Lecture 2

Big Data Processing Techniques (MapReduce)

Dr. Lydia Wahid

Agenda

MapReduce Definition

MapReduce Algorithm

MapReduce Examples

MapReduce Definition

MapReduce Definition

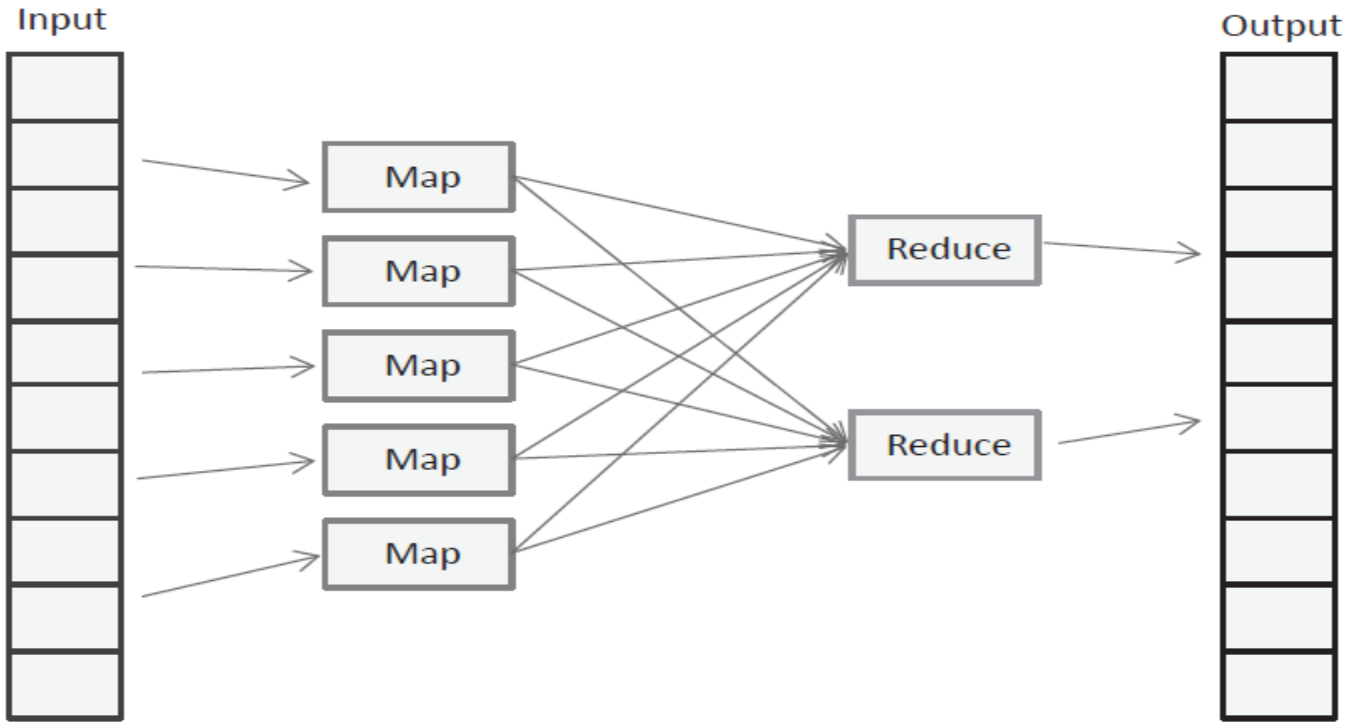
- MapReduce is a widely used Big data processing technique.
- It processes large datasets using **parallel processing** deployed over clusters of hardware.
- It is based on the principle of **divide-and-conquer**. It divides a big problem into a collection of smaller problems that can each be solved quickly.
- A **dataset is broken down** into multiple smaller parts, and operations are performed on each part independently and in parallel.
- The results from all operations are then **combined** to arrive at the result of the whole dataset.

MapReduce Definition

- Each MapReduce job is composed of a **map phase** and a **reduce phase** and each phase consists of multiple stages.
- The Map and Reduce phases run **sequentially** in a cluster.
- The Map phase is executed first then the Reduce phase.
- The output of the Map phase becomes the input of the Reduce phase.
- MapReduce does not require that the input data conform to any particular data model.

MapReduce Definition

- The following figure shows the data flow in MapReduce:

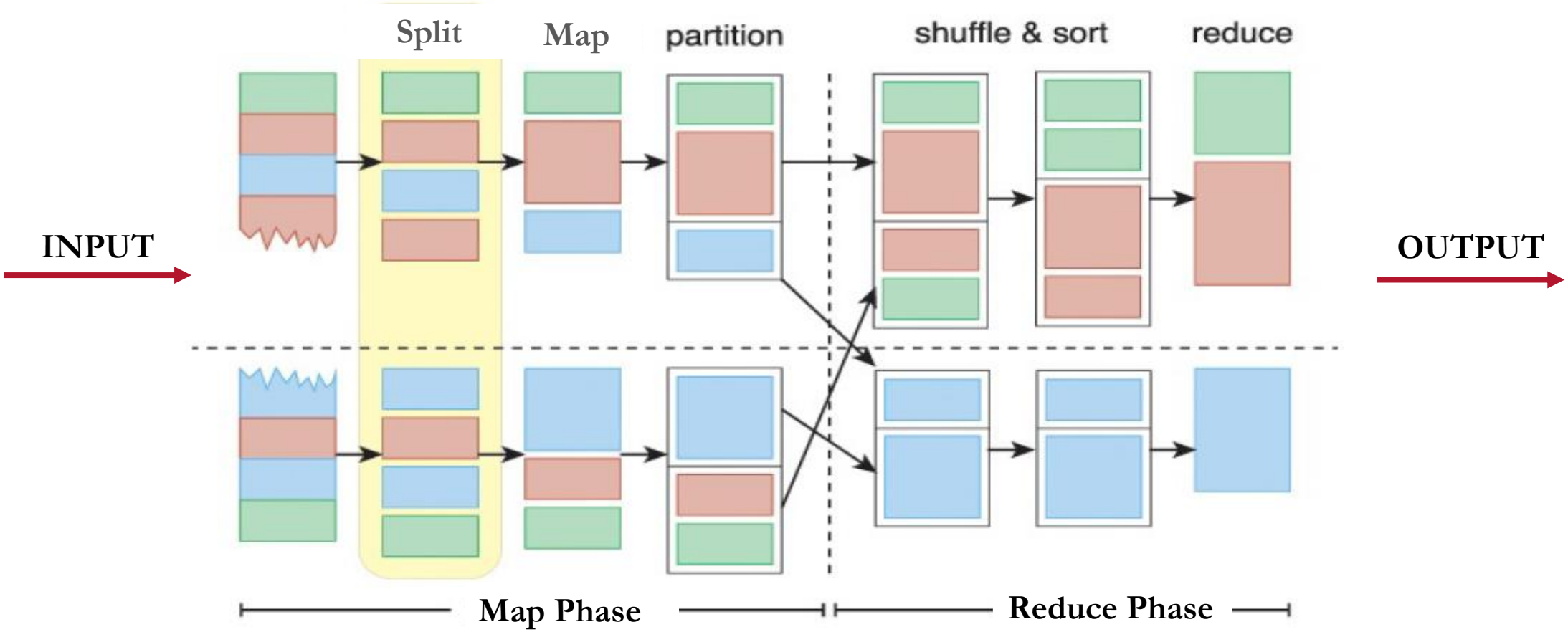


MapReduce Definition

- In MapReduce, all map and reduce tasks run in parallel.
- First of all, all map tasks are independently run.
- Meanwhile, reduce tasks wait until their respective maps are finished.
- Then, reduce tasks process their data concurrently and independently.

MapReduce Algorithm

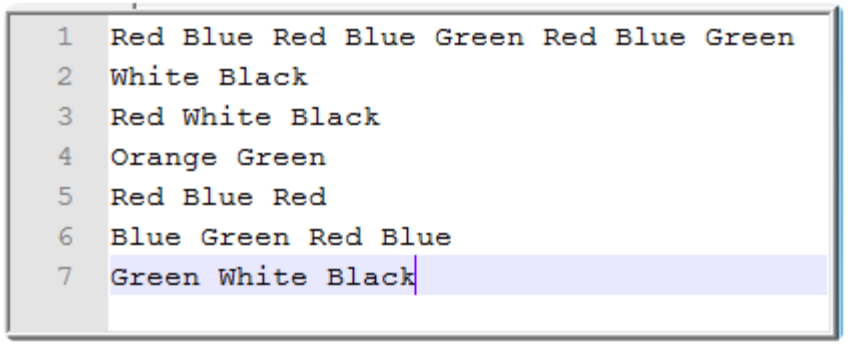
MapReduce Algorithm



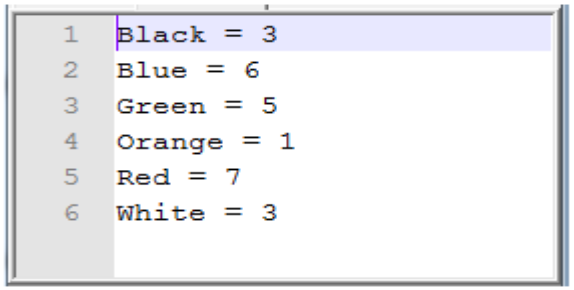
MapReduce Algorithm

- We will now apply and explain each stage on the following example:
  - **Problem Statement:**
    Count the number of occurrences of each word available in a DataSet.

Input Dataset



Required Output



MapReduce Algorithm

- Split stage:
  - Takes input DataSet and divides it into smaller Sub-DataSets called splits.
  - Each split is parsed into its constituent records as a key-value pair. The key is usually the ordinal position of the record, and the value is the actual record.
  - A common example will read a directory full of text files and return each line as a record.
  - The key-value pairs for each split are then sent to a map function (or mapper).
- By applying this stage on our example, we get the following:

MapReduce Algorithm

```
1  Red Blue Red Blue Green Red Blue Green
2  White Black
3  Red White Black
4  Orange Green
5  Red Blue Red
6  Blue Green Red Blue
7  Green White Black
```

Input Dataset

Split stage

```
1  Sub-DataSet-1
2  ----------------
3  Red Blue Red Blue Green Red Blue Green
4  White Black
5  Red White Black
```

```
1  Sub-DataSet-2
2  ----------------
3  Orange Green
4  Red Blue Red
5  Blue Green Red Blue
6  Green White Black
```

MapReduce Algorithm

- Map stage:
  - This is the **map function** or **mapper** that executes **user-defined logic**.
  - The mapper processes each key-value pair as per the user-defined logic and further generates a key-value pair as its output.
  - The output key can either be the same as the input key or a substring value from the input value, or another user-defined object.
  - Similarly, the output value can either be the same as the input value or a substring value from the input value, or another user-defined object.
  - When all records of the split have been processed, the output is a list of key-value pairs where multiple key-value pairs can exist for the same key.
- By applying this stage on our example, we get the following:

MapReduce Algorithm

```
1  Sub-DataSet-1
2  ----------------
3  Red Blue Red Blue Green Red Blue Green
4  White Black
5  Red White Black
```

```
1  Sub-DataSet-2
2  ----------------
3  Orange Green
4  Red Blue Red
5  Blue Green Red Blue
6  Green White Black
```

Map stage (mapper)

```
1  Sub-DataSet-1
2  ----------------
3  Red = 1
4  Blue = 1
5  Red = 1
6  Blue = 1
7  Green = 1
8  Red = 1
9  Blue = 1
10 Green = 1
11 White = 1
12 Black = 1
13 Red = 1
14 White = 1
15 Black = 1
```

```
1  Sub-DataSet-2
2  ----------------
3  Orange = 1
4  Green = 1
5  Red = 1
6  Blue = 1
7  Red = 1
8  Blue = 1
9  Green = 1
10 Red = 1
11 Blue = 1
12 Green = 1
13 White = 1
14 Black = 1
```

MapReduce Algorithm

- Partition stage:
  - During the partition stage, if more than one reducer is involved, a partitioner divides the output from the mapper into partitions between reducer instances.
  - The number of partitions will be equal to the number of reducers.
  - **All records for a particular key are assigned to the same partition**.
  - The MapReduce algorithm guarantees a random and fair distribution between reducers while making sure that all of the same keys across multiple mappers end up with the same reducer.
- Assume here in our example, that we have only one reducer.

MapReduce Algorithm

- Shuffle and Sort stage:
  - During the first stage of the reduce task, output from all partitioners is copied across the network to the nodes running the reduce task. This is known as **shuffling**.
  - The output list of key-value pairs from each partitioner can contain the same key multiple times, so **merging** and **sorting** of the key-value pairs is done according to the keys so that the output contains a sorted list of all input keys and their values with the same keys appearing together.
  - This merge creates a single key-value pair per group, where key is the group key and the value is the list of all group values.
  - The way in which keys are merged and sorted can be *customized*.
- By applying this stage on our example, we get the following:

MapReduce Algorithm

```
Sub-DataSet-1
-----------------
Red = 1
Blue = 1
Red = 1
Blue = 1
Green = 1
Red = 1
Blue = 1
Green = 1
White = 1
Black = 1
Red = 1
White = 1
Black = 1
```

```
Sub-DataSet-2
-----------------
Orange  = 1
Green  = 1
Red  = 1
Blue  = 1
Red  = 1
Blue  = 1
Green  = 1
Red  = 1
Blue  = 1
Green  = 1
White  = 1
Black = 1
```

Shuffling and Sorting stage

```
DataSet
-----------------
Black = {1,1,1}
Blue = {1,1,1,1,1,1}
Green = {1,1,1,1,1}
Orange  = {1}
Red = {1,1,1,1,1,1,1}
White = {1,1,1}
```

MapReduce Algorithm

- Reduce stage:
  - Reduce is the final stage of the reduce phase.
  - Depending on the **user-defined logic** specified in the **reduce function** or **reducer**, the reducer will either further summarize its input or will emit the output without making any changes.
  - The output key can either be the same as the input key or a substring value from the input value, or another user-defined object.
  - The output value can either be the same as the input value or a substring value from the input value, or another user-defined object.
- By applying this stage on our example, we get the following:

MapReduce Algorithm

```
DataSet
----------------
Black = {1,1,1}
Blue = {1,1,1,1,1,1}
Green = {1,1,1,1,1}
Orange  = {1}
Red = {1,1,1,1,1,1,1}
White = {1,1,1}
```

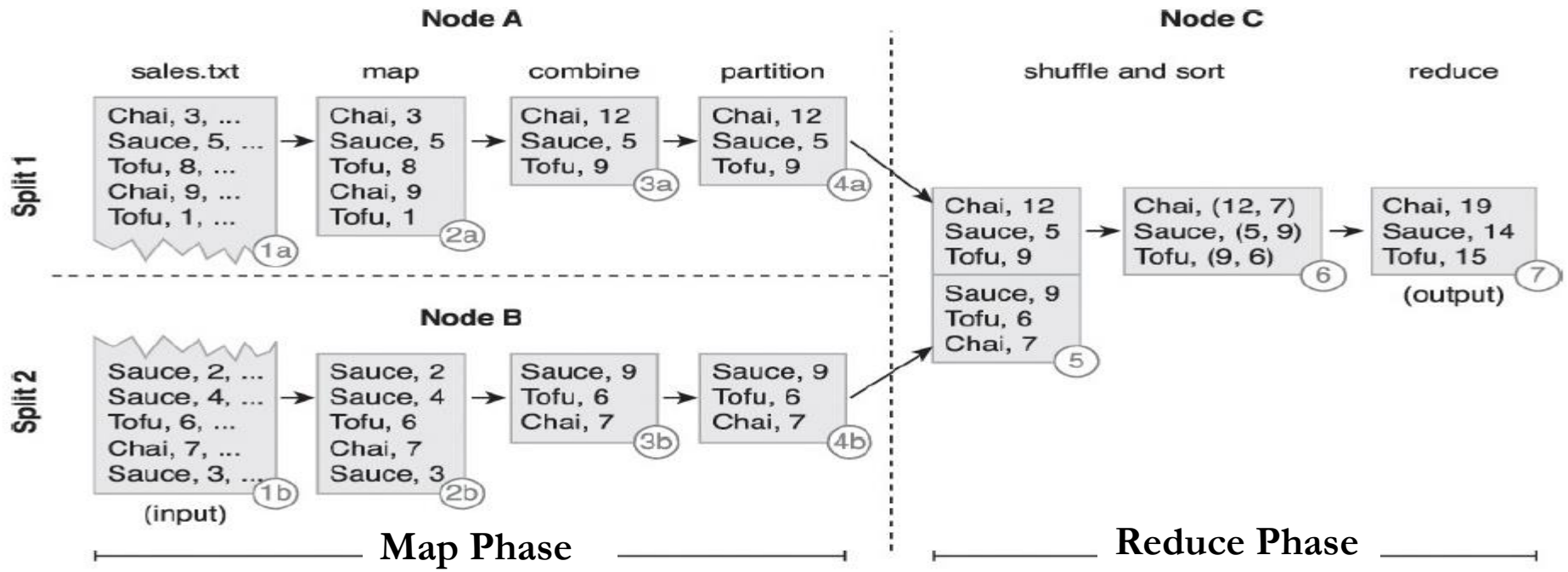
Reduce stage (reducer)

```
Black = 3
Blue = 6
Green = 5
Orange = 1
Red = 7
White = 3
```

Final Output

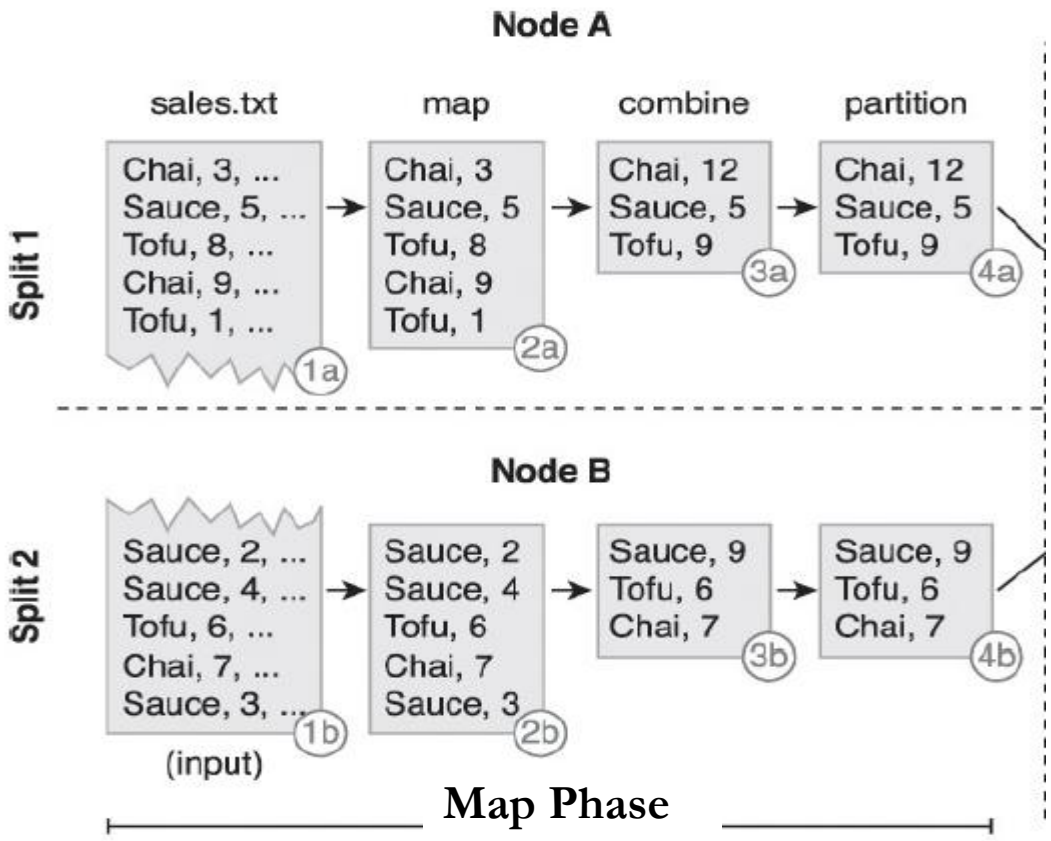
MapReduce Algorithm

- Consider another example as follows:
  - We have products information as input and we need as output the quantity of each product.



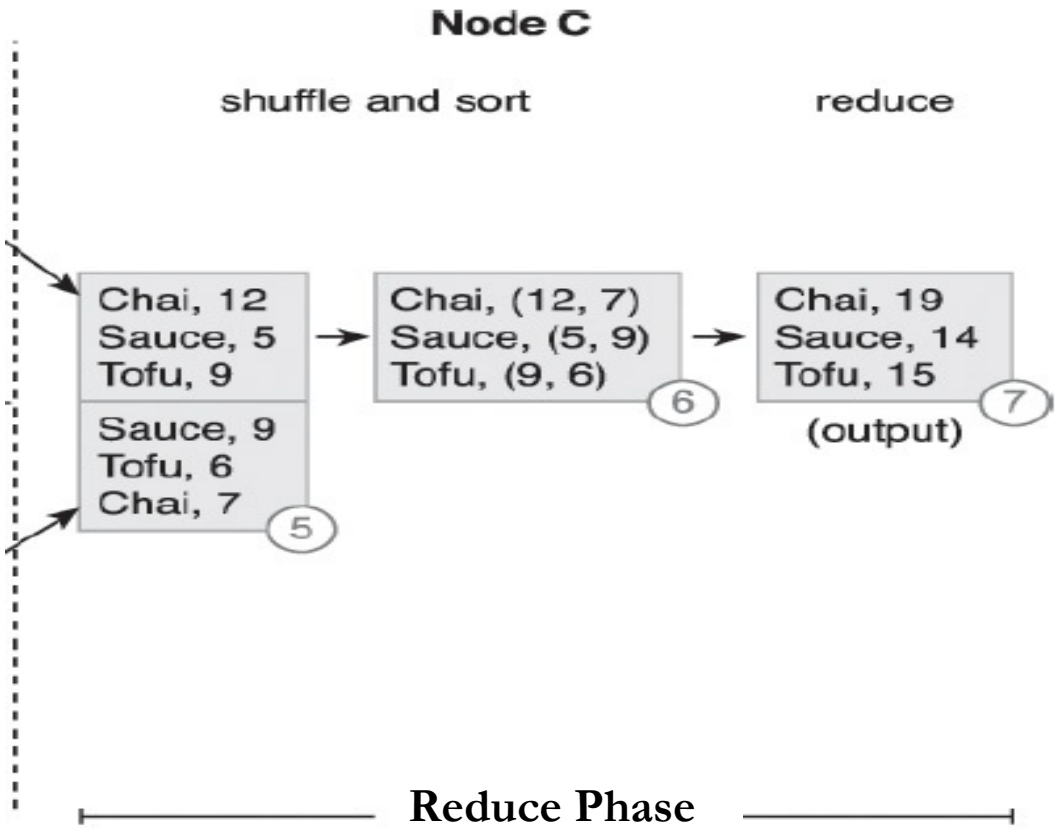
MapReduce Algorithm

1. The input (sales.txt) is divided into two splits.
2. Two map tasks running on two different nodes, Node A and Node B, extract product and quantity from the respective split's records in parallel. The output from each map function is a key-value pair where product is the key while quantity is the value.
3. The **combiner** then performs local summation of product quantities. (A combiner is essentially a reducer function that locally groups a mapper's output on the same node as the mapper.)
4. As there is only one reduce task, no partitioning is performed.



MapReduce Algorithm

5. The output from the two map tasks is then copied to a third node, Node C, that runs the shuffle stage as part of the reduce task.
6. The sort stage then groups all quantities of the same product together as a list.
7. The reduce function then sums up the quantities of each unique product in order to create the output.



MapReduce Examples

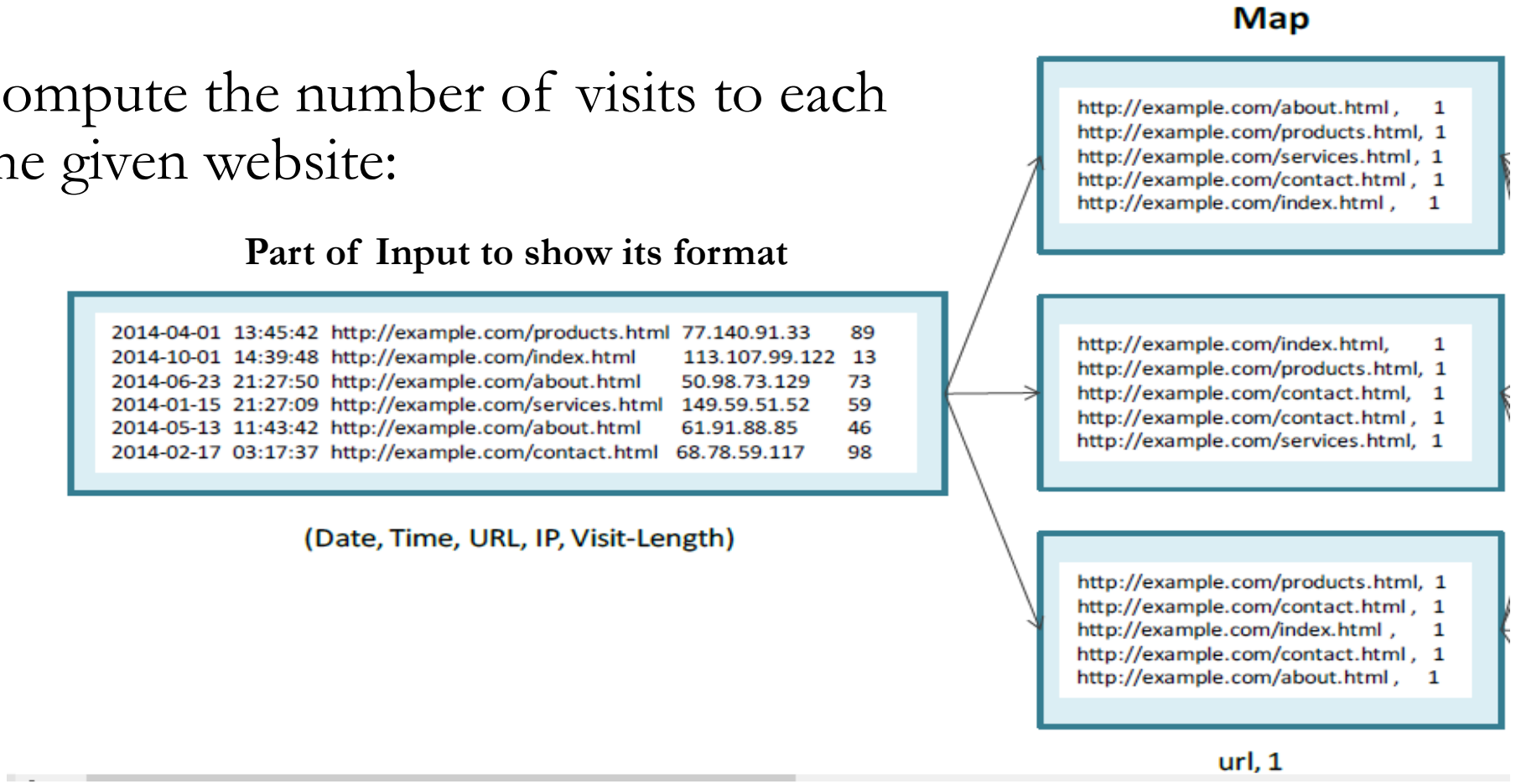
MapReduce Examples

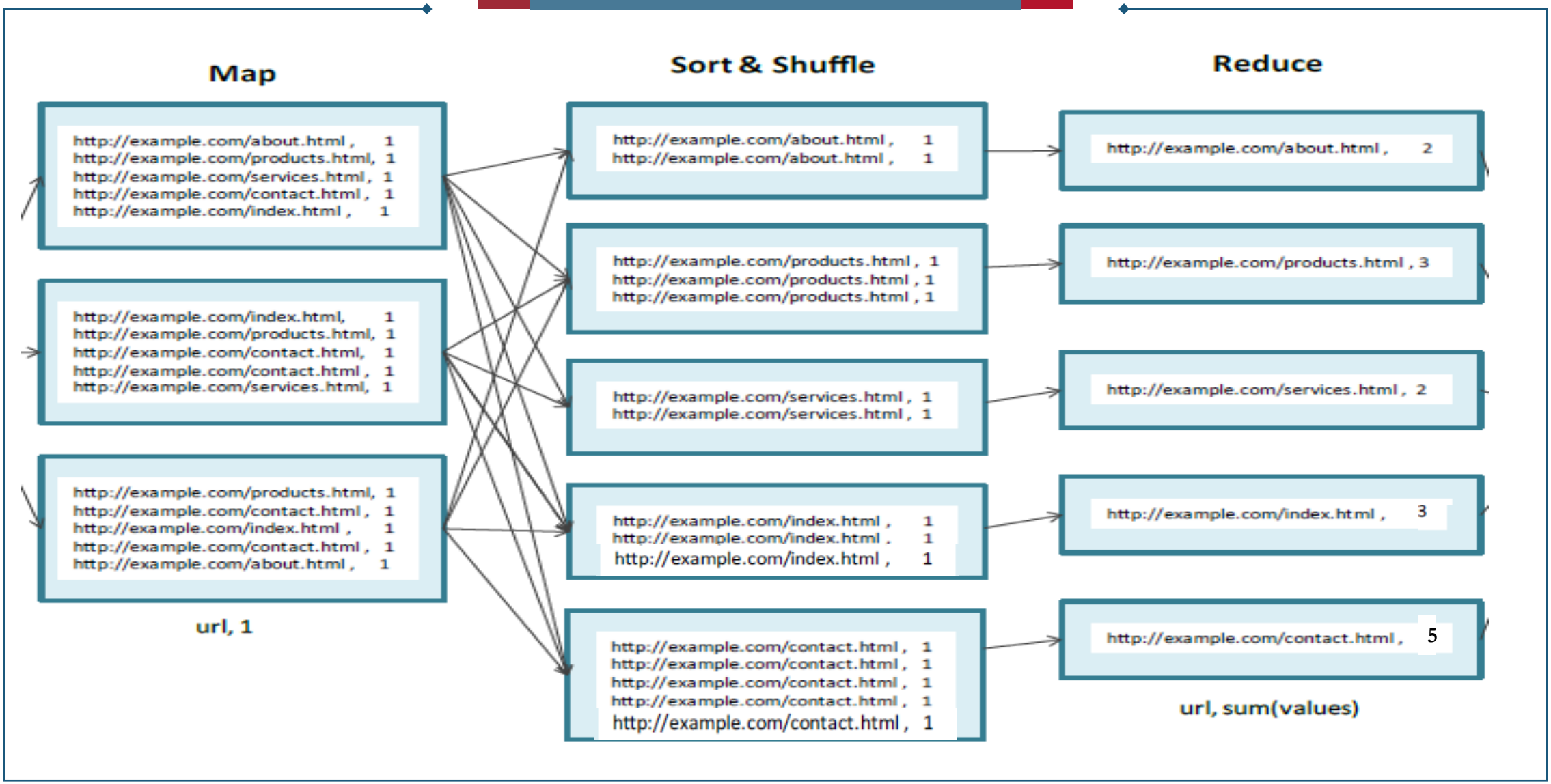
- For the examples in this section, we will use data similar to the data collected by a web analytics service that shows various statistics for page visits for a website.
- Each page has some tracking code which sends the visitor's IP address along with a timestamp to the web analytics service. The web analytics service keeps a record of all page visits and the visitor IP addresses and uses MapReduce programs for computing various statistics.
- Each visit to a page is logged as one row in the log. The log file contains the following columns:

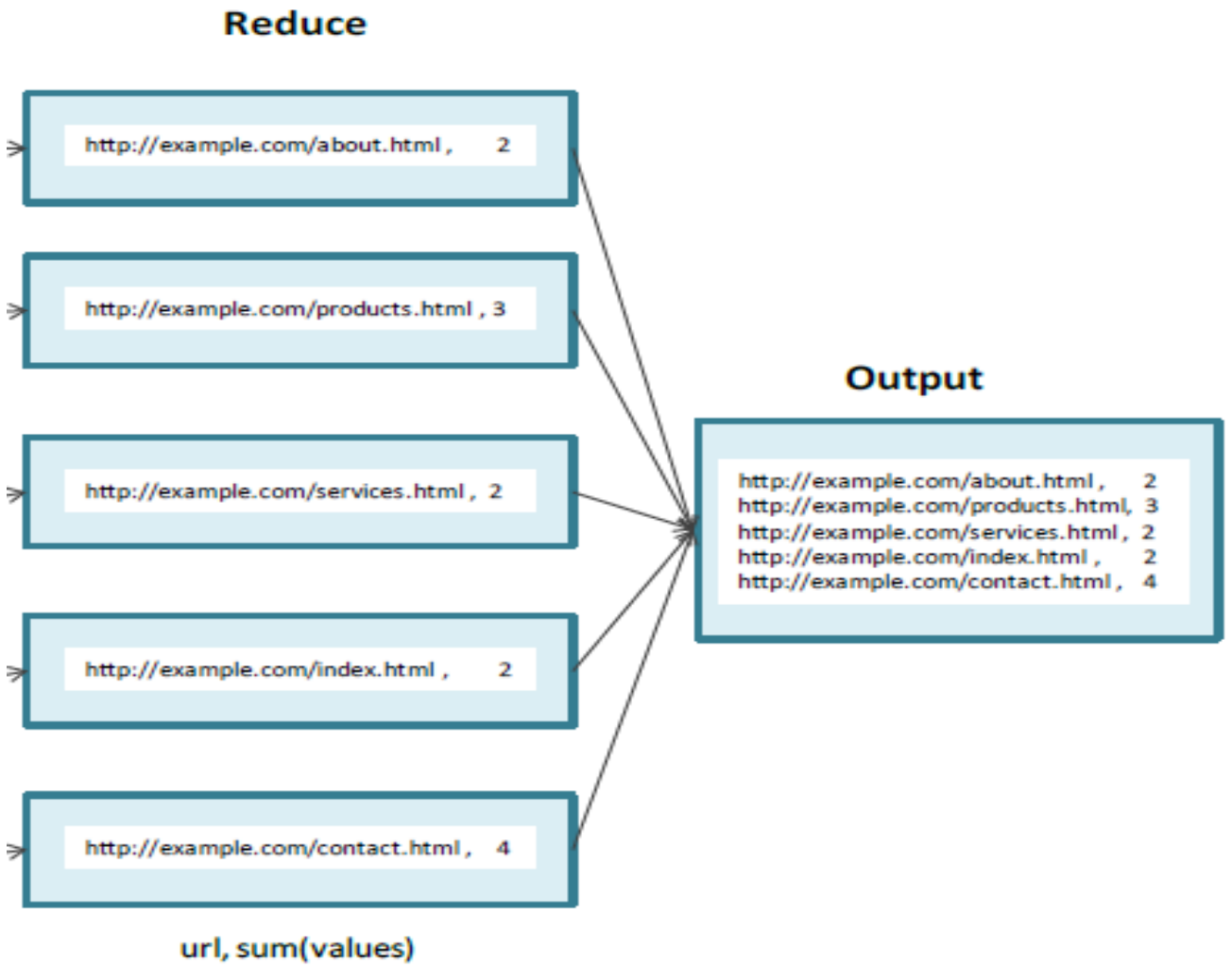
  Date (YYYY-MM-DD), Time (HH:MM:SS), URL, IP, Visit-Length.

MapReduce Examples

1. **Count:** Compute the number of visits to each page of the given website:




Map
http://example.com/about.html , 1
http://example.com/products.html, 1
http://example.com/services.html, 1
http://example.com/contact.html , 1
http://example.com/index.html , 1
http://example.com/index.html, 1
http://example.com/products.html, 1
http://example.com/contact.html, 1
http://example.com/contact.html , 1
http://example.com/services.html, 1
http://example.com/products.html, 1
http://example.com/contact.html , 1
http://example.com/index.html , 1
http://example.com/contact.html , 1
http://example.com/about.html , 1
url, 1
Sort & Shuffle
http://example.com/about.html , 1
http://example.com/about.html , 1
http://example.com/products.html , 1
http://example.com/products.html , 1
http://example.com/products.html , 1
http://example.com/services.html , 1
http://example.com/services.html , 1
http://example.com/index.html , 1
http://example.com/index.html , 1
http://example.com/index.html , 1
http://example.com/contact.html , 1
http://example.com/contact.html , 1
http://example.com/contact.html , 1
http://example.com/contact.html , 1
http://example.com/contact.html , 1
Reduce
http://example.com/about.html , 2
http://example.com/products.html , 3
http://example.com/services.html , 2
http://example.com/index.html , 3
http://example.com/contact.html , 5
url, sum(values)


Reduce
http://example.com/about.html , 2
http://example.com/products.html , 3
http://example.com/services.html , 2
http://example.com/index.html , 2
http://example.com/contact.html , 4
url, sum(values)
Output
http://example.com/about.html , 2
http://example.com/products.html, 3
http://example.com/services.html , 2
http://example.com/index.html , 2
http://example.com/contact.html , 4

MapReduce Examples

1. **Count computation Explanation:**
   - To compute count, the mapper function emits key-value pairs where the key is the field to group-by.
   - The mapper function in this example parses each line of the input and emits key-value pairs where the key is the URL and value is ‘1’.
   - The reducer function receives the key-value pairs grouped by the same key and adds up the values for each group to compute count.

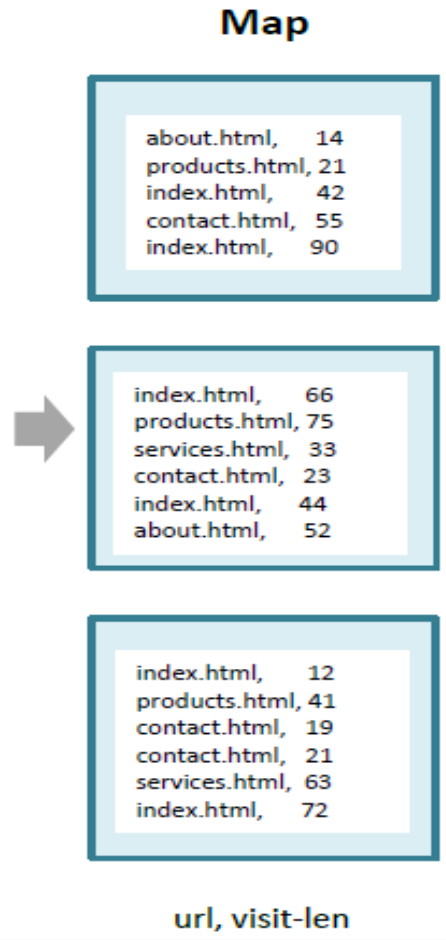
MapReduce Examples

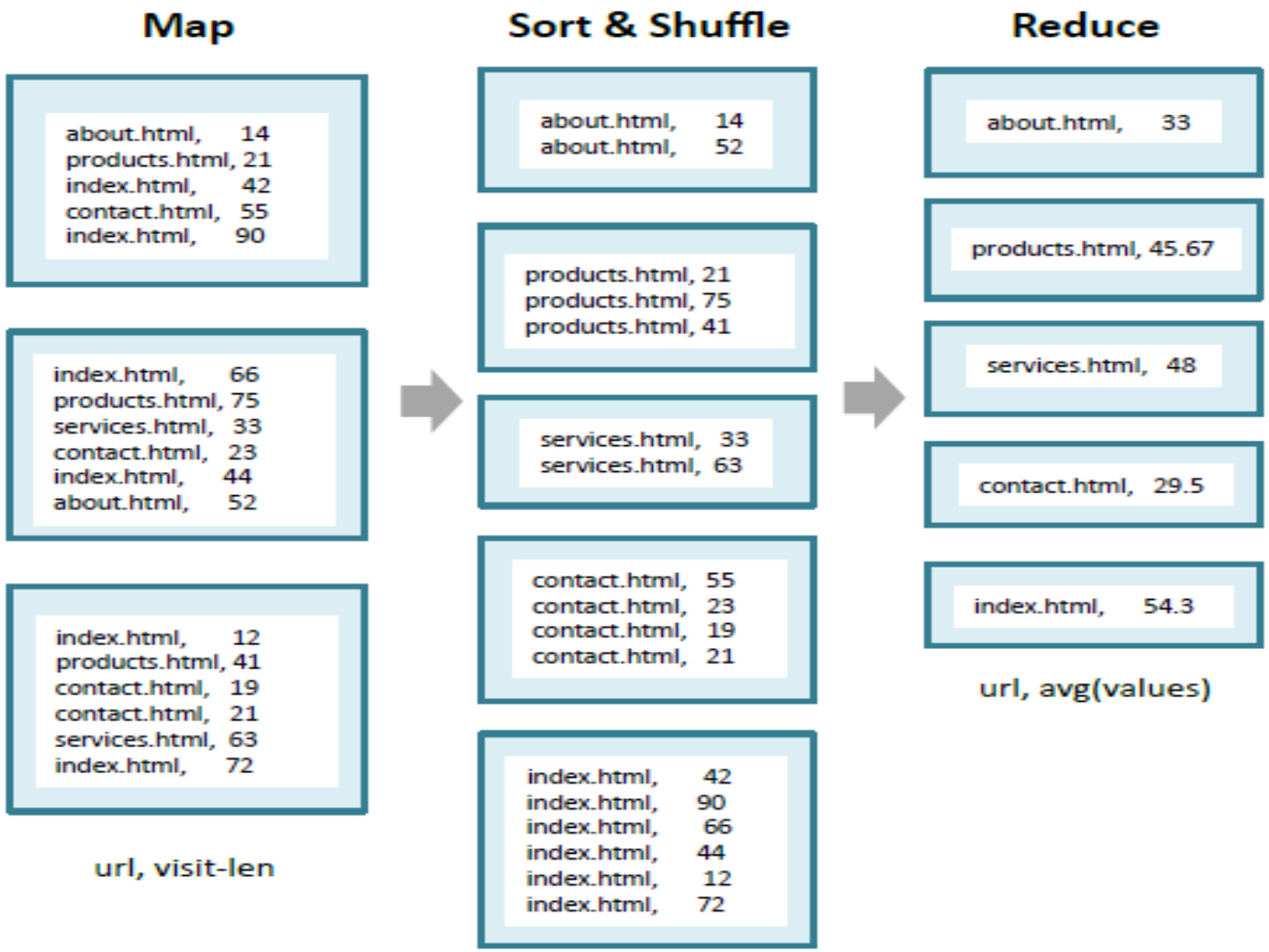
2. **Average:** Find the average time spent on each page in the given website:

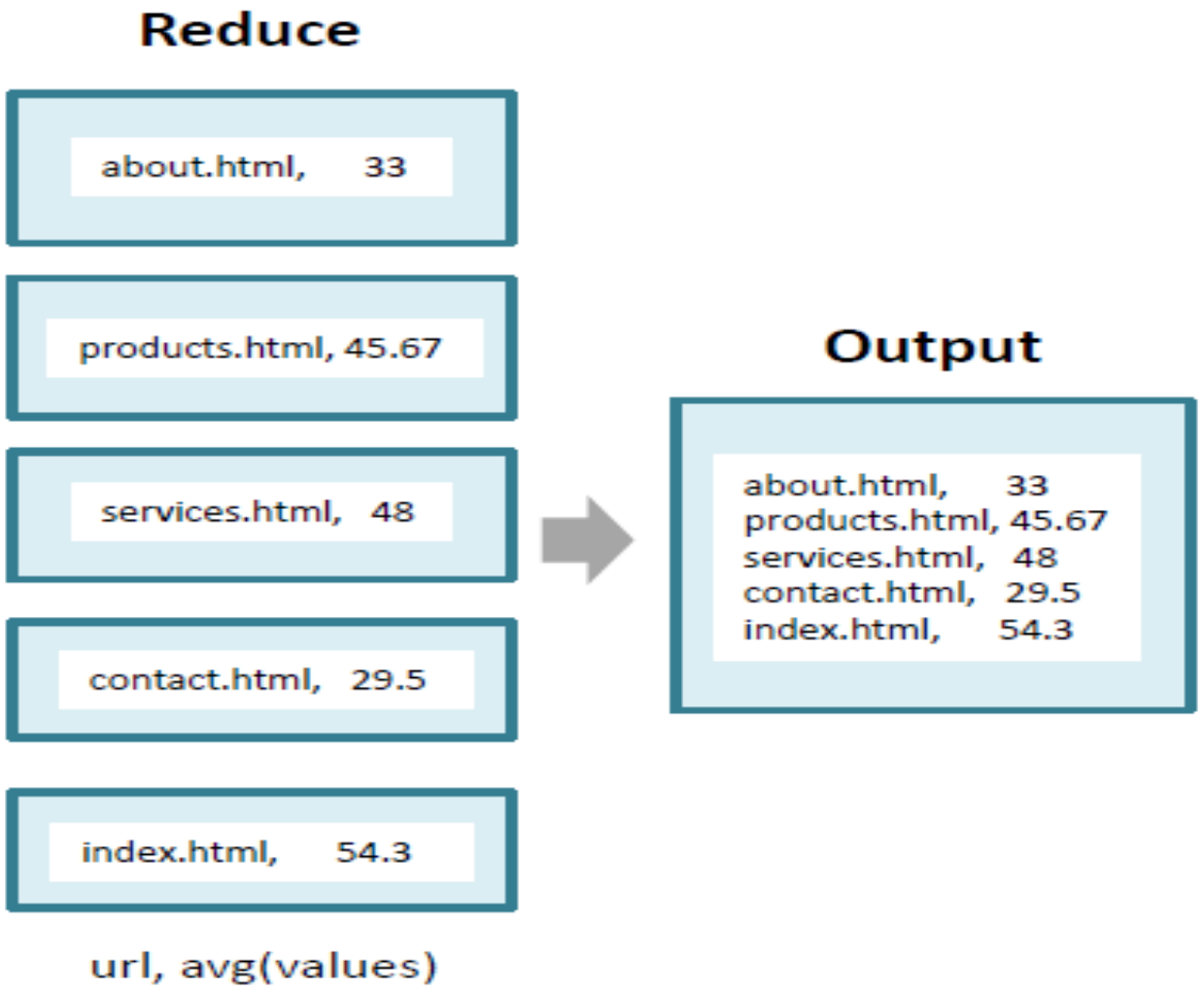
Part of Input to show its format

```
2014-04-01 13:45:42 http://example.com/products.html 77.140.91.33    89
2014-10-01 14:39:48 http://example.com/index.html    113.107.99.122  13
2014-06-23 21:27:50 http://example.com/about.html    50.98.73.129    73
2014-01-15 21:27:09 http://example.com/services.html 149.59.51.52    59
2014-05-13 11:43:42 http://example.com/about.html    61.91.88.85     46
2014-02-17 03:17:37 http://example.com/contact.html  68.78.59.117    98
```

(Date, Time, URL, IP, Visit-Length)




Map
about.html, 14
products.html, 21
index.html, 42
contact.html, 55
index.html, 90
index.html, 66
products.html, 75
services.html, 33
contact.html, 23
index.html, 44
about.html, 52
index.html, 12
products.html, 41
contact.html, 19
contact.html, 21
services.html, 63
index.html, 72
url, visit-len
Sort & Shuffle
about.html, 14
about.html, 52
products.html, 21
products.html, 75
products.html, 41
services.html, 33
services.html, 63
contact.html, 55
contact.html, 23
contact.html, 19
contact.html, 21
index.html, 42
index.html, 90
index.html, 66
index.html, 44
index.html, 12
index.html, 72
Reduce
about.html, 33
products.html, 45.67
services.html, 48
contact.html, 29.5
index.html, 54.3
url, avg(values)


Reduce
about.html, 33
products.html, 45.67
services.html, 48
contact.html, 29.5
index.html, 54.3
url, avg(values)
Output
about.html, 33
products.html, 45.67
services.html, 48
contact.html, 29.5
index.html, 54.3

MapReduce Examples

2. **Average computation Explanation:**
   - To compute the average, the mapper function emits key-value pairs where the key is the field to group-by and value contains related items required to compute the average.
   - The mapper function in this example parses each line of the input and emits key-value pairs where the key is the URL and value is the visit length.
   - The reducer receives the list of values grouped by the key (which is the URL) and finds the average of these values.

MapReduce Examples

3. **Top-N:** Find the top 3 most visited pages in the given website:

Part of Input to show its format

```
2014-04-01 13:45:42 http://example.com/products.html 77.140.91.33     89
2014-10-01 14:39:48 http://example.com/index.html    113.107.99.122   13
2014-06-23 21:27:50 http://example.com/about.html    50.98.73.129     73
2014-01-15 21:27:09 http://example.com/services.html 149.59.51.52     59
2014-05-13 11:43:42 http://example.com/about.html    61.91.88.85      46
2014-02-17 03:17:37 http://example.com/contact.html  68.78.59.117     98
```

(Date, Time, URL, IP, Visit-Length)

Map

```
about.html,    1
products.html, 1
index.html,    1
contact.html,  1
index.html,    1
```

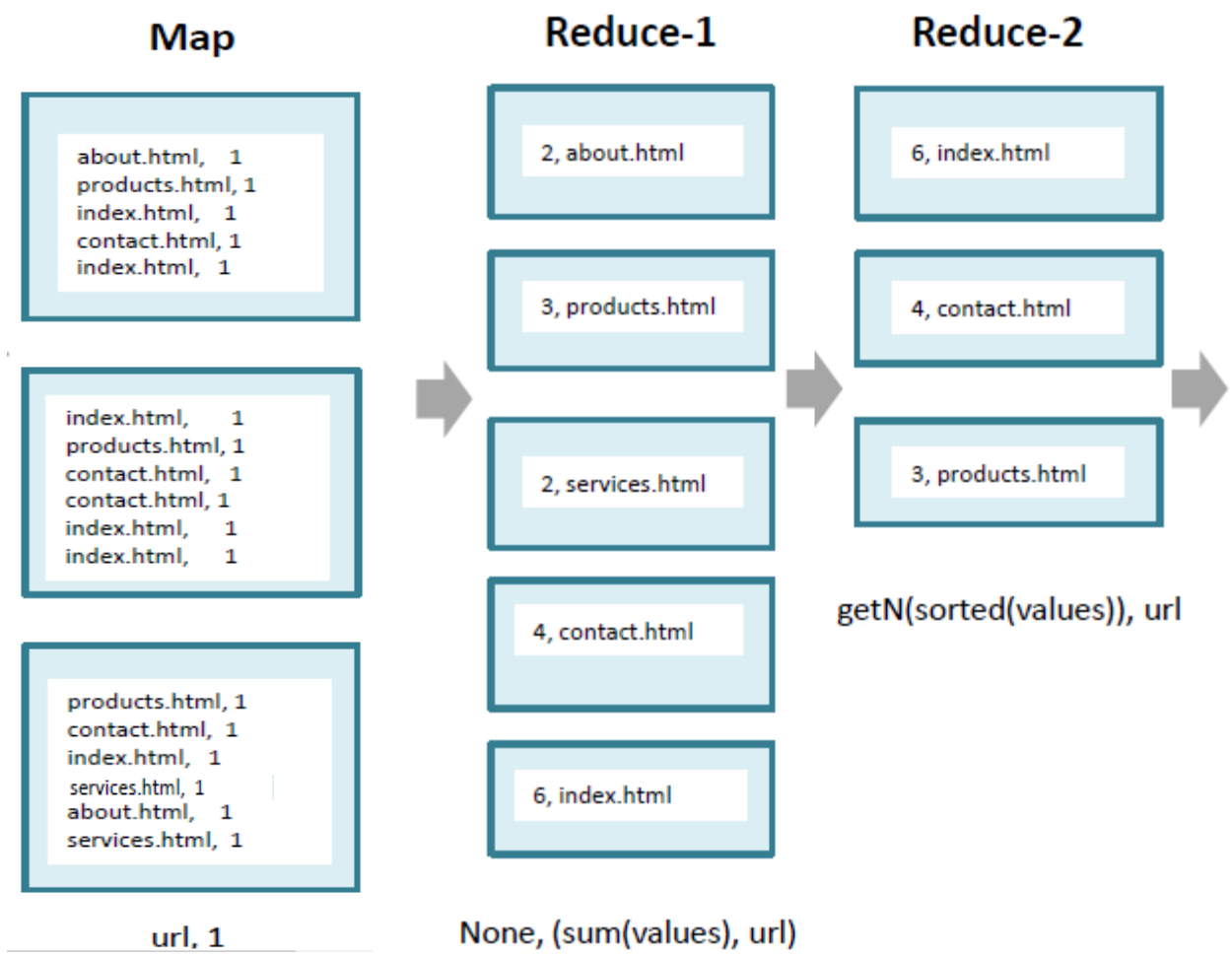
```
index.html,    1
products.html, 1
contact.html,  1
contact.html,  1
index.html,    1
index.html,    1
```

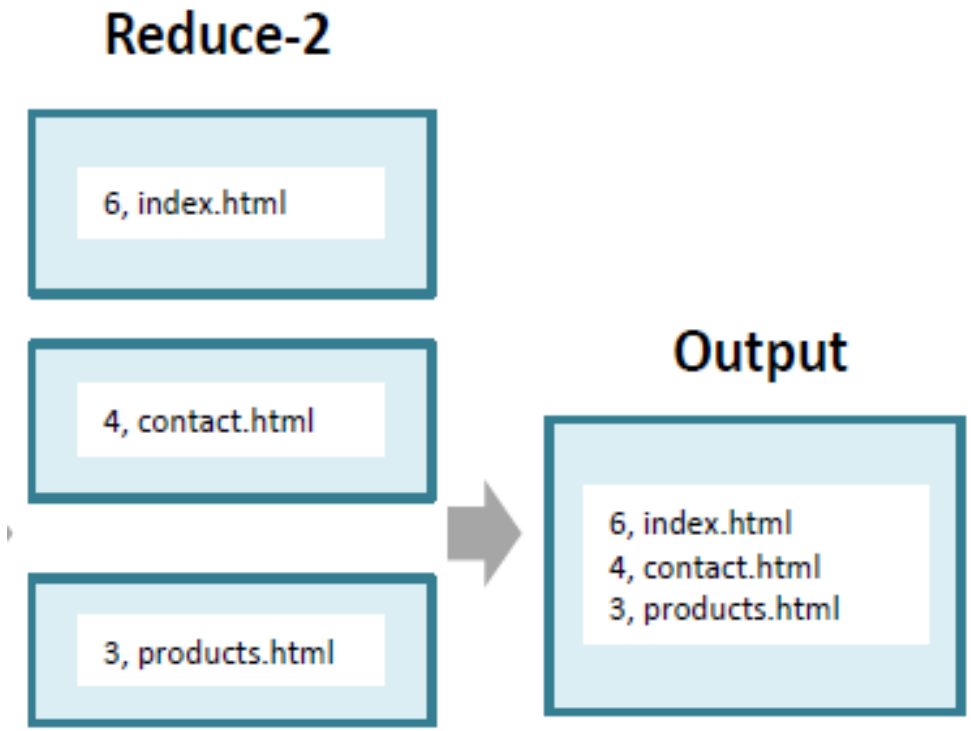
```
products.html, 1
contact.html,  1
index.html,    1
services.html, 1
about.html,    1
services.html, 1
```

url, 1

a. What will be the output of the shuffle and sort stage?

b. In Reduce-2, how many reducers do we have?





getN(sorted(values)), url

MapReduce Examples

3. **Top-N computation Explanation:**
   - The mapper function in this example parses each line of the input and emits key-value pairs where the key is the URL and value is '1'.
   - The reducer receives the list of values grouped by the key and sums up the values to count the visits for each page.
   - The first reducer emits None as the key and a tuple comprising of page visit count and page URL and the value.
   - The second reducer receives a list of (visit count, URL) pairs all grouped together (as the key is None). The reducer sorts the visit counts and emits top 3 visit counts along with the page URLs.
   - In this example, a two-step job was required because we need to compute the page visit counts first before finding the top 3 visited pages.

MapReduce Examples

4. **Filtering:**
   - Filter out a subset of the records based on a filtering criteria.
   - For example: filtering all page visits for the page ‘contact.html’ in the month of Dec 2014.

MapReduce Examples

Part of Input to show its format

```
2014-04-01 13:45:42 http://example.com/products.html 77.140.91.33    89
2014-10-01 14:39:48 http://example.com/index.html    113.107.99.122  13
2014-06-23 21:27:50 http://example.com/about.html    50.98.73.129    73
2014-01-15 21:27:09 http://example.com/services.html 149.59.51.52    59
2014-05-13 11:43:42 http://example.com/about.html    61.91.88.85     46
2014-02-17 03:17:37 http://example.com/contact.html  68.78.59.117    98
```

(Date, Time, URL, IP, Visit-Length)

Map

http://example.com/contact.html, (2014-12-14, 16:47:01, 108.147.78.88, 96)
http://example.com/contact.html, (2014-12-20, 21:00:49, 71.71.39.144, 21)
http://example.com/contact.html, (2014-12-15, 13:13:21, 144.84.67.149, 97)
http://example.com/contact.html, (2014-12-00, 10:24:57, 85.82.69.136, 80)

http://example.com/contact.html, (2014-12-26, 11:49:49, 124.131.37.81, 75)
http://example.com/contact.html, (2014-12-09, 13:35:34, 112.50.35.133, 96)
http://example.com/contact.html, (2014-12-29, 14:23:12, 89.107.69.46, 51)

URL, (Date, Time, IP, Visit-Length)

Output

http://example.com/contact.html, (2014-12-14, 16:47:01, 108.147.78.88, 96)
http://example.com/contact.html, (2014-12-20, 21:00:49, 71.71.39.144, 21)
http://example.com/contact.html, (2014-12-15, 13:13:21, 144.84.67.149, 97)
http://example.com/contact.html, (2014-12-00, 10:24:57, 85.82.69.136, 80)

http://example.com/contact.html, (2014-12-26, 11:49:49, 124.131.37.81, 75)
http://example.com/contact.html, (2014-12-09, 13:35:34, 112.50.35.133, 96)
http://example.com/contact.html, (2014-12-29, 14:23:12, 89.107.69.46, 51)

MapReduce Examples

4. **Filtering computation Explanation:**
   - Filtering is useful when you want to get a subset of the data for further processing.
   - Filtering requires only a Map task.
   - Each mapper filters out its local records based on the filtering criteria in the map function.
   - The mapper function in this example parses each line of the input, extracts the month, year and page URL and emits key-value pairs if the month and year are Dec 2014 and the page URL is 'http://example.com/contact.html'.
   - The key is the URL, and the value is a tuple containing the rest of the parsed fields.

MapReduce Examples

- **Inverted index:**
  - An inverted index is an index data structure which stores the mapping from the content (such as words in a document or on a webpage) to the location of the content (such as document filename or a page URL).
  - Search engines use inverted indexes to enable faster searching of documents or pages containing some specific content.

MapReduce Examples

- **Inverted index:**
  - Let us assume we have all the content from all the books combined into one large file with two fields separated by a pipe symbol ('|').
  - The first field is the filename and the second field contains all the content in the file.

Part of Input to show its format

```
milton.txt | Of Man's first disobedience , and the fruit Of that forbidden...
austen.txt | Emma Woodhouse , handsome , clever , and rich , with a...
edgeworth.txt | Near the ruins of the castle of Rossmore, in Ireland...
chesterton.txt | The flying ship of Professor Lucifer sang through the skies like...
shakespeare.txt | Actus Primus . Scoena Prima . Enter Flauius , Murellus , and...
```

Filename | File Contents

MapReduce Examples

- **Inverted index:**
  - The mapper function in this example parses each line of the input and emits key-value pairs where the **key** is *each word* in the line and **value** is the *filename*.

Part of Input to show its format

milton.txt | Of Man's first disobedience , and the fruit Of that forbidden...
austen.txt | Emma Woodhouse , handsome , clever , and rich , with a...
edgeworth.txt | Near the ruins of the castle of Rossmore, in Ireland...
chesterton.txt | The flying ship of Professor Lucifer sang through the skies like...
shakespeare.txt | Actus Primus . Scoena Prima . Enter Flauius , Murellus , and...

Filename | File Contents

Map

years, whitman.txt
woman, whitman.txt
yet, whitman.txt
clever, austen.txt
work, austen.txt

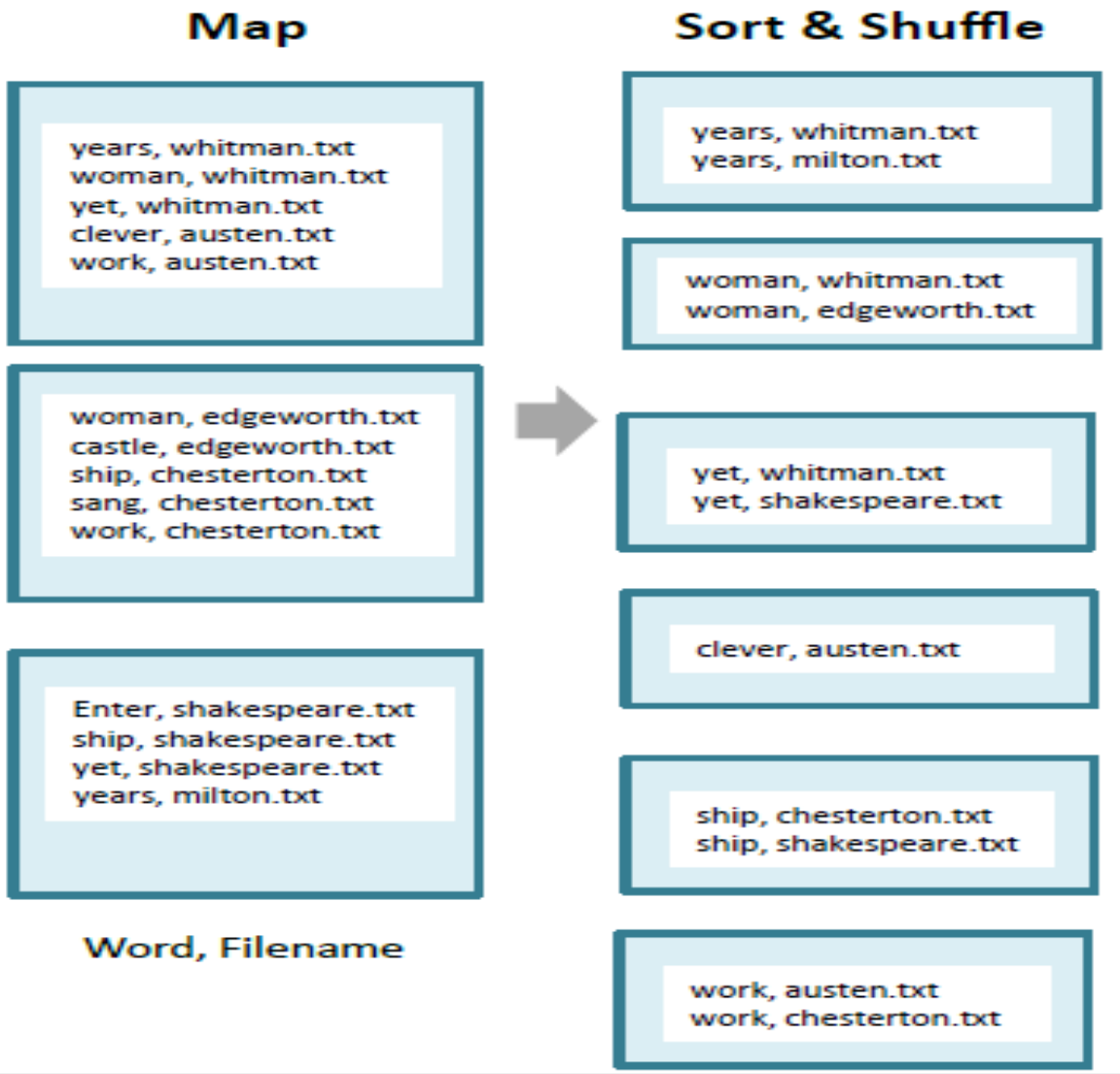
woman, edgeworth.txt
castle, edgeworth.txt
ship, chesterton.txt
sang, chesterton.txt
work, chesterton.txt

Enter, shakespeare.txt
ship, shakespeare.txt
yet, shakespeare.txt
years, milton.txt

Word, Filename

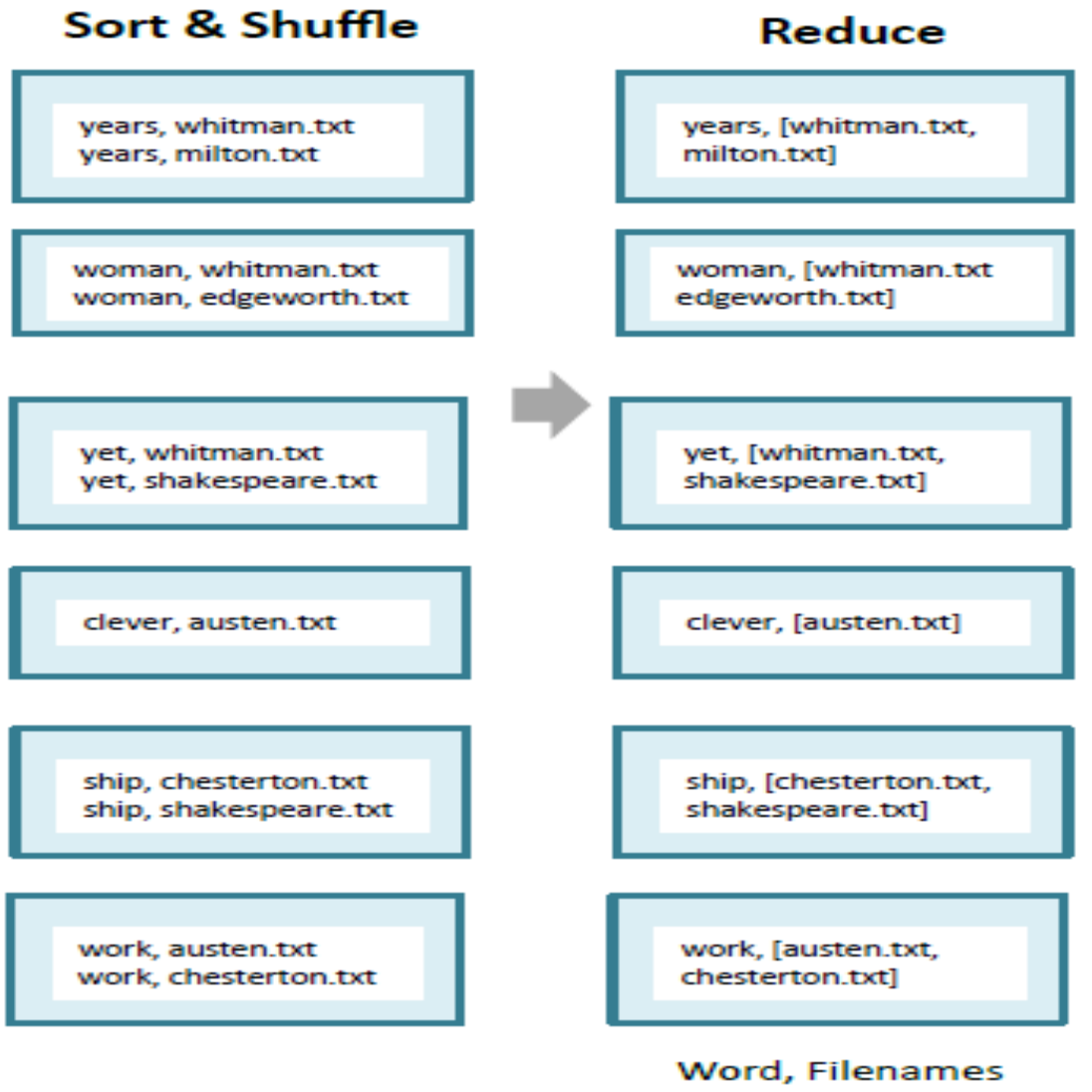
MapReduce Examples

- **Inverted index:**
  - The sort & shuffle groups filenames with same key.



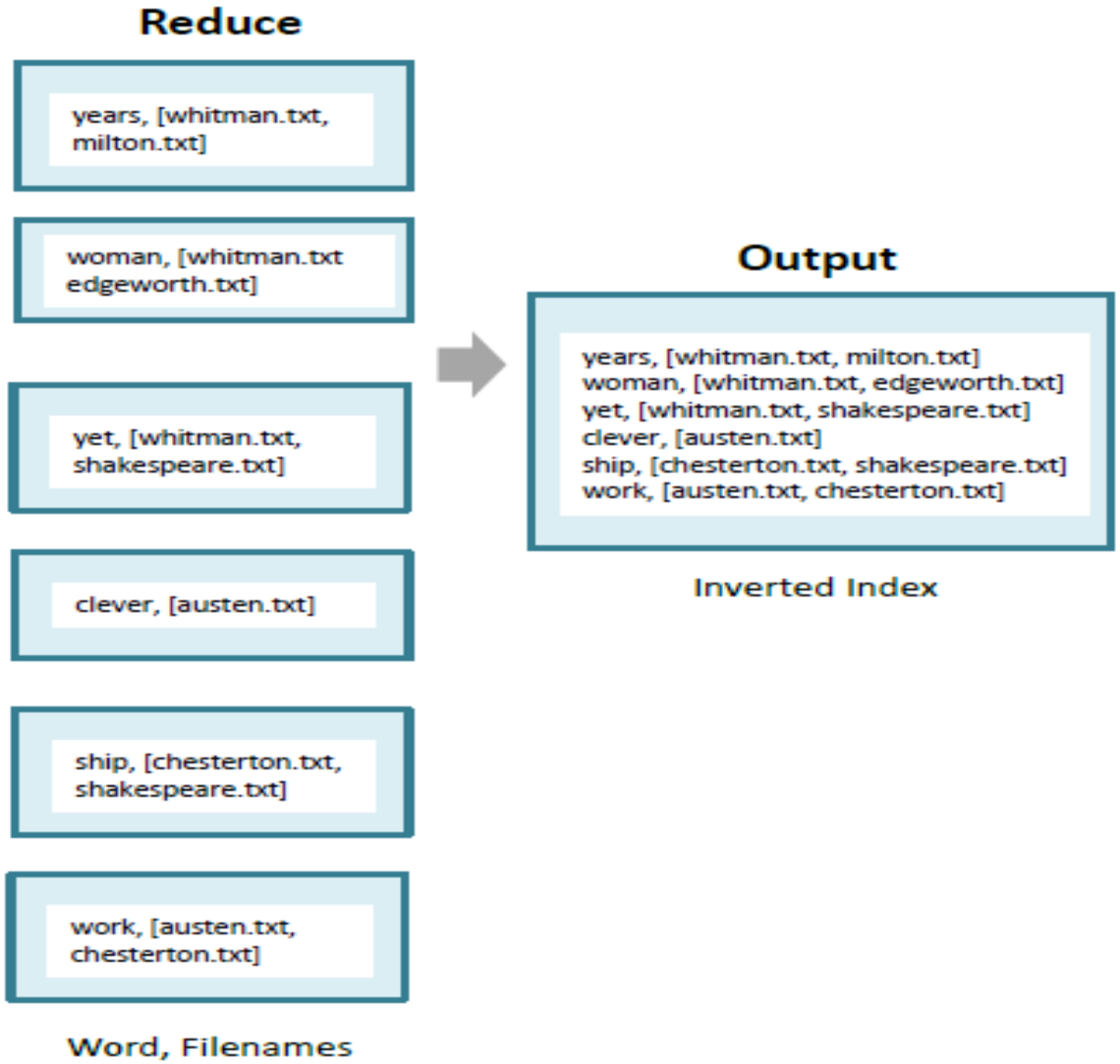
MapReduce Examples

- **Inverted index:**
  - The reducer receives the list of values (filenames) grouped by the key (word) and emits the word and the list of filenames in which the word occurs.



MapReduce Examples

- **Inverted index:**
  - The final output is the inverted index.

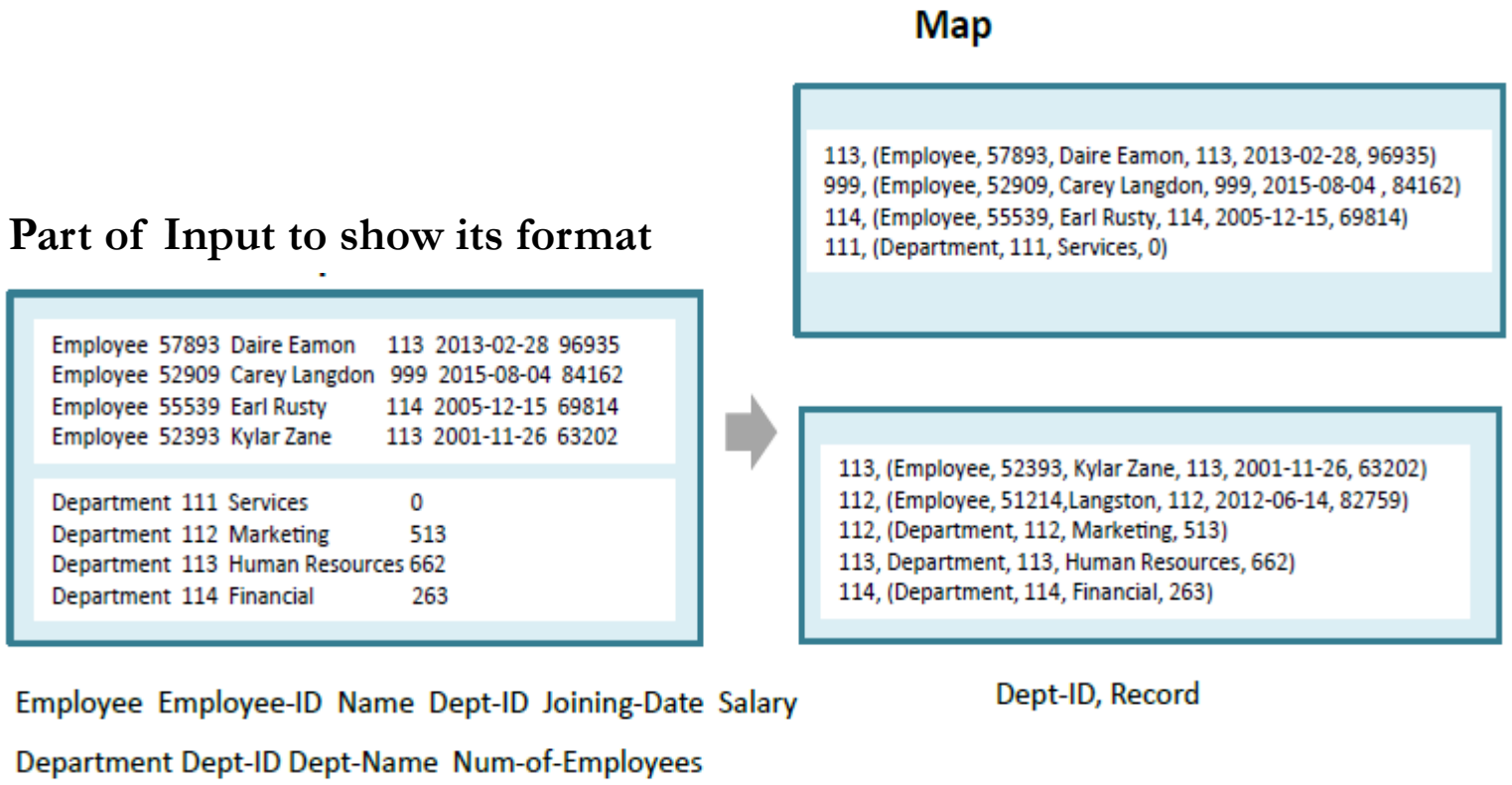


MapReduce Examples

- **Join:**
  - When datasets contain multiple files, joins are used to combine the records in the files for further processing.
  - Joins combine two or more datasets or records in multiple files, based on a field (called the join attribute or foreign key).
  - Let's take the example of joining two tables A and B.
  - We will use two datasets containing records of employees and departments in an organization and will join by the Department ID field.

MapReduce Examples

- **Join:**
  - The mapper function in this example parses each line of the input and emits key-value pairs where the **key** is the *Department ID* and **value** is the *complete record.*



MapReduce Examples

- **Join:**
  - The reducer receives the list of values all grouped by the Department ID.
  - In the reducer, we check the first field of each value and if the field is 'Employee', we add the value to an employees list and if the first field is 'Department', we add the value to the departments list.
  - Next, we iterate over both lists and perform the join.

Sort & Shuffle

113, (Employee, 57893, Daire Eamon, 113, 2013-02-28, 96935)
113, (Employee, 52393, Kylar Zane, 113, 2001-11-26, 63202)
113, Department, 113, Human Resources, 662)

999, (Employee, 52909, Carey Langdon, 999, 2015-08-04 , 84162)

114, (Employee, 55539, Earl Rusty, 114, 2005-12-15, 69814)
114, (Department, 114, Financial, 263)

111, (Department, 111, Services, 0)

112, (Employee, 51214,Langston, 112, 2012-06-14, 82759)
112, (Department, 112, Marketing, 513)

Reduce

Employee, 57893, Daire Eamon, 113, 2013-02-28, 96935, Department, 113, Human Resources, 662
Employee, 52393, Kylar Zane, 113, 2001-11-26, 63202, Department, 113, Human Resources, 662

Employee, 55539, Earl Rusty, 114, 2005-12-15, 69814, Department, 114, Financial, 263

Employee, 51214,Langston, 112, 2012-06-14, 82759, Department, 112, Marketing, 513

Output

Employee, 57893, Daire Eamon, 113, 2013-02-28, 96935, Department, 113, Human Resources, 662
Employee, 52393, Kylar Zane, 113, 2001-11-26, 63202, Department, 113, Human Resources, 662
Employee, 55539, Earl Rusty, 114, 2005-12-15, 69814, Department, 114, Financial, 263
Employee, 51214,Langston, 112, 2012-06-14, 82759, Department, 112, Marketing, 513

Employee, Employee-ID , Name, Dept-ID, Joining-Date, Salary, Department, Dept-ID, Dept-Name, Num-of-Employees

Thank You