By
Maimoona Khilji
Registration no. 195300273
BS-Data Science
Semester - V

Challenges in Traditional approach



History

- MapReduce was **first popularized as a programming model in 2004** by Jeffery Dean and Sanjay Ghemawat of Google (Dean & Ghemawat, 2004).

- In their paper, “MAPREDUCE: SIMPLIFIED DATA PROCESSING ON LARGE CLUSTERS,” they discussed Google's approach to collecting and analyzing website data for search optimizations

What is MapReduce?

- Processing component of apache Hadoop
- Processes data parallel in distributed environment.
- MapReduce is a programming model or pattern within the Hadoop framework
- It is used to access big data stored in the Hadoop File System (HDFS).

Comparison between Traditional and Hadoop mapReduce

Comparison between Traditional and Hadoop mapReduce

1. Parallel Processing
   - Divide and Conquer process
2. Data Locality
   - Move the processing unit to the data in the MapReduce Framework

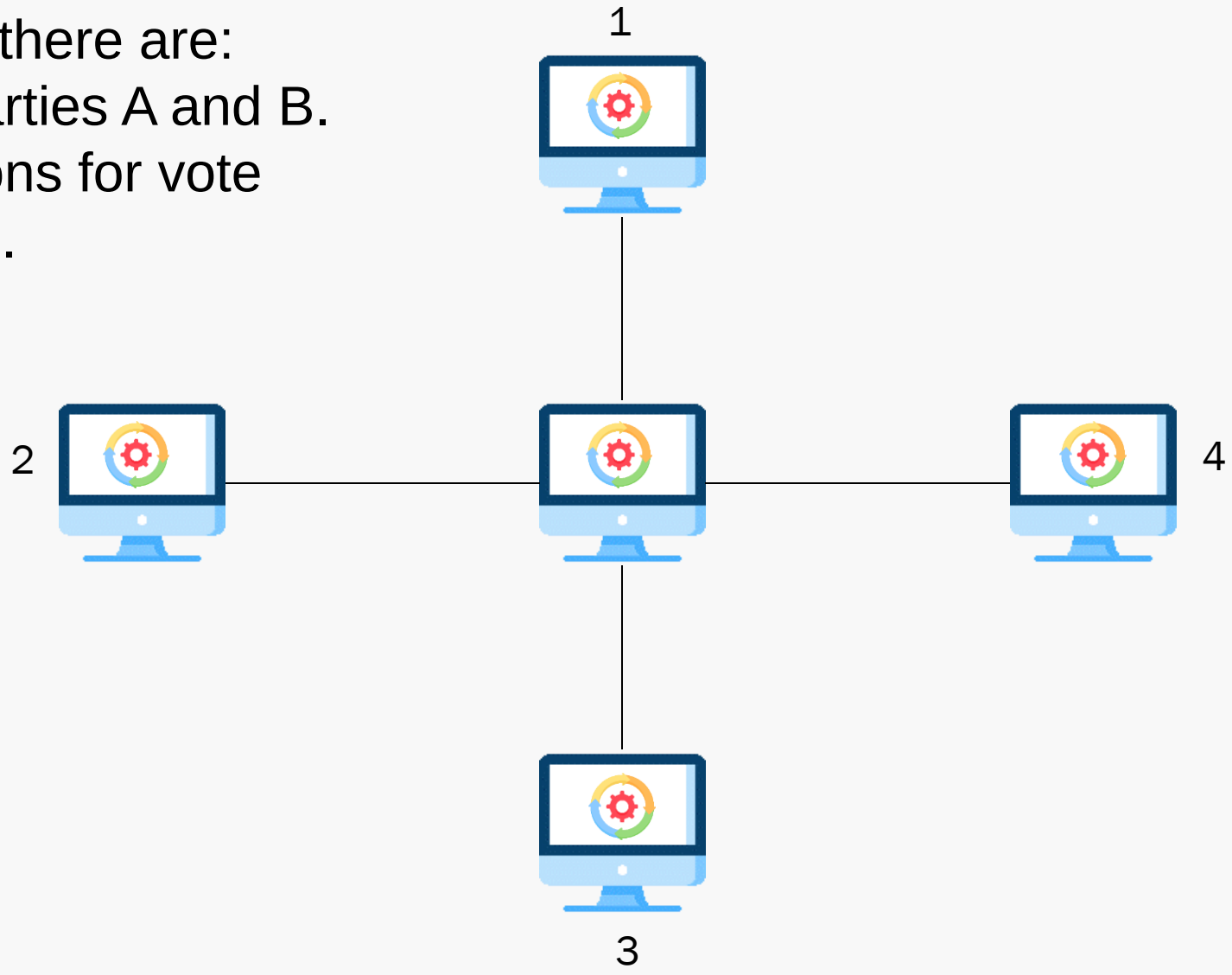
Mechanism

- MapReduce facilitates concurrent processing
  - *by splitting petabytes of data into smaller chunks*
  - *processing them in parallel on Hadoop commodity servers.*
  - *In the end, it aggregates all the data from multiple servers to return an output back to the application.*

Votes Counting

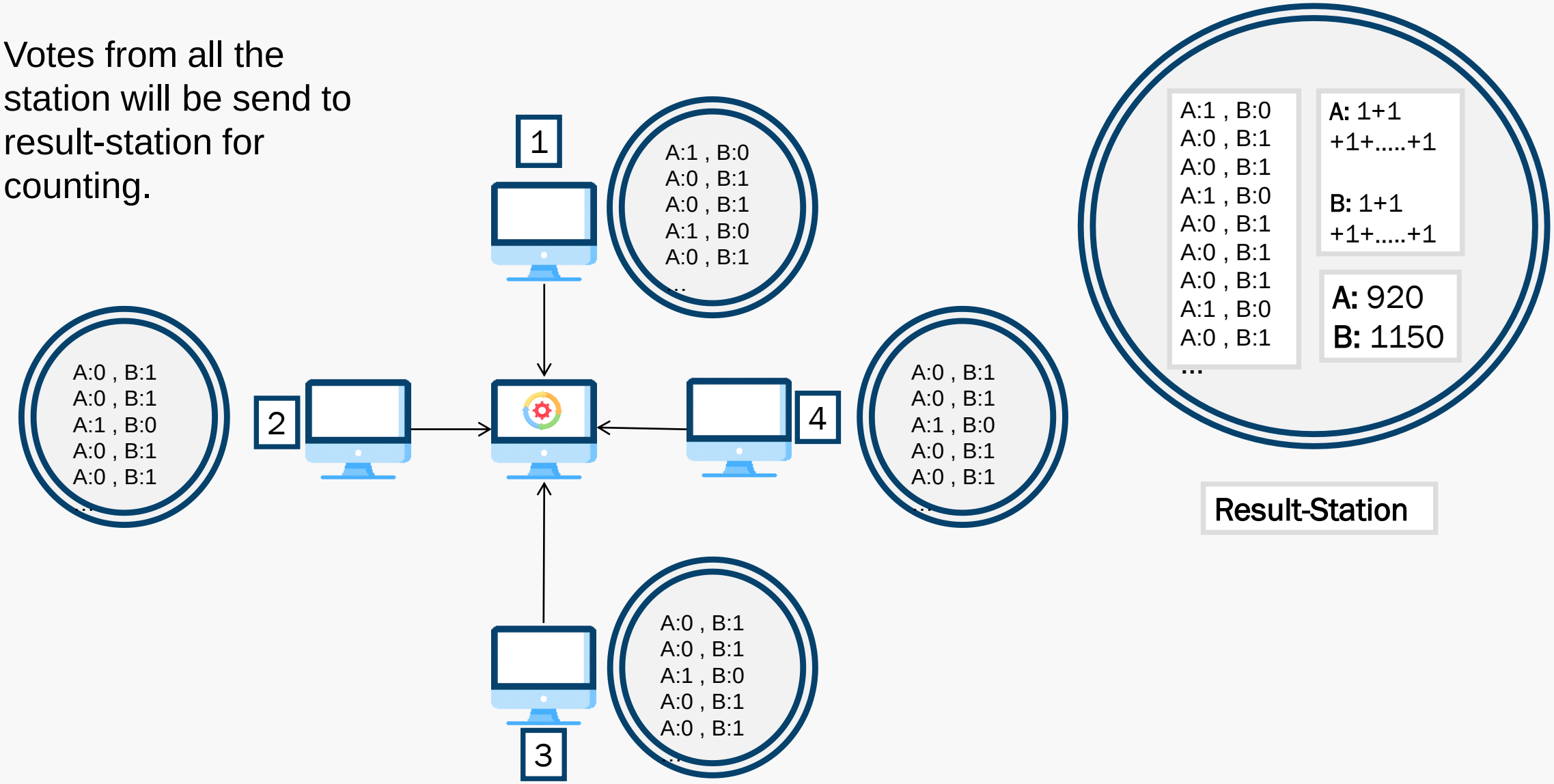
Suppose there are:

- Two parties A and B.
- 4 stations for vote casting.



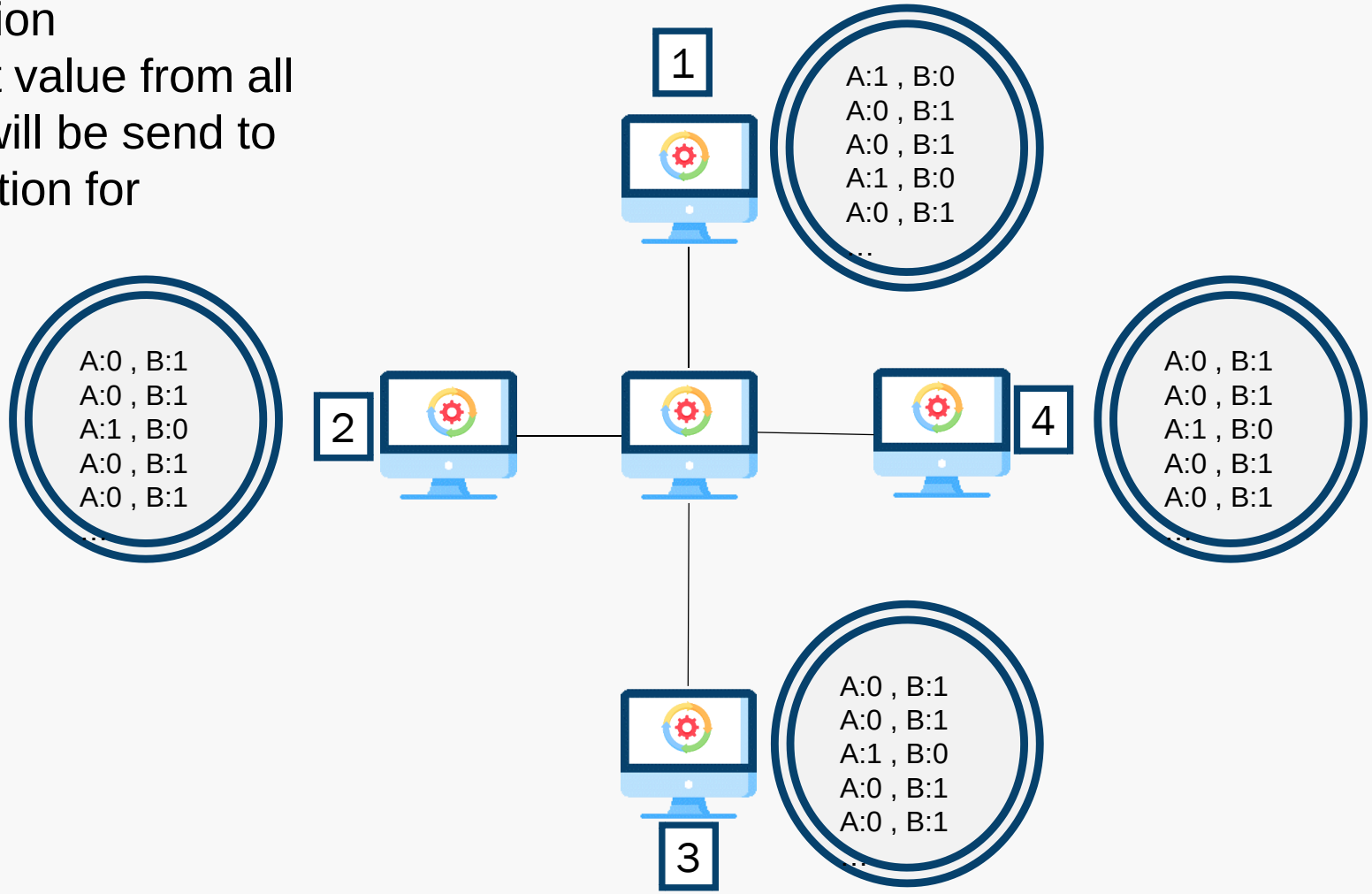
Traditional way of vote counting

- Votes from all the station will be send to result-station for counting.



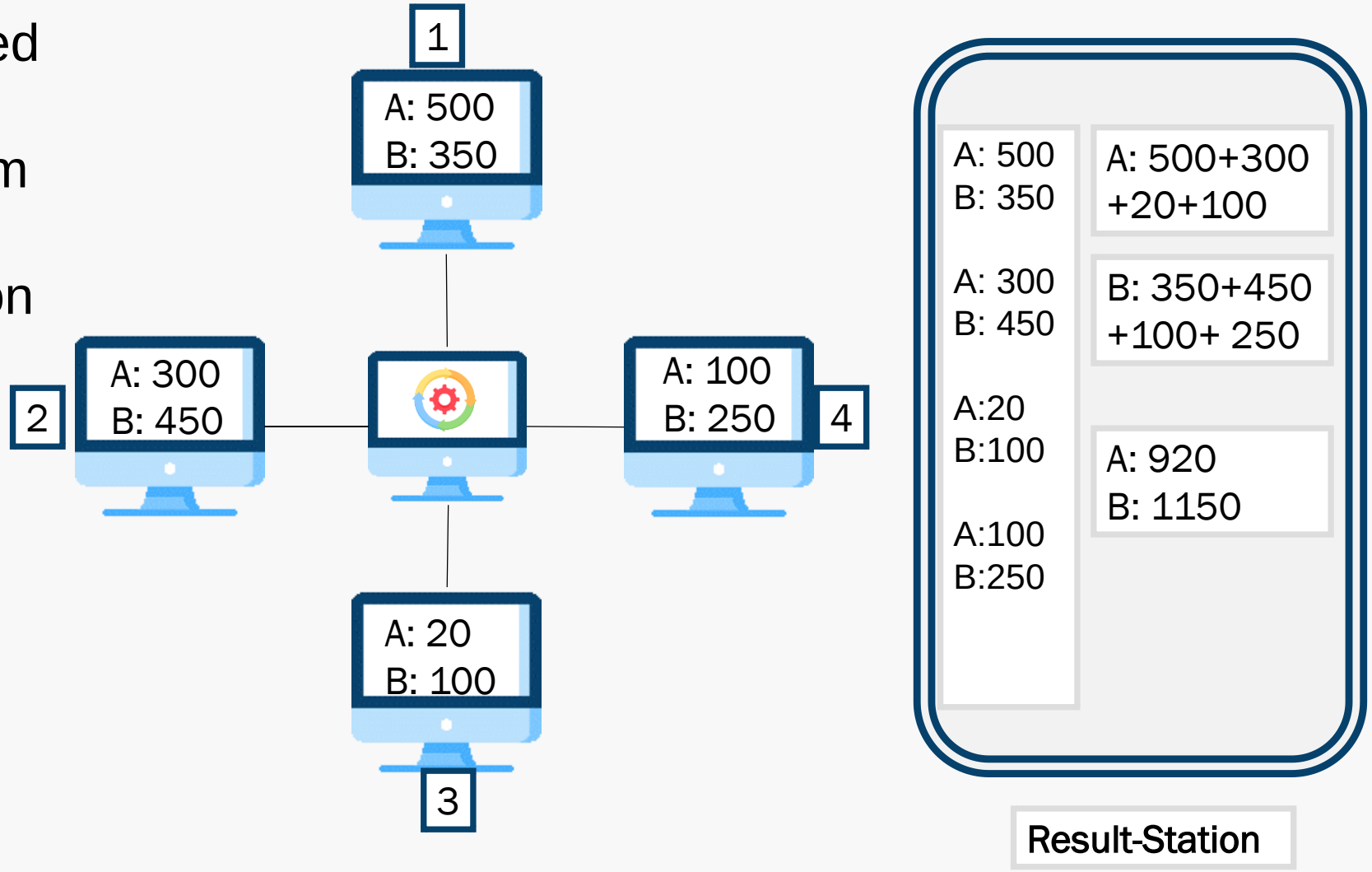
MapReduce way of vote counting

- Votes will be counted in each station
- Resultant value from all stations will be send to result-station for finalizing.

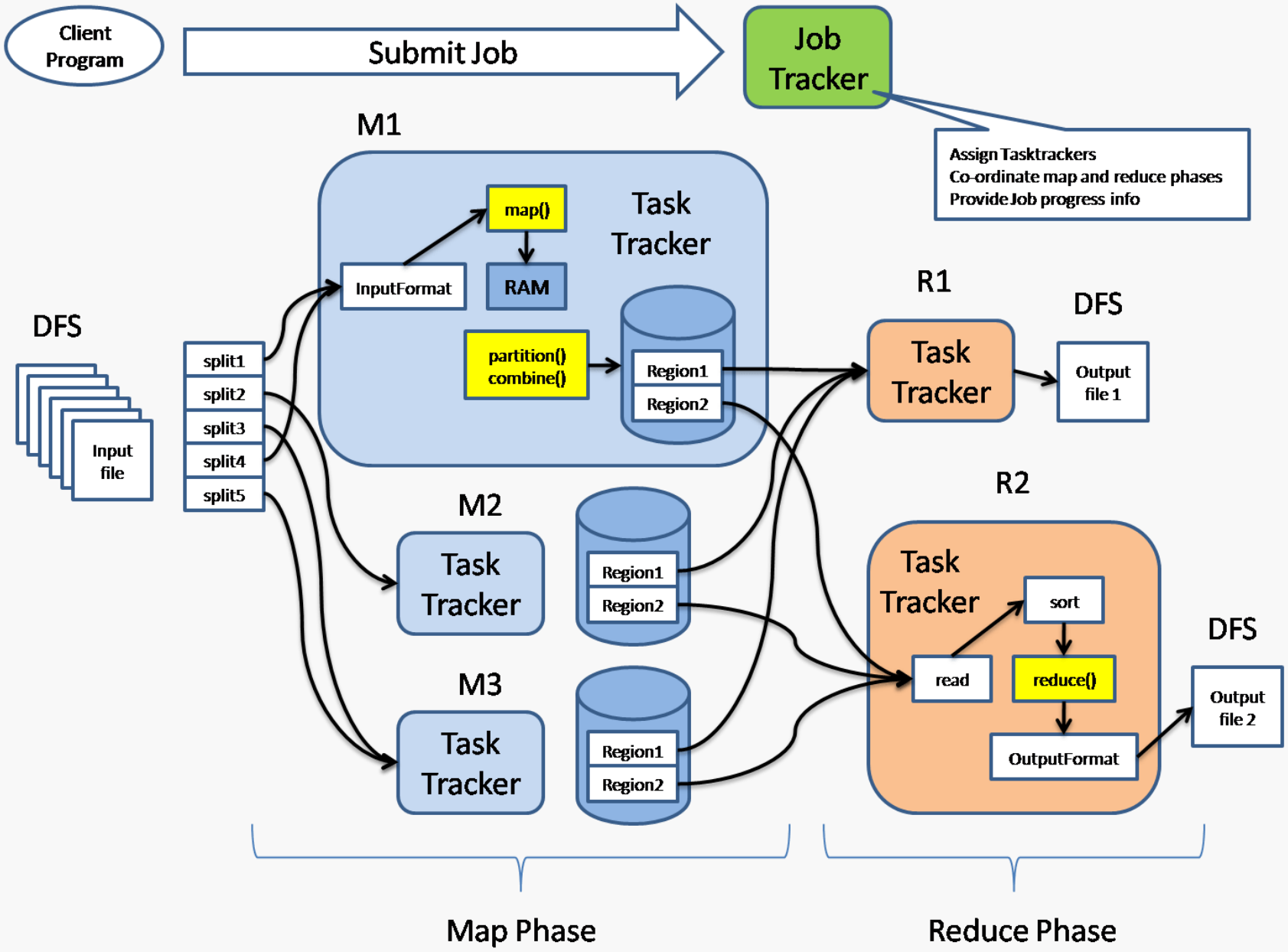


MapReduce way of vote counting

- Votes will be counted in each station
- Resultant value from all stations will be send to result-station for finalizing.



How MapReduce in Hadoop works



MapReduce Functions

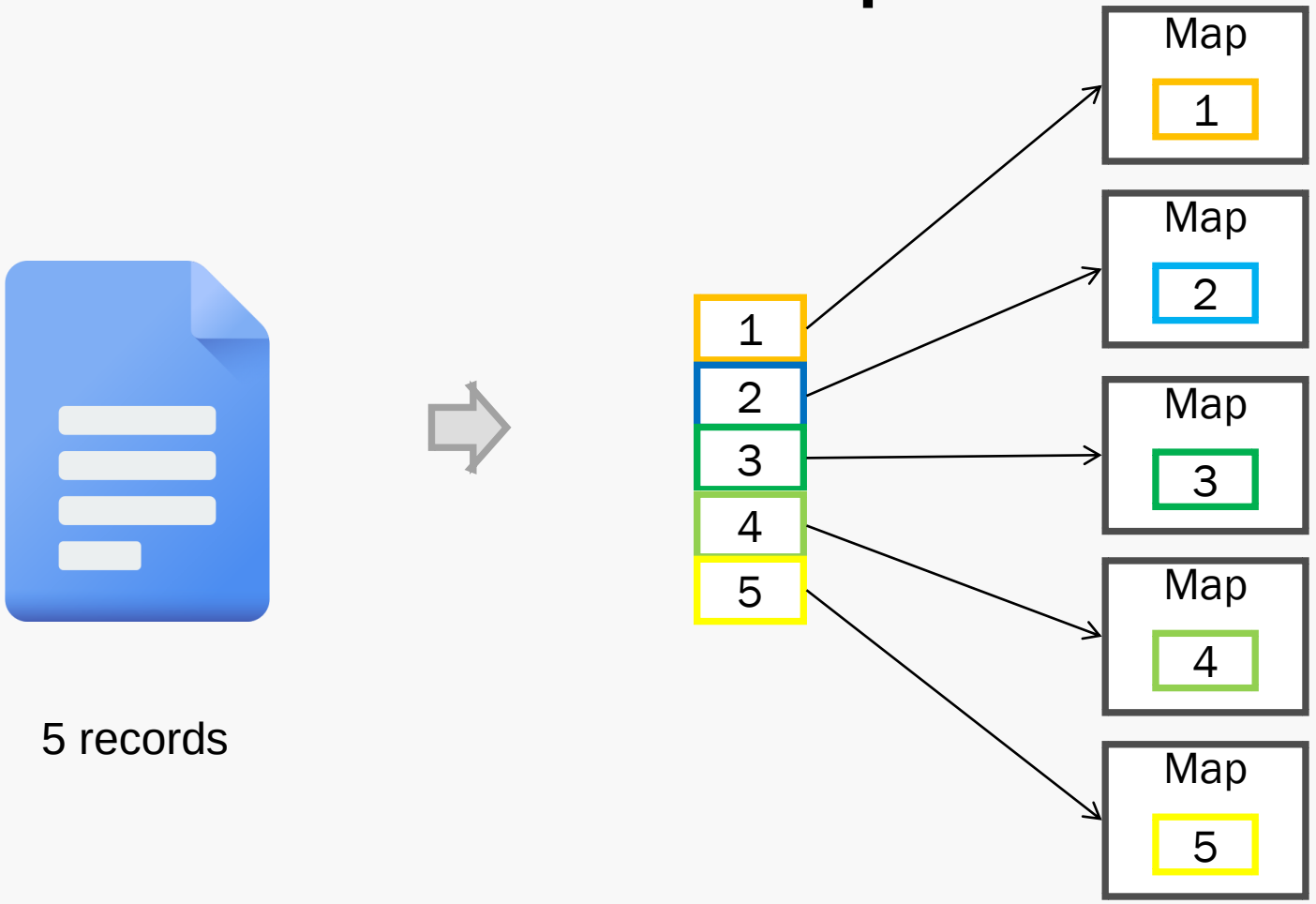
- At the crux of MapReduce are two functions: Map and Reduce. They are sequenced one after the other.
  - *The **Map** function takes input from the disk as <key,value> pairs, processes them, and produces another set of intermediate <key,value> pairs as output.*
  - *The **Reduce** function also takes inputs as <key,value> pairs, and produces <key,value> pairs as output.*

Map

- The input data is first split into smaller blocks. Each block is then assigned to a mapper for processing.

- For example, if a file has 5 records to be processed, 5 mappers can run together to process single records each.

Map



The Hadoop framework decides how many mappers to use, based on the size of the data to be processed and the memory block available on each mapper server.

Reduce

- After all the mappers complete processing, the framework shuffles and sorts the results before passing them on to the reducers.
- A reducer cannot start while a mapper is still in progress.
- All the map output values that have the same key are assigned to a single reducer, which then aggregates the values for that key.

Let’s have a practical view of MapReduce

Property Dataset

	A	B	C	D	E
1	**ID**	**Year**	**Month**	**Type**	**Price**
2	237062	2005	11	Flat	10000000
3	346905	2005	10	Flat	6900000
4	386513	2007	7	House	16500000
5	656161	2007	12	House	43500000
6	841645	2004	11	House	7000000
168448	..	..	..	..	..
168449	17355287	2017	3	House	9000000

Problem: <u>How many times has a particular type of property been sold?</u>

Problem: How many times has a particular type of property been sold?

Input Data

	ID	Year	Month	Type	Price
1.	237062	2005	11	Farm House	10000000
2.	346905	2005	10	Flat	6900000
3.	386513	2007	7	Room	16500000
4.	656161	2007	12	House	43500000
5.	841645	2004	11	Lower Portion	7000000
6.	850762	2007	9	Penthouse	34500000
7.	937975	2007	9	Upper Portion	27000000
...	...	...	...	...	...
168446	17355287	2017	3	House	9000000

Mapping

Farm House, 1

Flat, 1

Room, 1

House, 1

Lower Portion, 1

Penthouse, 1

Upper portion, 1

...

House, 1

Shuffle & Sort

Farm House, 1
Farm House, 1

Flat, 1
Flat, 1

Room, 1
Room, 1

House, 1
House, 1

Lower Portion, 1
Lower Portion, 1

Penthouse, 1
Penthouse, 1

Upper portion, 1
Upper portion, 1

...

Reduce

Farm House, 657

Flat, 38238

Room, 685

House, 105468

Lower Portion, 9229

Penthouse, 395

Upper portion,

...

Implementation in Spyder

property.py

```python
from mrjob.job import MRJob

#mapreduce

class PropertyType(MRJob):

    def mapper(self, key, line):
        ID,Year,Month,Type,Price = line.split(",")
        yield Type, 1

    def reducer(self,Type,occurence):
        yield Type,sum(occurence)

if __name__ == '__main__':
    PropertyType.run()
```

Name	Date Modified
extra.py	11/1/2021 9:48 PM
graphs.ipynb	11/2/2021 12:22 AM
Property.csv	12/15/2021 12:29 AM
property.py	11/1/2021 2:03 AM
PropertyYearSold.py	11/3/2021 6:41 PM

Variable explorer | Help | Plots | Files

Console 3/A

```
In [1]:
```

Implementation in Spyder

property.py

```python
from mrjob.job import MRJob

#mapreduce

class PropertyType(MRJob):

    def mapper(self, key, line):
       ID,Year,Month,Type,Price = line.split(",")
       yield Type, 1

    def reducer(self,Type,occurence):
        yield Type,sum(occurence)

if __name__ == '__main__':
    PropertyType.run()
```

Name	Date Modified
extra.py	11/1/2021 9:48 PM
graphs.ipynb	11/2/2021 12:22 AM
Property.csv	12/15/2021 12:29 AM
property.py	12/15/2021 12:37 AM
PropertyYearSold.py	11/3/2021 6:41 PM

Variable explorer | Help | Plots | Files

Console 3/A

```
In [1]: !python property.py Property.csv
```

Implementation in Spyder

property.py

```python
from mrjob.job import MRJob

#mapreduce

class PropertyType(MRJob):

    def mapper(self, key, line):
        ID,Year,Month,Type,Price = line.split(",")
        yield Type, 1

    def reducer(self,Type,occurence):
        yield Type,sum(occurence)

if __name__ == '__main__':
    PropertyType.run()
```

Name	Date Modified
extra.py	11/1/2021 9:48 PM
graphs.ipynb	11/2/2021 12:22 AM
Property.csv	12/15/2021 12:29 AM
property.py	12/15/2021 12:37 AM
PropertyYearSold.py	11/3/2021 6:41 PM

Variable explorer | Help | Plots | Files

Console 3/A

```
In [1]: !python property.py Property.csv
"FarmHouse" 657
"Flat"  38238
"House" 105468
"LowerPortion"  9229
"Penthouse" 395
"Room"  685
"UpperPortion"  13774
```

Result

Problem: How many times has a particular type of property been sold?

Result:

A number of particular type of property been sold is:

- **Farmhouse** has been sold **657** times
- **Flat** has been sold **38238** times
- **House** has been sold **105468** times
- **Lower Portion** has been sold **9229** times
- **Penthouse** has been sold **395** times
- **Room** has been sold **685** times
- **Upper Portion** has been sold **13774** times

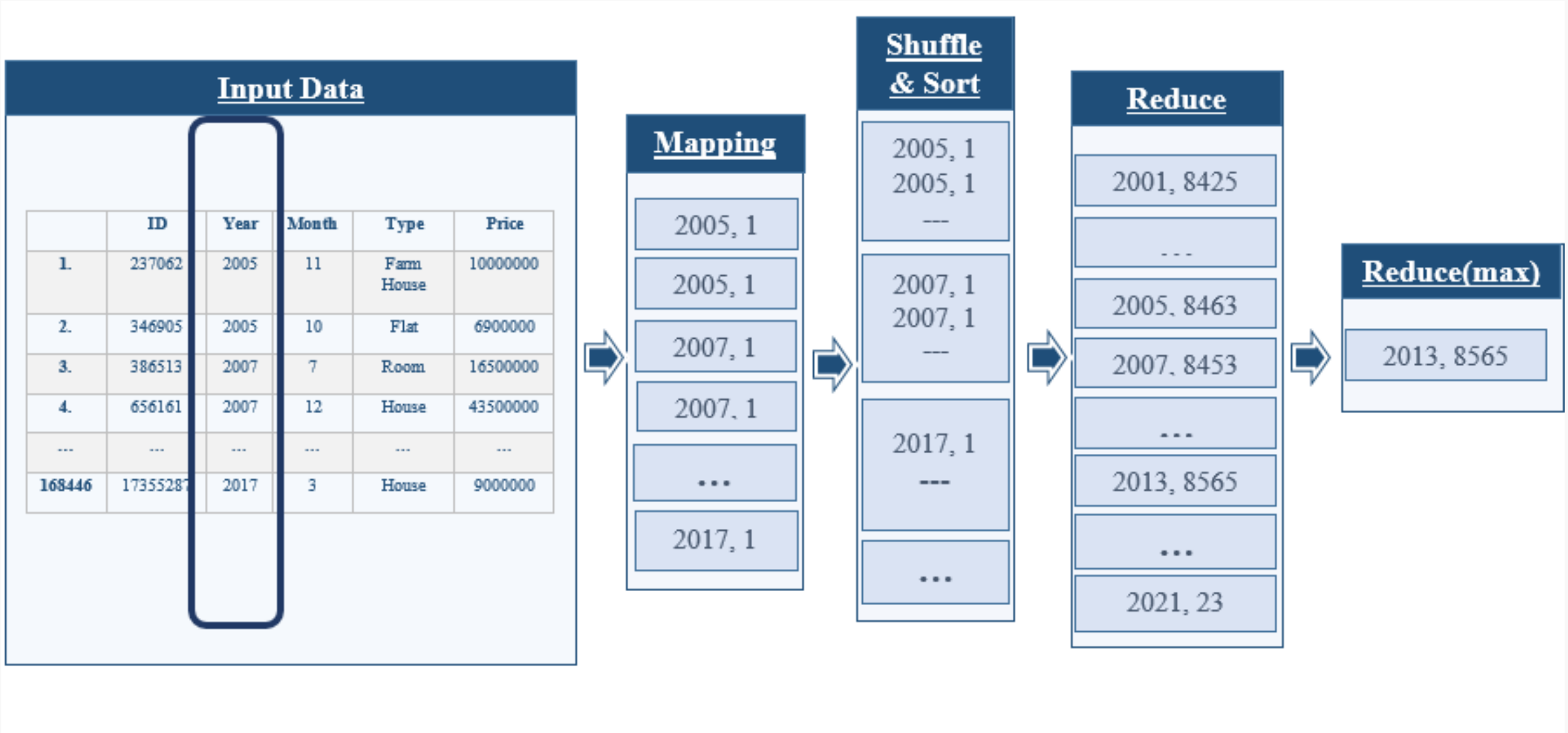
Property Dataset

	A	B	C	D	E
1	ID	Year	Month	Type	Price
2	237062	2005	11	Flat	10000000
3	346905	2005	10	Flat	6900000
4	386513	2007	7	House	16500000
5	656161	2007	12	House	43500000
6	841645	2004	11	House	7000000
168448	..	..	..	..	..
168449	17355287	2017	3	House	9000000

Problem: <u>In what year were most of the properties sold?</u>

1. Count of properties Sold per year?
2. Maximum of that count?

Problem: In what year were most of the properties sold?



	ID	Year	Month	Type	Price
1.	237062	2005	11	Farm House	10000000
2.	346905	2005	10	Flat	6900000
3.	386513	2007	7	Room	16500000
4.	656161	2007	12	House	43500000
...	...	...	...	...	...
168446	1735528[illegible]	2017	3	House	9000000

Implementation in Spyder

PropertyYearSold.py

```python
from mrjob.job import MRJob
from mrjob.step import MRStep
#mapreduce

class MaxPropertyPrice(MRJob):

    def steps(self):
        return [
            MRStep(mapper=self.mapper_get_ratings,
                   reducer=self.reducer_count_ratings),
            MRStep(mapper=self.mapper_passthrough,
                   reducer=self.reducer_find_max)
        ]

    def mapper_get_ratings(self, _, line):
        ID,Year,Month,Type,Price = line.split(",")
        yield Year, 1

    def mapper_passthrough(self, key, value):
        yield key, value

    def reducer_count_ratings(self, key, values):
        yield None, (sum(values),key)

    def reducer_find_max(self, key, values):
        yield max(values)

if __name__ == '__main__':
    MaxPropertyPrice.run()
```

Name	Date Modified
extra.py	11/1/2021 9:48 PM
graphs.ipynb	11/2/2021 12:22 AM
Property.csv	12/15/2021 12:29 AM
PropertyTypeSold.py	12/15/2021 12:37 AM
PropertyYearSold.py	12/15/2021 1:05 AM

Variable explorer | Help | Plots | Files

Console 3/A

```
In [6]:
```

Implementation in Spyder

PropertyYearSold.py

```python
from mrjob.job import MRJob
from mrjob.step import MRStep
#mapreduce

class MaxPropertyPrice(MRJob):

    def steps(self):
        return [
            MRStep(mapper=self.mapper_get_ratings,
                   reducer=self.reducer_count_ratings),
            MRStep(mapper=self.mapper_passthrough,
                   reducer=self.reducer_find_max)
        ]

    def mapper_get_ratings(self, _, line):
        ID,Year,Month,Type,Price = line.split(",")
        yield Year, 1

    def mapper_passthrough(self, key, value):
        yield key, value

    def reducer_count_ratings(self, key, values):
        yield None, (sum(values),key)

    def reducer_find_max(self, key, values):
        yield max(values)

if __name__ == '__main__':
    MaxPropertyPrice.run()
```

Name	Date Modified
extra.py	11/1/2021 9:48 PM
graphs.ipynb	11/2/2021 12:22 AM
Property.csv	12/15/2021 12:29 AM
PropertyTypeSold.py	12/15/2021 12:37 AM
PropertyYearSold.py	12/15/2021 1:05 AM

Variable explorer | Help | Plots | Files

Console 3/A

```
In [6]: !python PropertyYearSold.py Property.csv
```

Implementation in Spyder

PropertyYearSold.py

```python
from mrjob.job import MRJob
from mrjob.step import MRStep
#mapreduce

class MaxPropertyPrice(MRJob):

    def steps(self):
        return [
            MRStep(mapper=self.mapper_get_ratings,
                   reducer=self.reducer_count_ratings),
            MRStep(mapper=self.mapper_passthrough,
                   reducer=self.reducer_find_max)
        ]

    def mapper_get_ratings(self, _, line):
        ID,Year,Month,Type,Price = line.split(",")
        yield Year, 1

    def mapper_passthrough(self, key, value):
        yield key, value

    def reducer_count_ratings(self, key, values):
        yield None, (sum(values),key)

    def reducer_find_max(self, key, values):
        yield max(values)

if __name__ == '__main__':
    MaxPropertyPrice.run()
```

Name	Date Modified
extra.py	11/1/2021 9:48 PM
graphs.ipynb	11/2/2021 12:22 AM
Property.csv	12/15/2021 12:29 AM
PropertyTypeSold.py	12/15/2021 12:37 AM
PropertyYearSold.py	12/15/2021 1:05 AM

Variable explorer | Help | Plots | Files

Console 3/A

```
In [6]: !python PropertyYearSold.py Property.csv
8565    "2013"
```

Result

Problem: In what year were most of the properties sold?

Result

Most of the properties were sold in **2013.**

- **As** 8565 Properties were sold in 2013 which is the maximum count.)

Store Dataset

	A	B	C	D	E	F	G	H	I	J	K
1	OrderDate	Region	City	Category	Product	Quantity	UnitPrice	TotalPrice	Payable_tax	Payable_amount	
2	43577	West	Los Angel	Bars	Carrot	20	1.77	Rs 35.40	2.83	38.23	
3	43628	West	San Diego	Crackers	Whole Wheat	20	3.49	Rs 69.80	5.58	75.38	
4	43760	West	San Diego	Bars	Carrot	20	1.77	Rs 35.40	2.83	38.23	
5	44039	West	Los Angel	Cookies	Arrowroot	20	2.18	Rs 43.60	3.49	47.09	
6	44090	East	New York	Snacks	Potato Chips	20	1.68	Rs 33.60	2.69	36.29	
7	44156	West	San Diego	Bars	Carrot	20	1.77	Rs 35.40	2.83	38.23	
8	44165	East	Boston	Crackers	Whole Wheat	20	3.49	Rs 69.80	5.58	75.38	
9	43598	West	Los Angel	Crackers	Whole Wheat	21	3.49	Rs 73.29	5.86	79.15	
10	43700	East	Boston	Crackers	Whole Wheat	21	3.49	Rs 73.29	5.86	79.15	

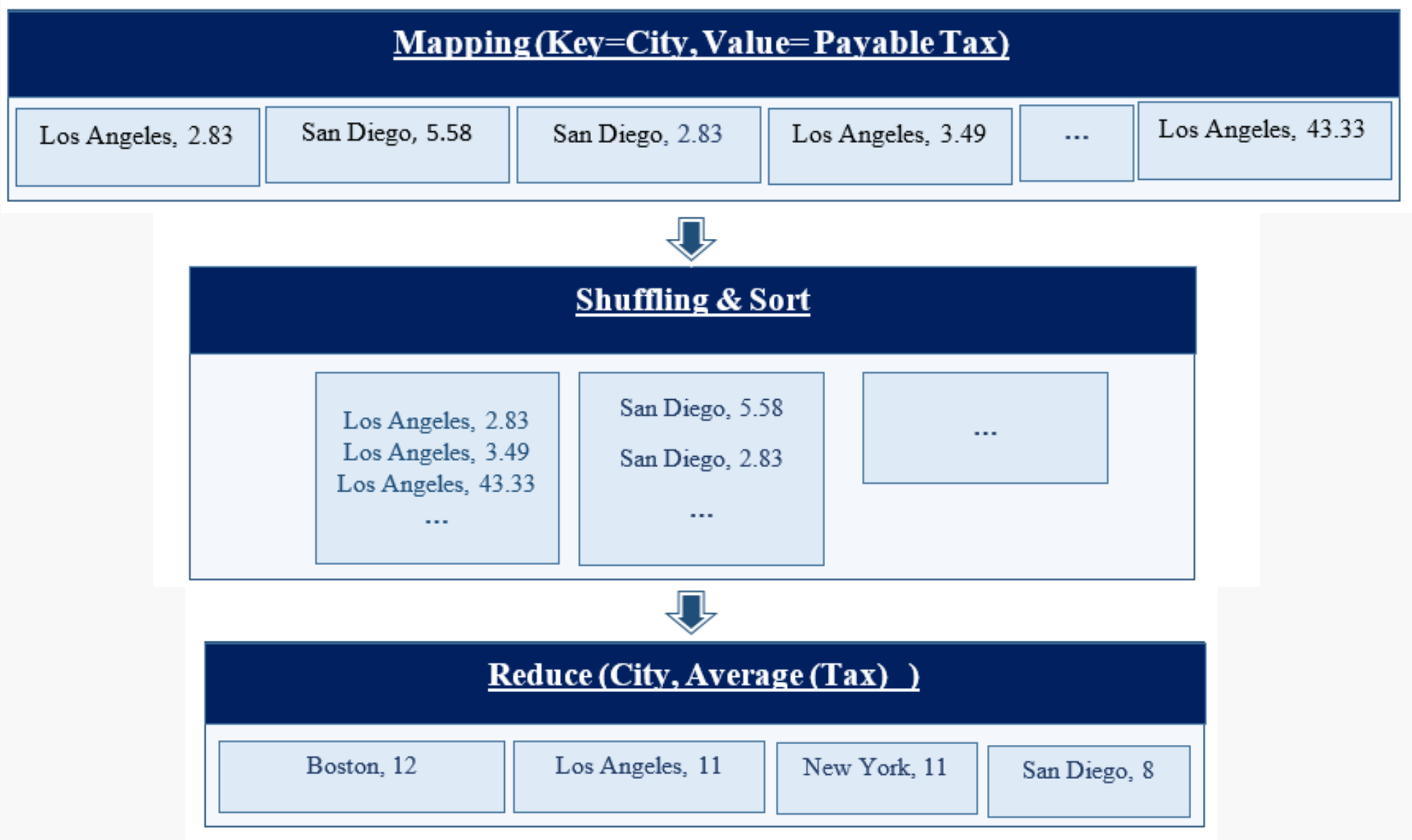
Problem: <u>What is the average tax paid by each City?</u>

Problem: What is the average tax paid by each City?

Input Data

	OrderDate	Region	City	Category	Product	Quantity	UnitPrice	TotalPrice	Payable_tax	Total_Payable_amount
1.	43577	West	Los Angeles	Bars	Carrot	20	1.77	Rs 35.40	2.83	38.23
2.	43628	West	San Diego	Crackers	Whole Wheat	20	3.49	Rs 69.80	5.58	75.38
3.	43760	West	San Diego	Bars	Carrot	20	1.77	Rs 35.40	2.83	38.23
4.	44039	West	Los Angeles	Cookies	Arrowroot	20	2.18	Rs 43.60	3.49	47.09
...	...									
244	43637	West	Los Angeles	Bars	Carrot	306	1.77	Rs 541.62	43.33	584.95

Problem: What is the average tax paid by each City?



Implementation in Spyder

AverageTax.py

```python
from mrjob.job import MRJob


class AverageTax(MRJob):

    def mapper(self, key, line):
        OrderDate,Region,City,Category,Product,Quantity,UnitPrice,TotalPrice,\
            Payable_tax,Total_Payable_amount = line.split(",")
        yield City, float(Payable_tax)

    def reducer(self,City,tax):
        num=0
        total=0
        for i in tax:
            num=num+1
            total=total+i

        yield City,round(total/num)

if __name__ == '__main__':
    AverageTax.run()
```

Name	Date Modified
AverageTax.py	11/3/2021 5:10 PM
QuantityPerRegion.py	11/3/2021 5:20 PM
Store.csv	12/15/2021 1:44 AM

Variable explorer | Help | Plots | Files

Console 4/A

```
In [1]:
```

Implementation in Spyder

AverageTax.py

```python
from mrjob.job import MRJob


class AverageTax(MRJob):

    def mapper(self, key, line):
        OrderDate,Region,City,Category,Product,Quantity,UnitPrice,TotalPrice,\
            Payable_tax,Total_Payable_amount = line.split(",")
        yield City, float(Payable_tax)

    def reducer(self,City,tax):
        num=0
        total=0
        for i in tax:
            num=num+1
            total=total+i

        yield City,round(total/num)

if __name__ == '__main__':
    AverageTax.run()
```

Name	Date Modified
AverageTax.py	11/3/2021 5:10 PM
QuantityPerRegion.py	11/3/2021 5:20 PM
Store.csv	12/15/2021 1:44 AM

Variable explorer | Help | Plots | Files

Console 4/A

```
In [1]: !python AverageTax.py Store.csv
```

Implementation in Spyder

AverageTax.py

```python
from mrjob.job import MRJob


class AverageTax(MRJob):

    def mapper(self, key, line):
        OrderDate,Region,City,Category,Product,Quantity,UnitPrice,TotalPrice,\
            Payable_tax,Total_Payable_amount = line.split(",")
        yield City, float(Payable_tax)

    def reducer(self,City,tax):
        num=0
        total=0
        for i in tax:
            num=num+1
            total=total+i

        yield City,round(total/num)

if __name__ == '__main__':
    AverageTax.run()
```

Name	Date Modified
AverageTax.py	11/3/2021 5:10 PM
QuantityPerRegion.py	11/3/2021 5:20 PM
Store.csv	12/15/2021 1:44 AM

Variable explorer | Help | Plots | Files

Console 4/A

```
In [1]: !python AverageTax.py Store.csv
"Boston"    12
"Los Angeles"   11
"New York"  11
"San Diego" 8
```

Result

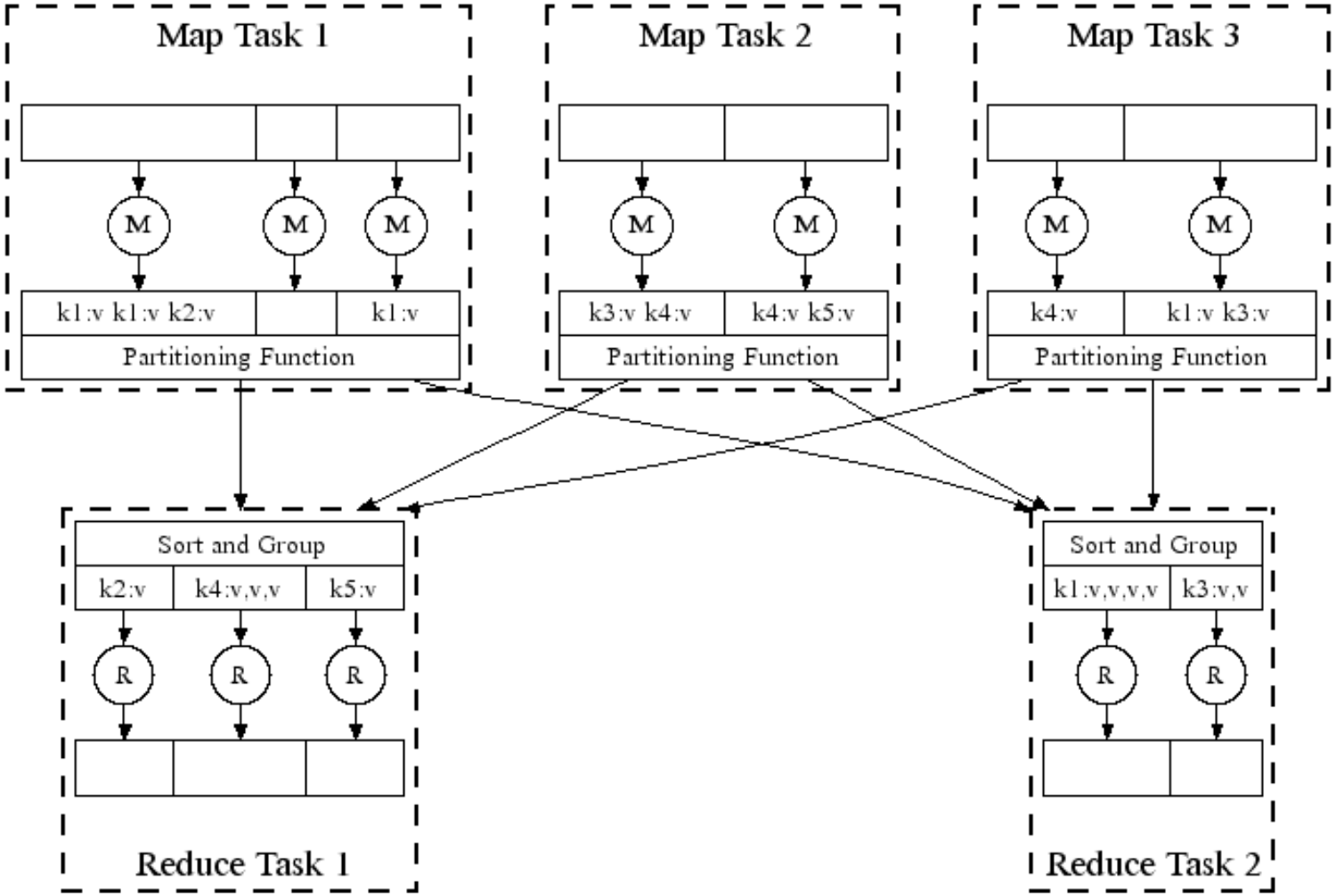
Problem: <u>What is the average tax paid by each City?</u>

<u>Result</u>

The average tax paid by each City is given below:

- **Boston** has paid average tax of **12.**
- **Los Angeles** has paid average tax of **11.**
- **New York** has paid average tax of **11.**
- **San Diego** has paid average tax of **8.**

How it is related to Big Data?



Let’s Discuss a Problem

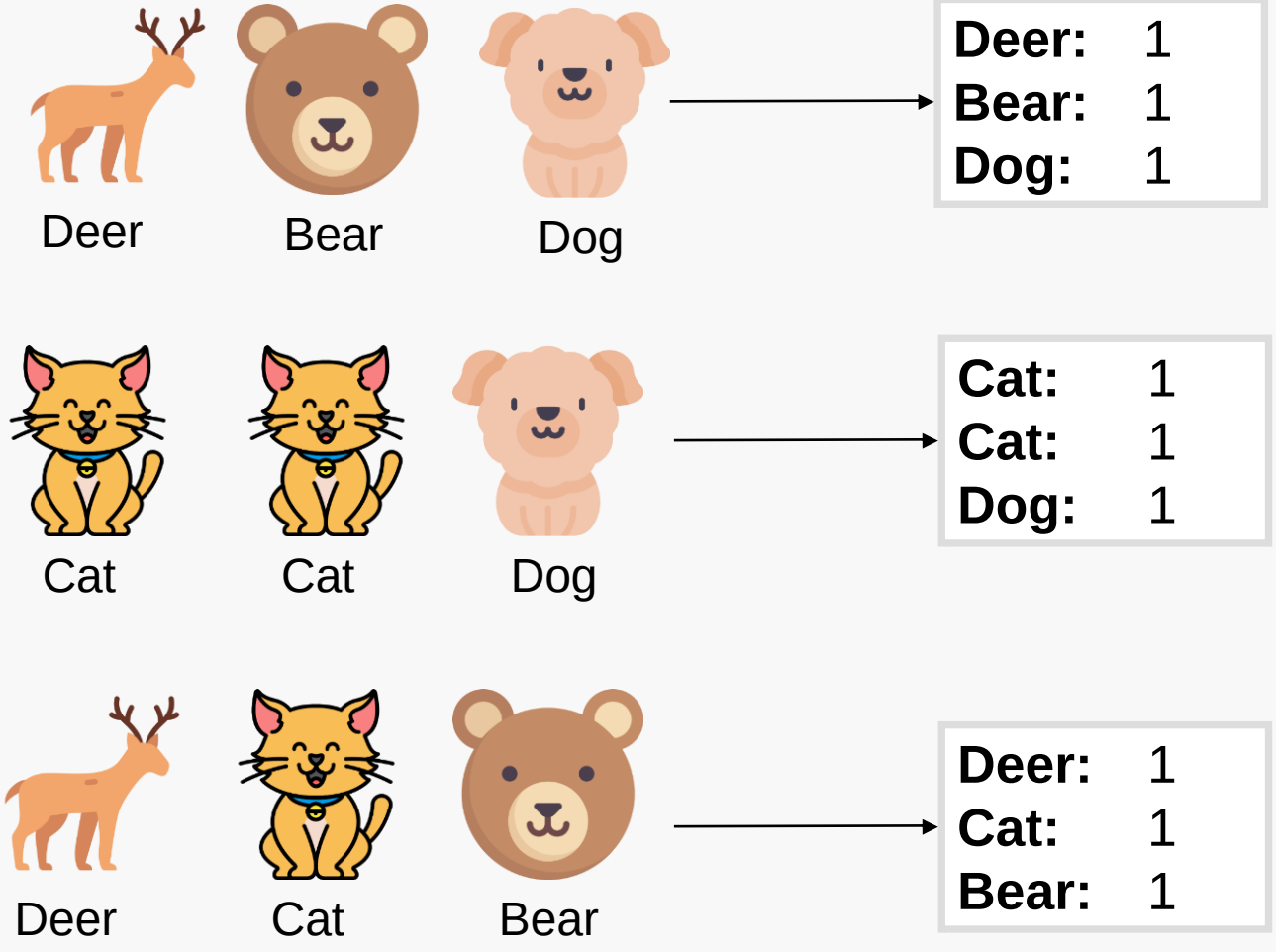
Problem: Word Count

- **Statement:** Different observers saw different animals in forest. Count the occurrence of each animal, use Map-Reduce to the count (occurrence) of words.

Deer, Bear, Dog
Cat, Cat, Dog
Deer, Cat, Bear

Question: What are the Mapper and Reducer Functions?

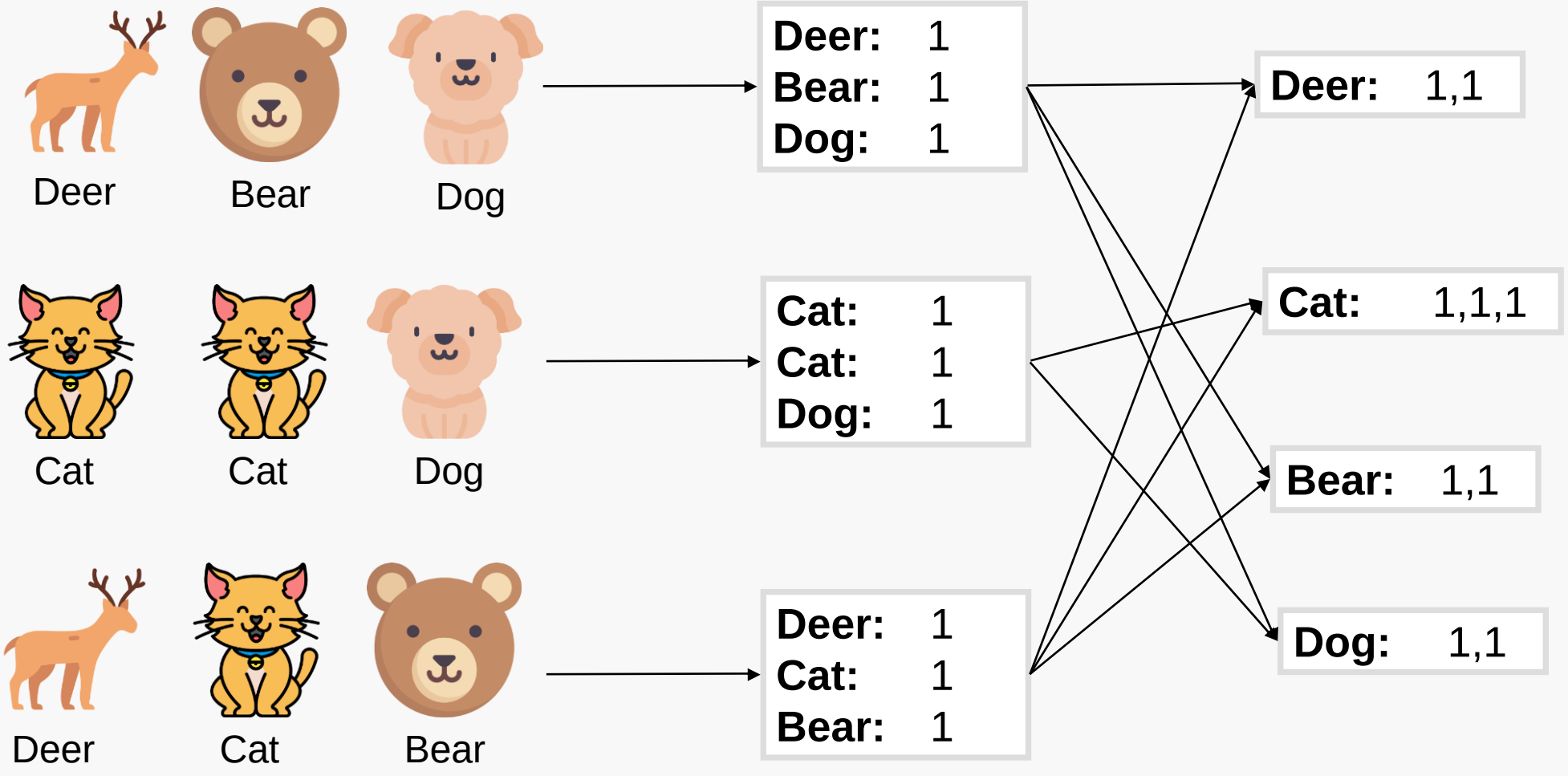
Mapping



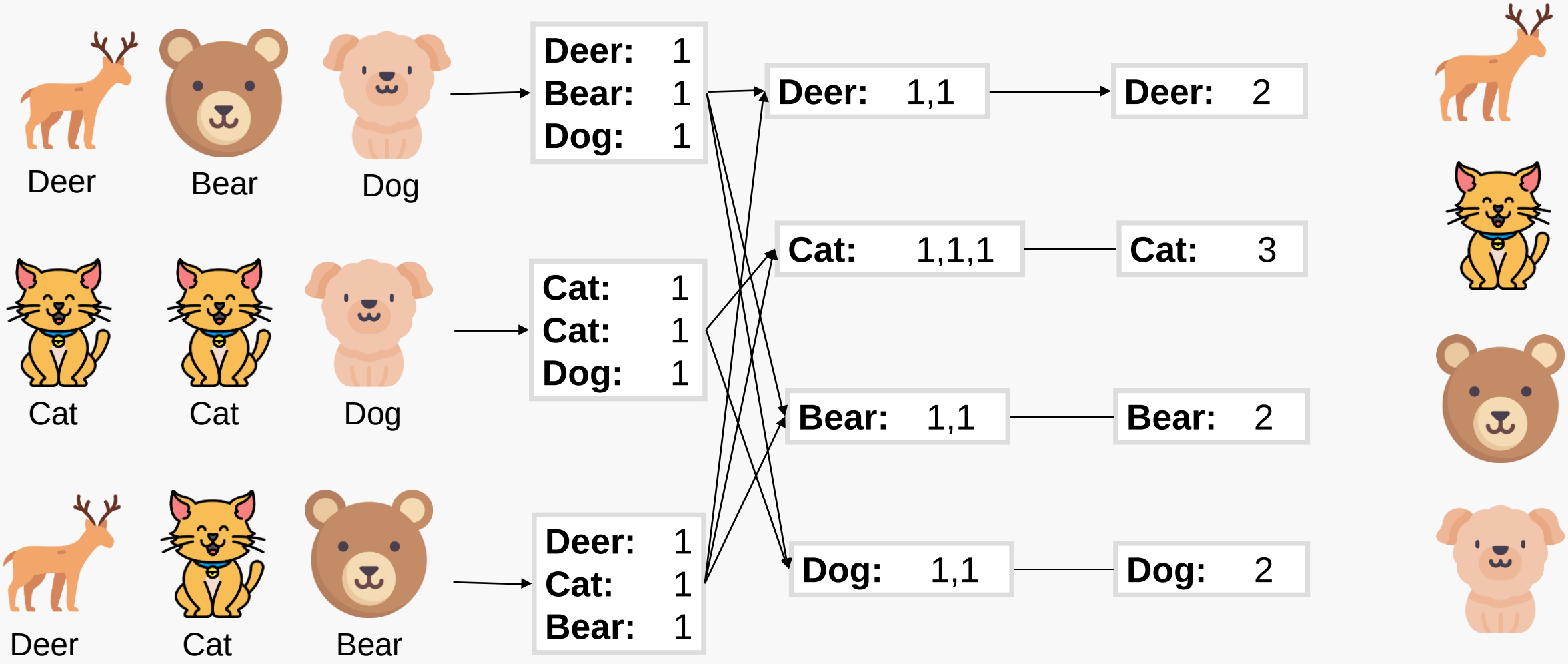
MAP

(key, value)
(animal, occurrence)

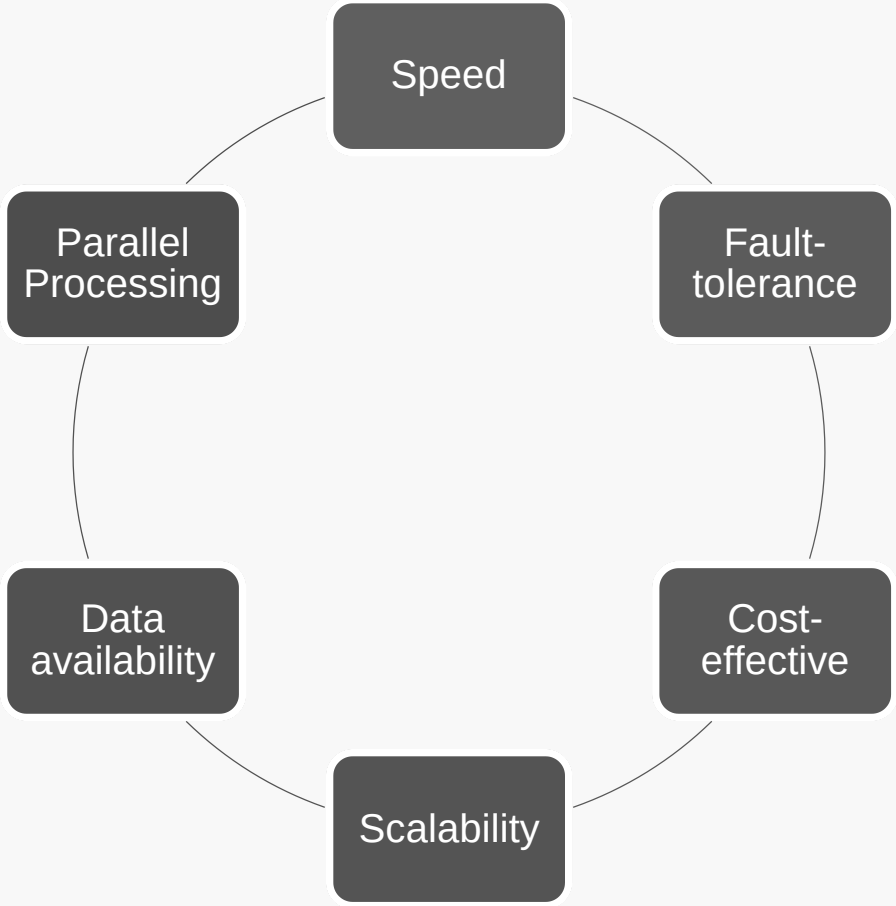
Shuffling and Sort



Reduce


Deer
Bear
Dog
Cat
Cat
Dog
Deer
Cat
Bear
Deer: 1
Bear: 1
Dog: 1
Cat: 1
Cat: 1
Dog: 1
Deer: 1
Cat: 1
Bear: 1
Deer: 1,1
Cat: 1,1,1
Bear: 1,1
Dog: 1,1
Deer: 2
Cat: 3
Bear: 2
Dog: 2

Benefits of Hadoop MapReduce



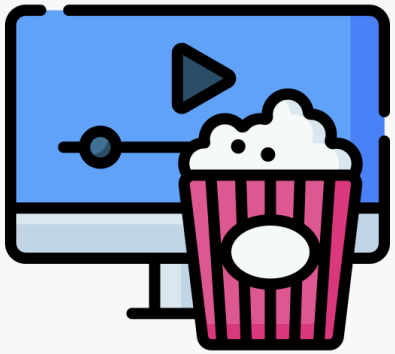
Used-in

- Index and Search
- Classification
- Recommendation system
- Analytics

Applications of MapReduce

E-Commerce

Social Media



Entertainment

Summary

- MapReduce is a crucial processing component of the Hadoop framework. It’s a quick, scalable, and cost-effective program that can help data analysts and developers process huge data.
- This programming model is a suitable tool for analyzing usage patterns on websites and e-commerce platforms. Companies providing online services can utilize this framework to improve their marketing strategies.


Commonly used
The original Hadoop processing engine which is primarily Java based
hadoop
MapReduce
An extensive and mature fault tolerance framework
Based on the map and reduce programming model

Limitation

Map Reduce is bad for:

- Frequently changing data
- Dependent
- Interactive Analytics

References

- **Fundamentals of MapReduce with MapReduce Example**

https://medium.com/edureka/mapreduce-tutorial-3d9535ddbe7c

- **History and Advantages of Hadoop MapReduce Programming**

https://mindmajix.com/mapreduce/history-and-advantages-of-hadoop-mapreduce-programming

- **What is MapReduce?**

https://www.talend.com/resources/what-is-mapreduce/

- **MapReduce:**
  Simplified Data Processing on Large Clusters

https://research.google.com/archive/mapreduce-osdi04-slides/index.html

Thank You!