CSC 369: Distributed Computing

Alex Dekhtyar

May 1

Day 12: MapReduce

Housekeeping

Quiz:

Stat	Individual	Team	Lift
Mean	16.94		
Median	17		
Standard Deviation	4.73		
Max	27		
Min	10		

Housekeeping

Quiz:

Stat	Individual	Team	Lift
Mean	16.94	21.85	4.91
Median	17	21.5	4.25
Standard Deviation	4.73	4.37	4.97
Max	27	29	14
Min	10	13	-7

Housekeeping

Lab 4:

Test Cases are now correct

Remote MongoDB connection

“server”: “ambari-head.csc.calpoly.edu”
Cal Poly VPN

Robot Password Changes

Housekeeping

Lab 4:

Test Cases are now correct

Remote MongoDB connection

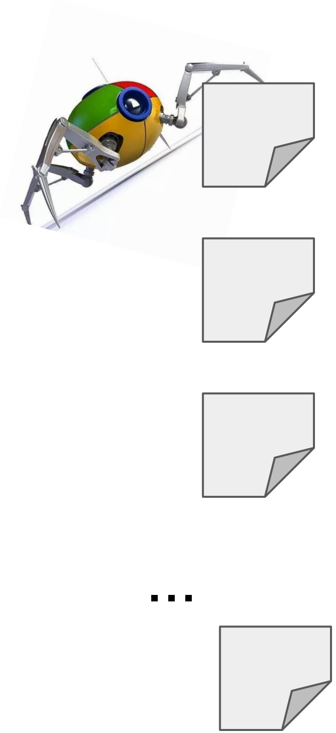
“server”: “ambari-head.csc.calpoly.edu”
Cal Poly VPN

Robot Password Changes

MapReduce

Motivation: The Google Example

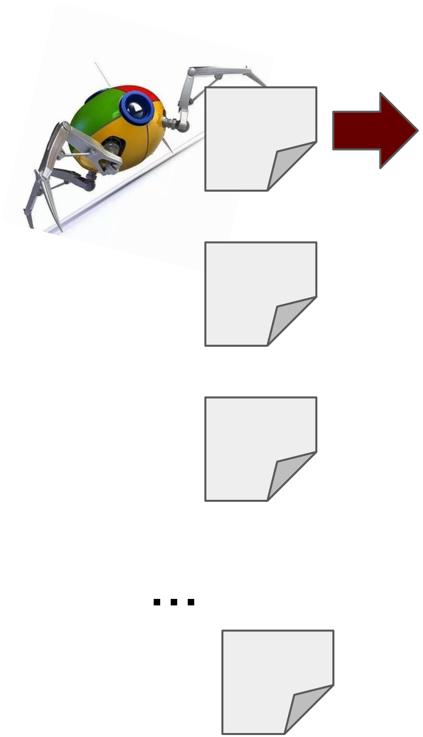
The World Wide Web:



...

Motivation: The Google Example

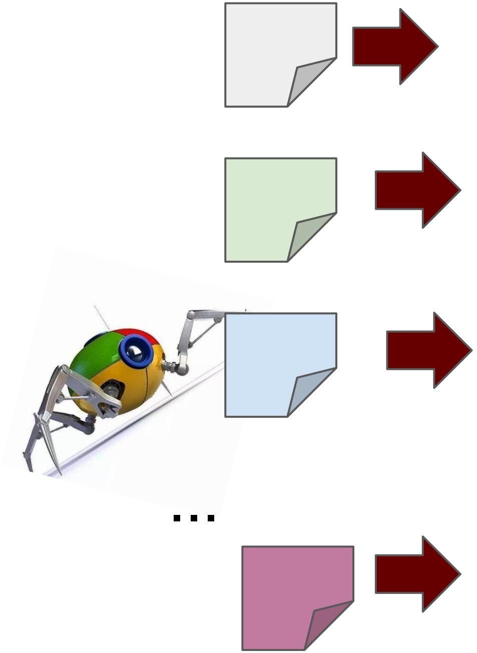
The World Wide Web:



{“Cal”, “Poly”, “San”, “Luis”, “Obispo”,.... }

Motivation: The Google Example

The World Wide Web



{“Cal”, “Poly”, “San”, “Luis”, “Obispo”, “university”.... }

{“Covid-19”, “San”, “Luis”, “Obispo”, “positive”...}

{“Covid-19”, “Newsom”, “beach”, “stay-at-home”...}

{“students”, “university”, “on-line”, “classes”, “sleep”

Motivation: The Google Example

The Inverted Index

“university”

“Covid-19”

“Luis”

“Obispo”

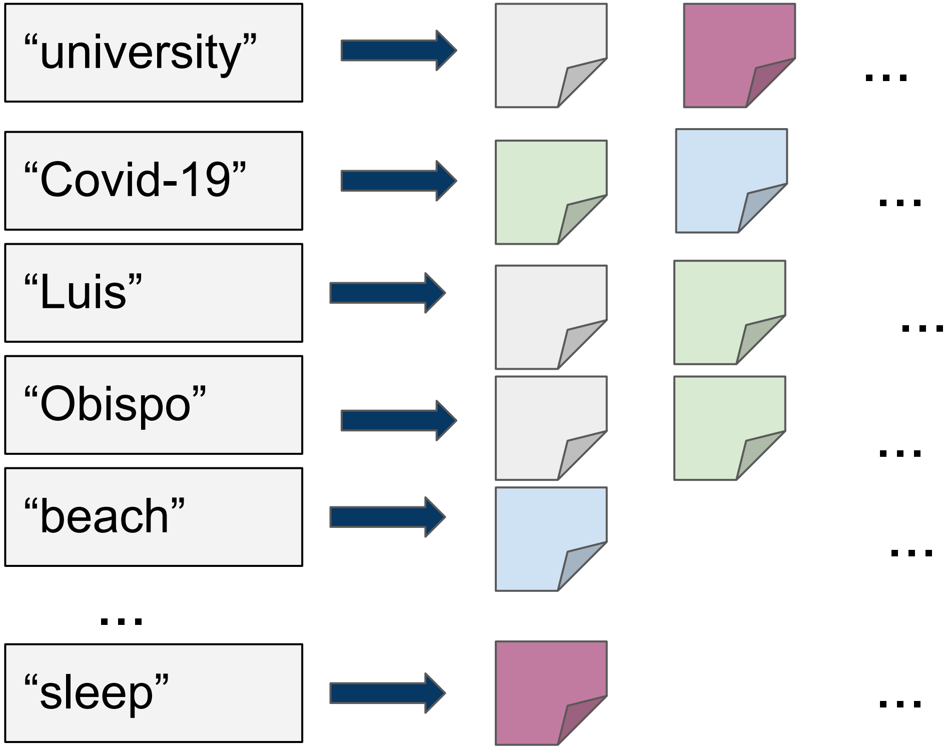
“beach”

...

“sleep”

Motivation: The Google Example

The Inverted Index



"university" ...

"Covid-19" ...

"Luis" ...

"Obispo" ...

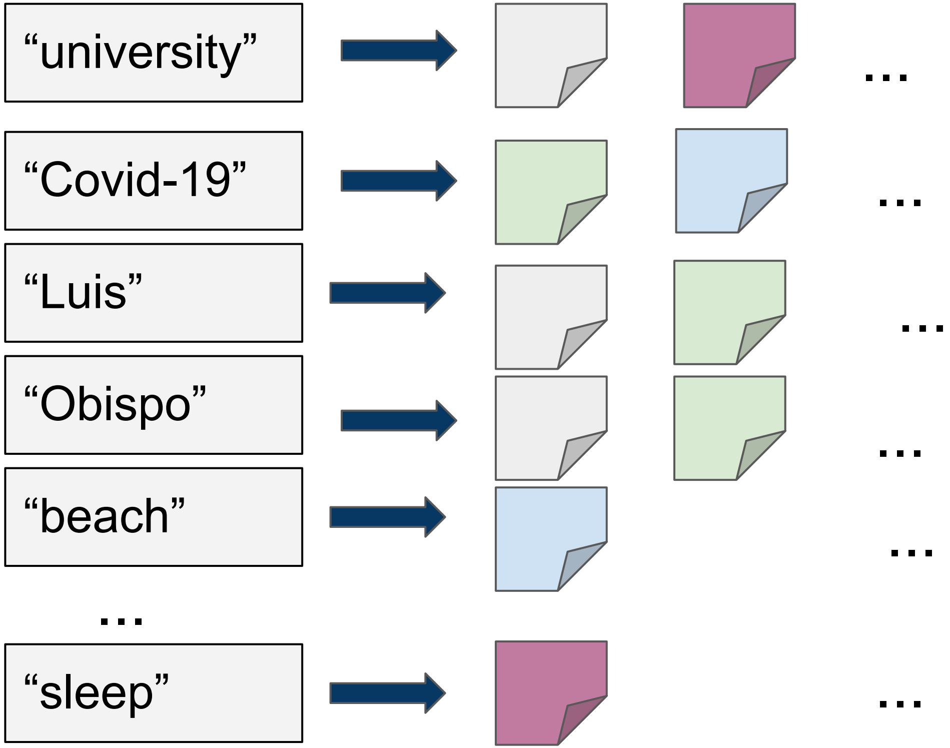
"beach" ...

...

"sleep" ...

Motivation: The Google Example

The Inverted Index



“university”

“Covid-19”

“Luis”

“Obispo”

“beach”

…

“sleep”

BUT HOW?

Distributed
(Petabyte scale index)

Fast

Simple to write

MapReduce

Jeffrey Dean, Sanjay Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*

Noticed that a lot of code of distributed computing kept doing same “types” of things.

Writing distributed code is hard

Proposed a level of abstraction

Data

<key,value> pairs

Data Processing

<key,value> pairs

All distributed computing reduced to three types of operations

Map: from <key, value> → <key1, value1>

Shuffle: collect keys

Reduce: from <key, [value1,value2,..,valueN] → <key1, value1>

Data Processing

<key,value> pairs

All distributed computing reduced to three types of operations

Map: from <key, value> → <key1, value1>

Shuffle: collect keys **(most always the same)**

Reduce: from <key, [value1,value2,..,valueN] → <key1, value1>

MapReduce

Write a Map() and Reduce() transformations of data

- Simple code

Build a distributed computing framework that does the rest

MapReduce: Inverted Index

```
Map(key, value):   //key=url, value= bag of words
  for word in value do
   emit(word, key)
  end for
```

```
Reduce(key, values)://key=word, values= [url1,...,urln]
   return(key, values)
```

More Formally: Map()

$$Map:\ K \times V \rightarrow K' \times V'$$

K, K' -- universes of keys

V, V' -- universes of values (can be compound)

Transformation

More Formally: Map()

$$Map: K \times V \rightarrow \{K' \times V'\}$$

K, K' -- universes of keys

V, V' -- universes of values (can be compound)

Transformation

More Formally: Map()

$$Map:\ K \times V \rightarrow \{K' \times V'\}$$

K, K' -- universes of keys

V, V' -- universes of values (can be compound)

emit() instead of return()

Transformation

More Formally: Map()

$$Map:\ K\ x\ V \rightarrow \{K'\ x\ V'\}$$

```
map(key, value):    //value - bag of words
    for word in value:
        emit(word,1)
    end for
```

More Formally: Reduce()

$$\mathit{Reduce}: \ K \times (V)^* \rightarrow (V)^*$$

$$\mathit{Reduce} \ \ K \times (V)^* \rightarrow K \times (V)^*$$

Aggregation

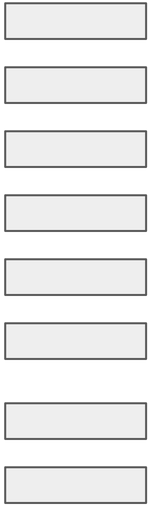
More Formally: Reduce()

$$\mathit{Map{:}}\ \ K \times (V)^* \rightarrow (V)^*$$

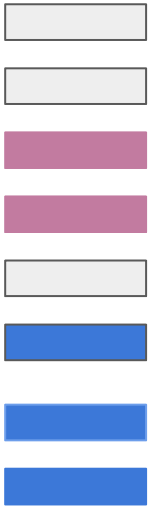
$$\mathit{Map{:}}\ \ K \times (V)^* \rightarrow K \times (V)^*$$

```
reduce(key, value):     //value - [1,1,1,...,1]
    count := 0
    for x in value:
        count := count+x
    end for
    emit(key, count)
```

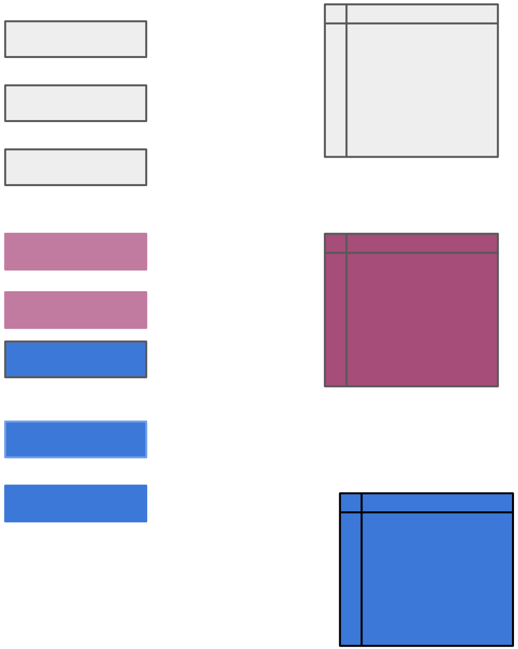
Map-Shuffle-Reduce



Map-Shuffle-Reduce

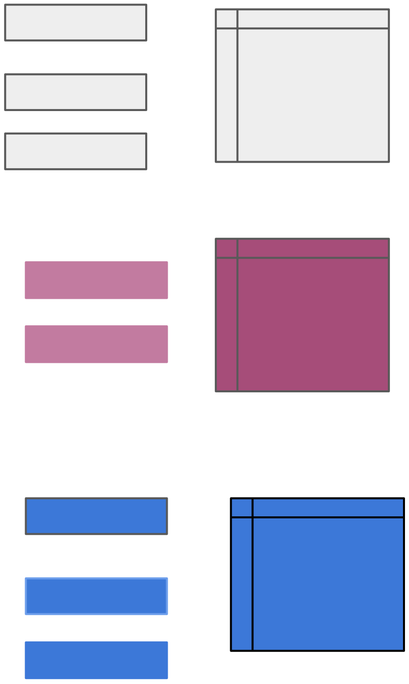


Map-Shuffle-Reduce



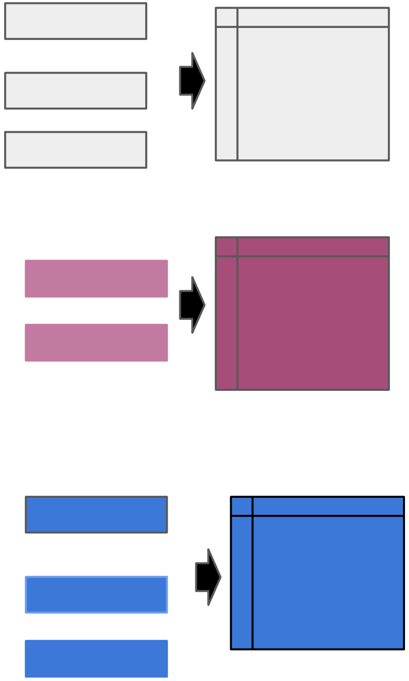
Mappers

Map-Shuffle-Reduce



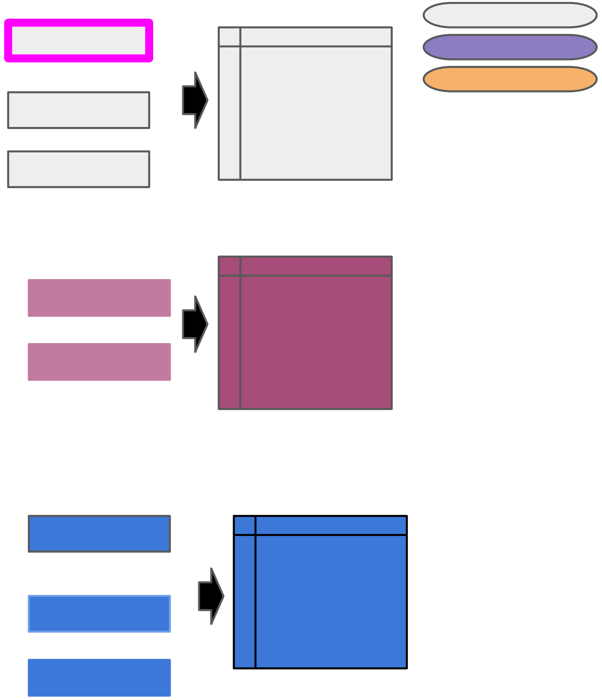
Mappers

Map-Shuffle-Reduce

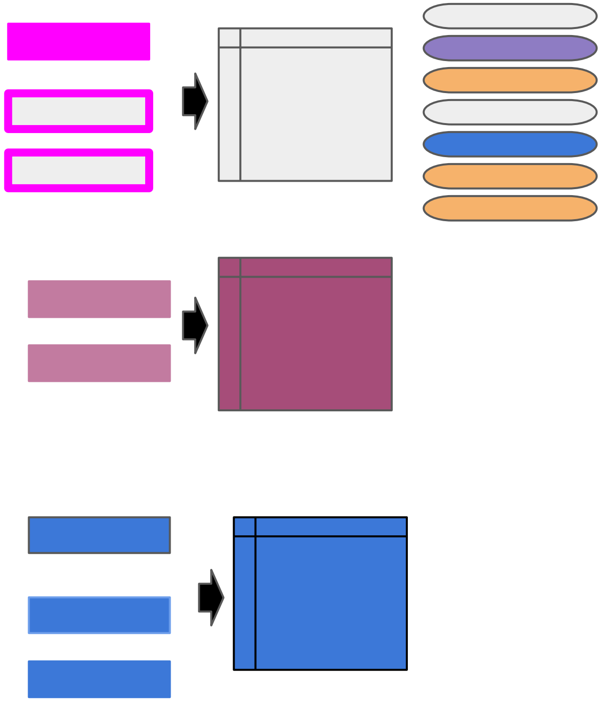


Mappers

Map-Shuffle-Reduce

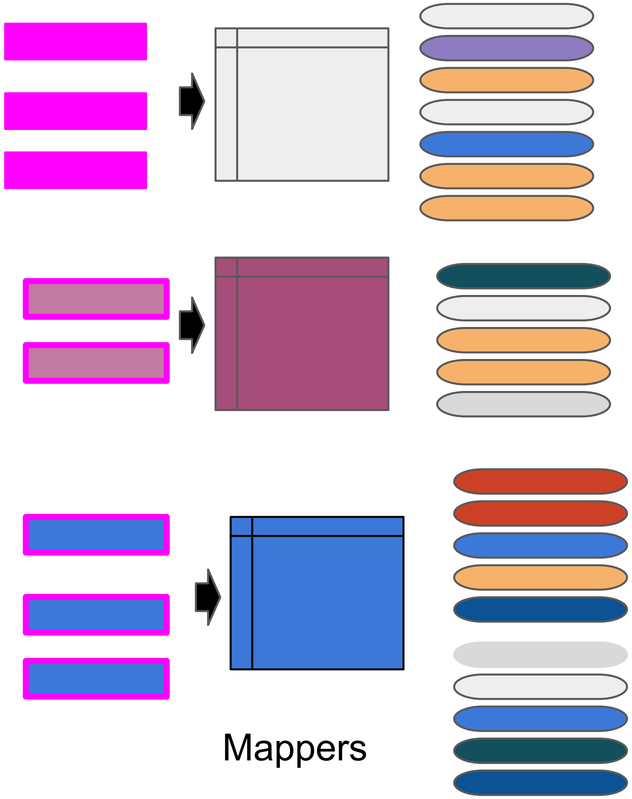


Map-Shuffle-Reduce

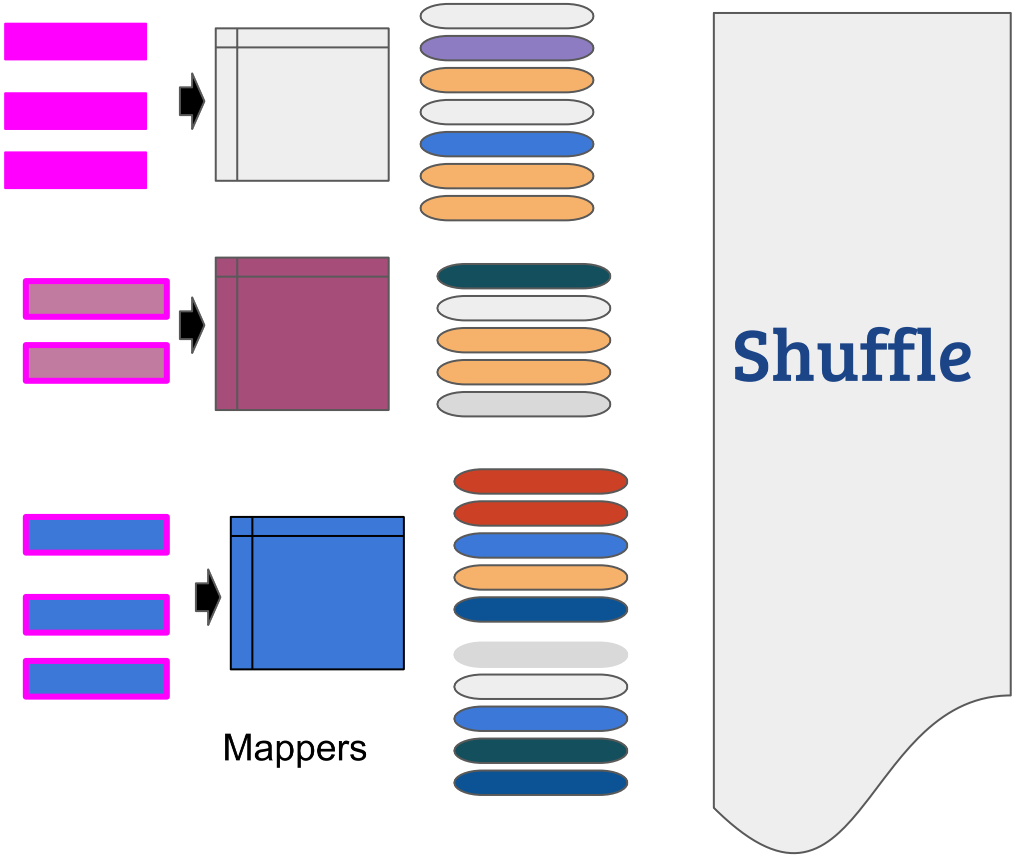


Mappers

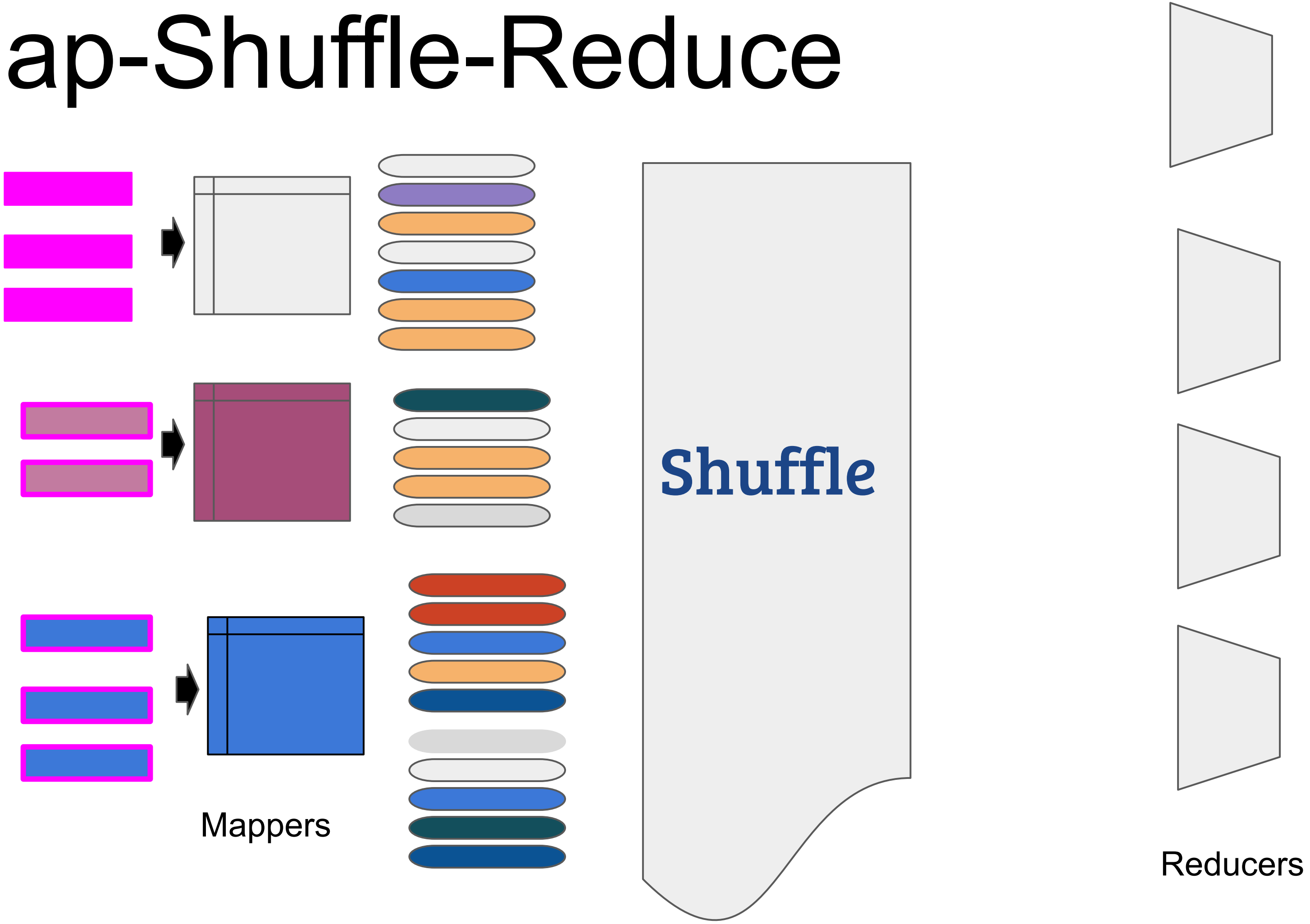
Map-Shuffle-Reduce



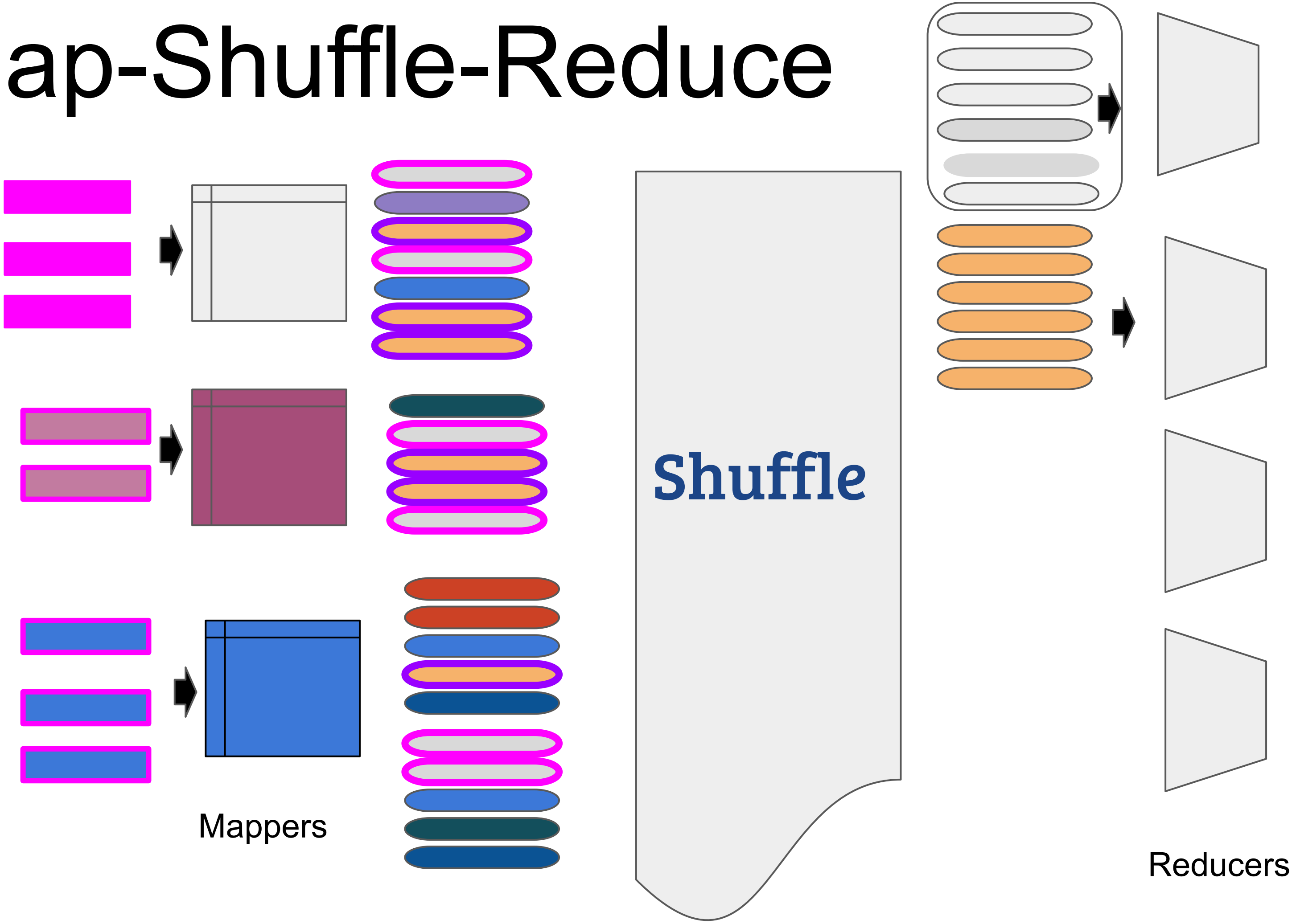
Map-Shuffle-Reduce



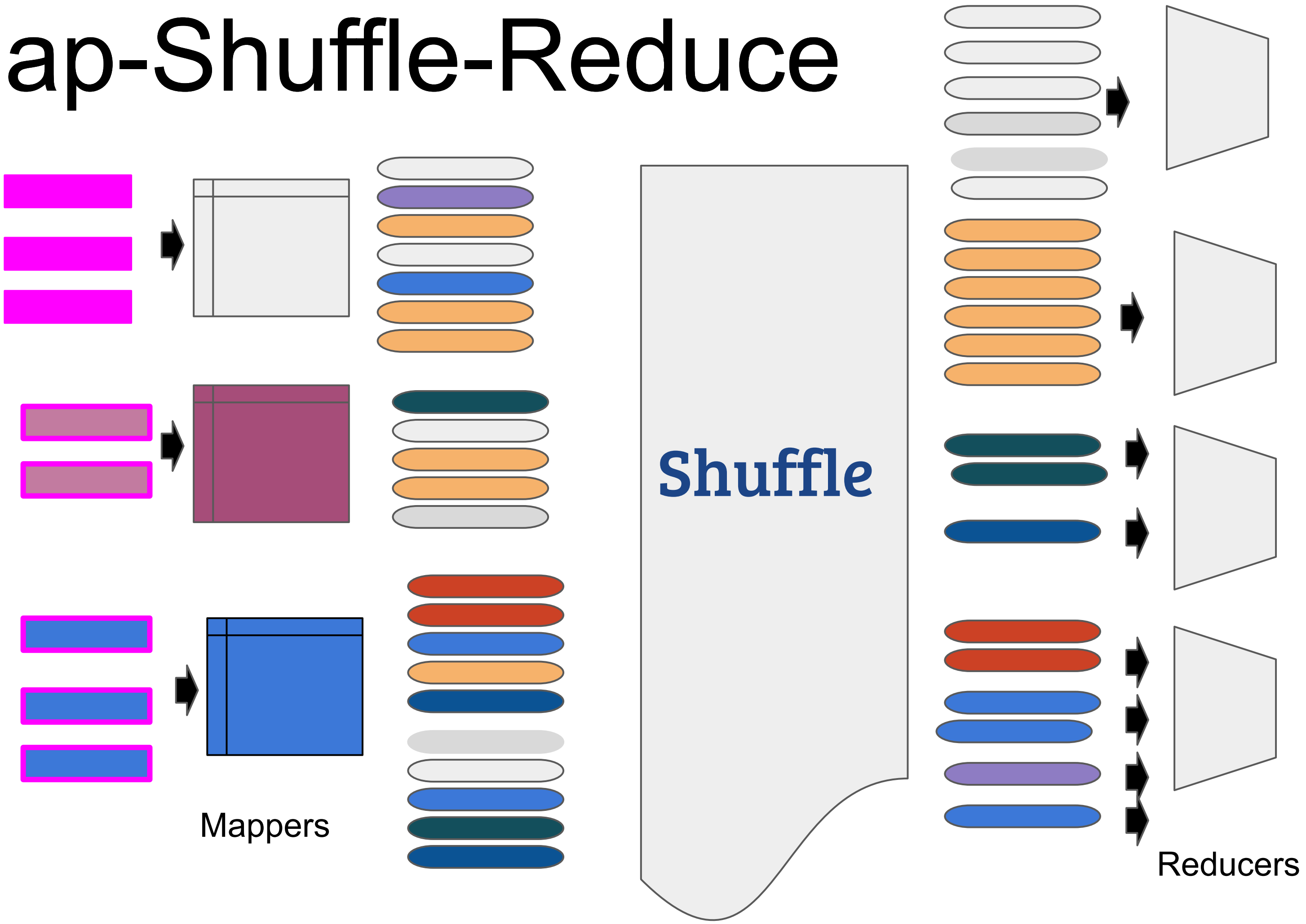
Map-Shuffle-Reduce



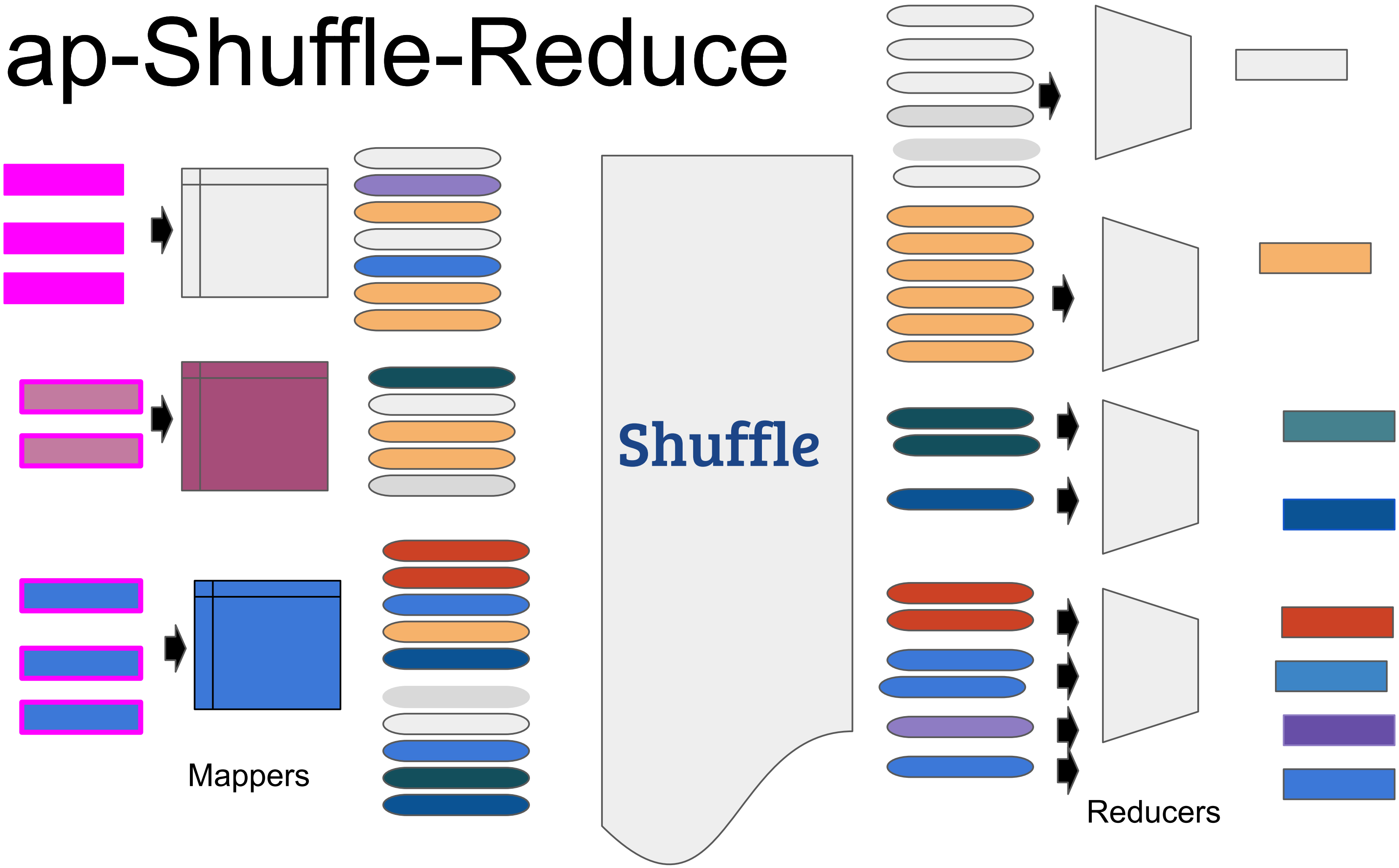
Map-Shuffle-Reduce



Map-Shuffle-Reduce



Map-Shuffle-Reduce



Shuffle
Mappers
Reducers