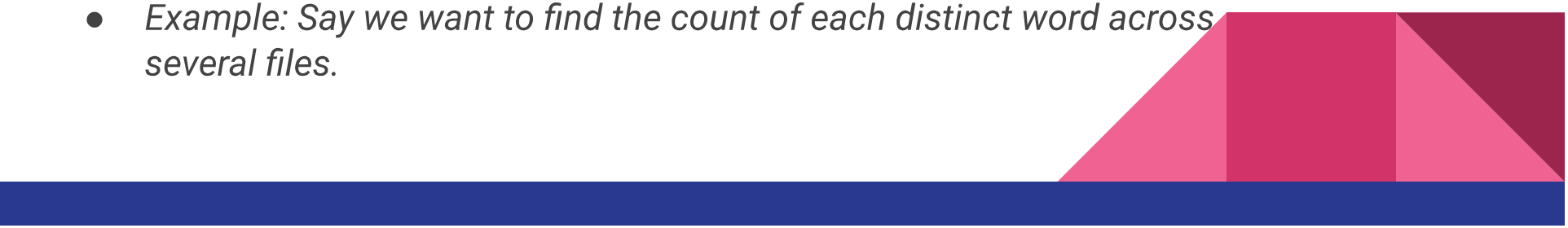
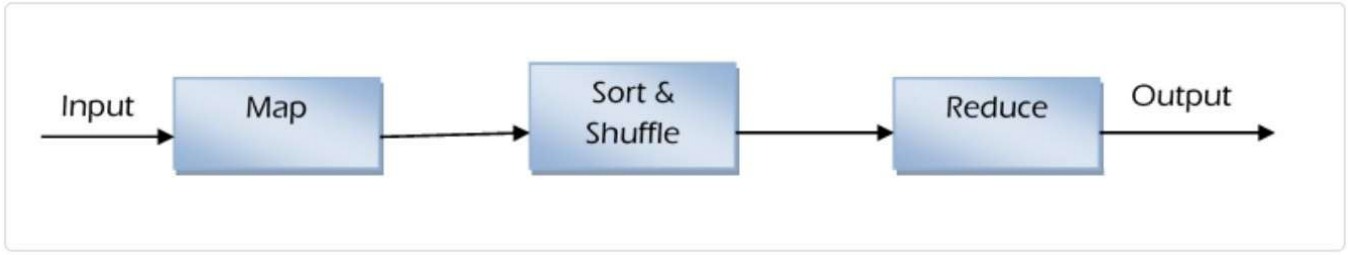
Map Reduce Algorithm

Alec Mazzoli

The Basics

- You have a huge amount of data
- You could process it one at a time/sequentially, but that would take ages
- Map Reduce divides data into smaller sub-tasks, which can be executed at the same time
- So, Map Reduce models a common situation in parallel processing
- Three steps:
  - Map
  - Sort and shuffle
  - Reduce
- *Example: Say we want to find the count of each distinct word across several files.*

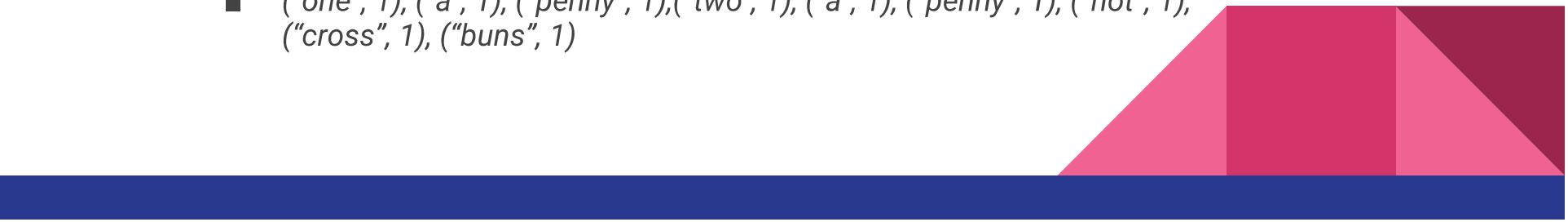


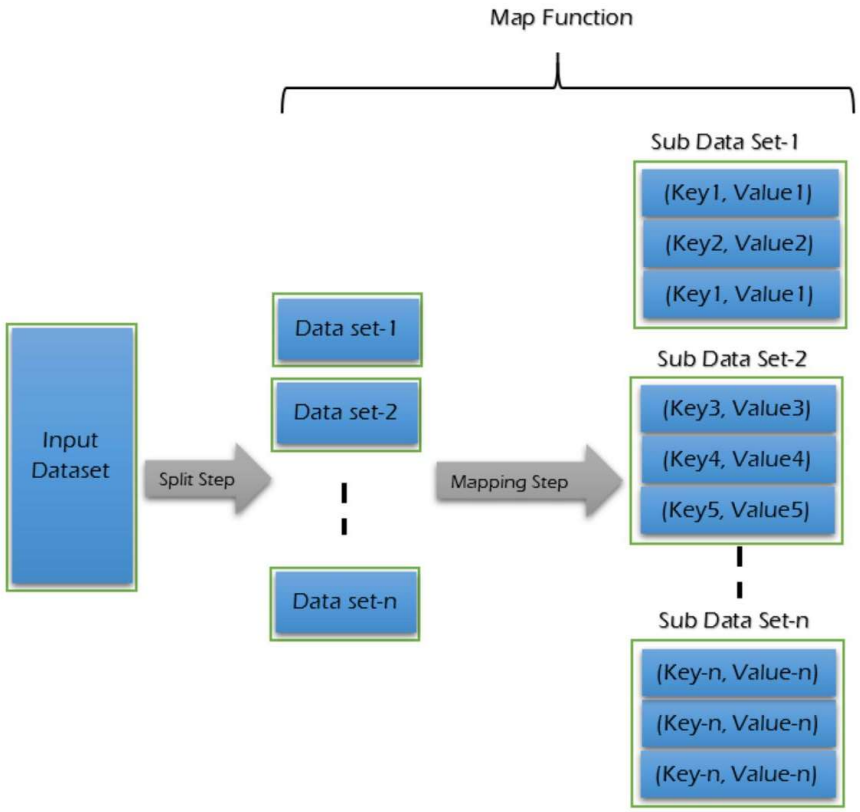


Phase	Input	Output
Mapper	(K,V)	(K,V)
Shuffle & Sort	(K,V)	(K, list(V))
Reducer	(K,list(V))	(K,V)

The Map Step

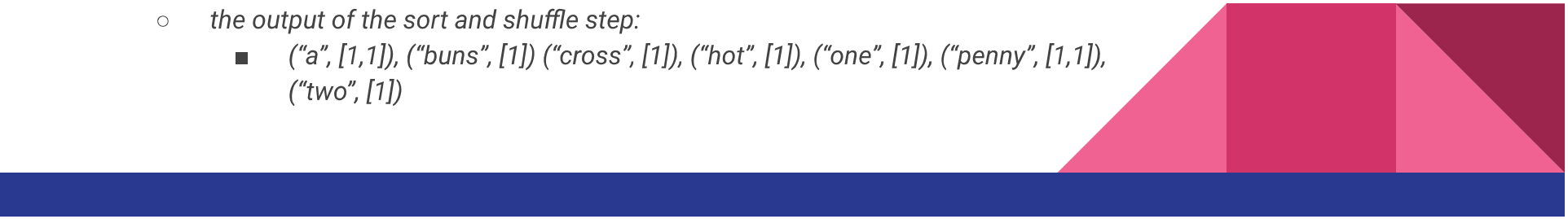
- Executed in parallel on multiple machines
- map() is invoked on each “input record” - usually defaults as one line of one file, but you can specify your own function to break input files into records
- Output is a set of (key, value) pairs
- *Example*
  - *In our word count function, map() breaks up each input record (line) into individual words, and for each word outputs a (key, value) pair - (“word”, 1)*
  - *Input to the map() step:*
    - *“One a penny, two a penny, hot cross buns.”*
  - *Output from the map() step:*
    - *(“one”, 1), (“a”, 1), (“penny”, 1),(“two”, 1), (“a”, 1), (“penny”, 1), (“hot”, 1), (“cross”, 1), (“buns”, 1)*

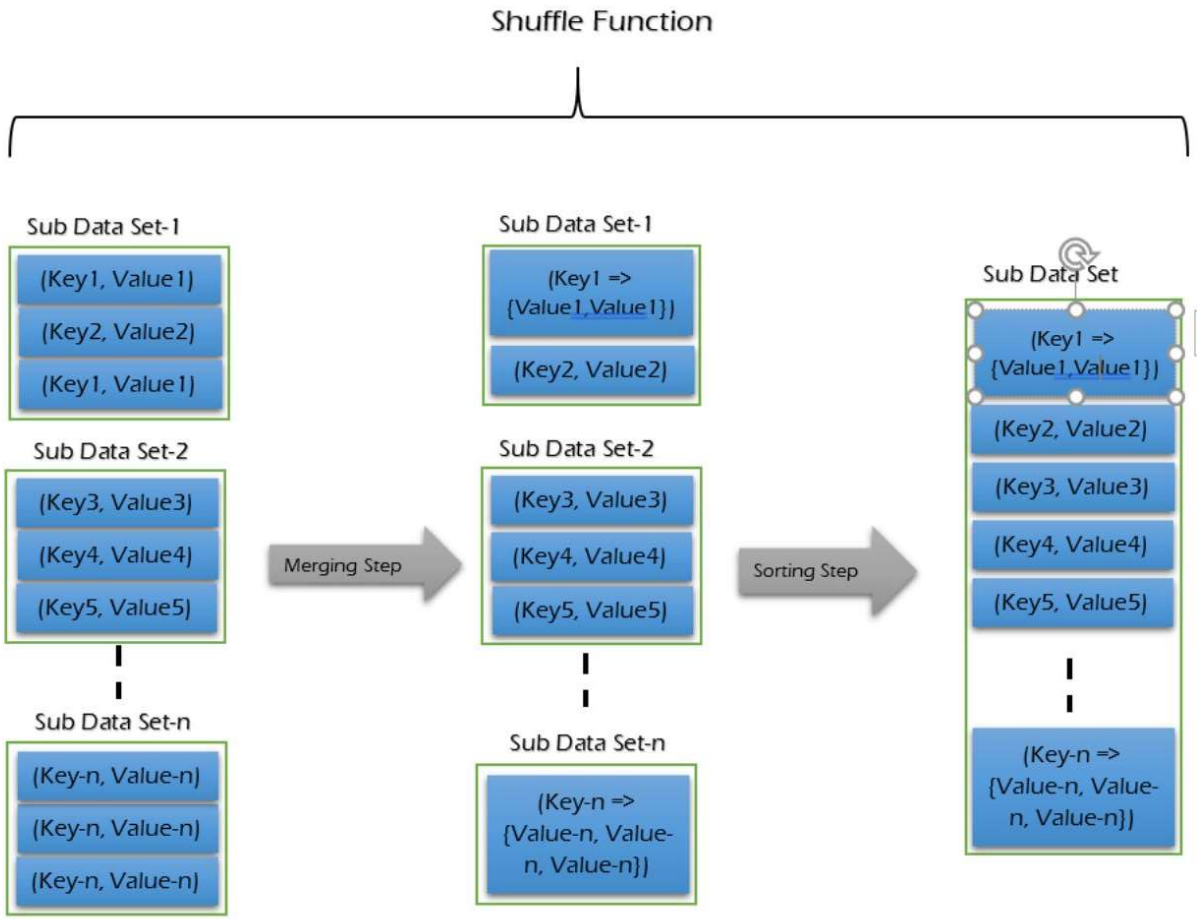



Map Function
Input Dataset
Split Step
Data set-1
Data set-2
Data set-n
Mapping Step
Sub Data Set-1
(Key1, Value1)
(Key2, Value2)
(Key1, Value1)
Sub Data Set-2
(Key3, Value3)
(Key4, Value4)
(Key5, Value5)
Sub Data Set-n
(Key-n, Value-n)
(Key-n, Value-n)
(Key-n, Value-n)

The Sort and Shuffle Step

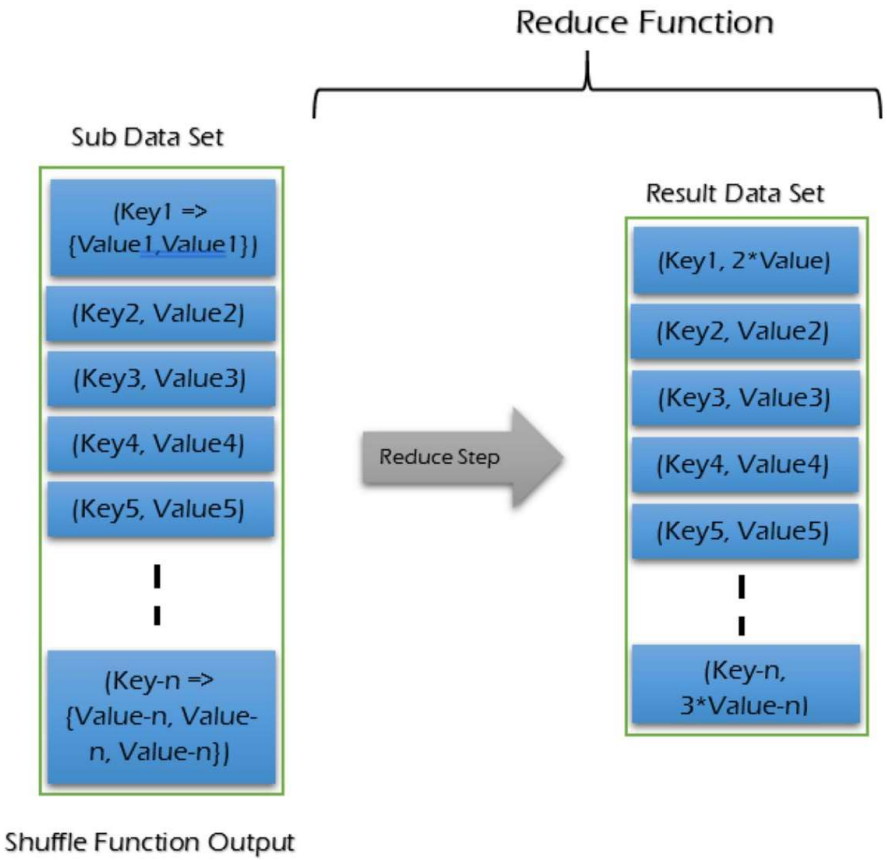
- Performs data exchange between machines
- Sorts the (key, value) pairs to bring all the values for a key together
- 2 sub-steps:
  - Merging
    - Combines all (key, value) pairs that have the same keys and returns (key, list<value>) with all the distinct values associated with that key in a list
  - Sorting
    - Sorts all the (key, value) pairs by key
- *Example*
  - *the output of the map() function/input to the sort and shuffle step:*
    - *(“one”, 1), (“a”, 1), (“penny”, 1),(“two”, 1), (“a”, 1), (“penny”, 1), (“hot”, 1), (“cross”, 1), (“buns”, 1)*
  - *the output of the sort and shuffle step:*
    - *(“a”, [1,1]), (“buns”, [1]) (“cross”, [1]), (“hot”, [1]), (“one”, [1]), (“penny”, [1,1]), (“two”, [1])*




Shuffle Function
Sub Data Set-1
(Key1, Value1)
(Key2, Value2)
(Key1, Value1)
Sub Data Set-2
(Key3, Value3)
(Key4, Value4)
(Key5, Value5)
Sub Data Set-n
(Key-n, Value-n)
(Key-n, Value-n)
(Key-n, Value-n)
Merging Step
Sub Data Set-1
(Key1 => {Value1,Value1})
(Key2, Value2)
Sub Data Set-2
(Key3, Value3)
(Key4, Value4)
(Key5, Value5)
Sub Data Set-n
(Key-n => {Value-n, Value-n, Value-n})
Sorting Step
Sub Data Set
(Key1 => {Value1,Value1})
(Key2, Value2)
(Key3, Value3)
(Key4, Value4)
(Key5, Value5)
(Key-n => {Value-n, Value-n, Value-n})

The Reduce Step

- Takes set of (key, list<value>) pairs from shuffle and sort step
- For each (key, list<value>) pair, combines all the values in the list to get one (key, value)
  - Ex. the reduce step adds all the values in the list, and returns the sum as one value
- Returns sorted set of (key, value) pairs
- *Example*
  - *The output of the sort and shuffle step/input to the reduce step:*
    - *(“a”, [1,1]), (“buns”, [1]) (“cross”, [1]), (“hot”, [1]), (“one”, [1]), (“penny”, [1,1]), (“two”, [1])*
  - *The output of the reduce step:*
    - *(“one”, 1), (“a”, 2), (“penny”, 2), (“two”, 1), (“hot”, 1), (“cross”, 1), (“buns”, 1)*


Reduce Function
Sub Data Set
(Key1 => {Value1,Value1})
(Key2, Value2)
(Key3, Value3)
(Key4, Value4)
(Key5, Value5)
(Key-n => {Value-n, Value-n, Value-n})
Shuffle Function Output
Reduce Step
Result Data Set
(Key1, 2*Value)
(Key2, Value2)
(Key3, Value3)
(Key4, Value4)
(Key5, Value5)
(Key-n, 3*Value-n)

Example #2 - Log Processing

```
...
2013/02/21 10:31:22.00EST /slide-dir/11.ppt
2013/02/21 10:43:12.00EST /slide-dir/12.ppt
2013/02/22 18:26:45.00EST /slide-dir/13.ppt
2013/02/22 18:26:48.00EST /exer-dir/2.pdf
2013/02/22 18:26:54.00EST /exer-dir/3.pdf
2013/02/22 20:53:29.00EST /slide-dir/12.ppt
...
```

- We have a file showing accesses to a website
  - Want to know how many times each of the files in the slide-dir directory was accessed between 2013/01/01 and 2013/01/31
  - Each line is an input record
- map() - break line into fields. If date in range, output (“filename”, 1)
- Sort and shuffle - combine all filenames - (key, list<value>)
- reduce() - add up all accesses for each filename

Sources

- The textbook - 10.3
- https://www.tutorialscampus.com/map-reduce/algorithm.htm