MapReduce

MapReduce Outline

- MapReduce Architecture
- MapReduce Internals
- MapReduce Examples
- JobTracker Interface

MapReduce: A Real World Analogy

Coins Deposit

MapReduce: A Real World Analogy

Coins Deposit

Coins Counting Machine

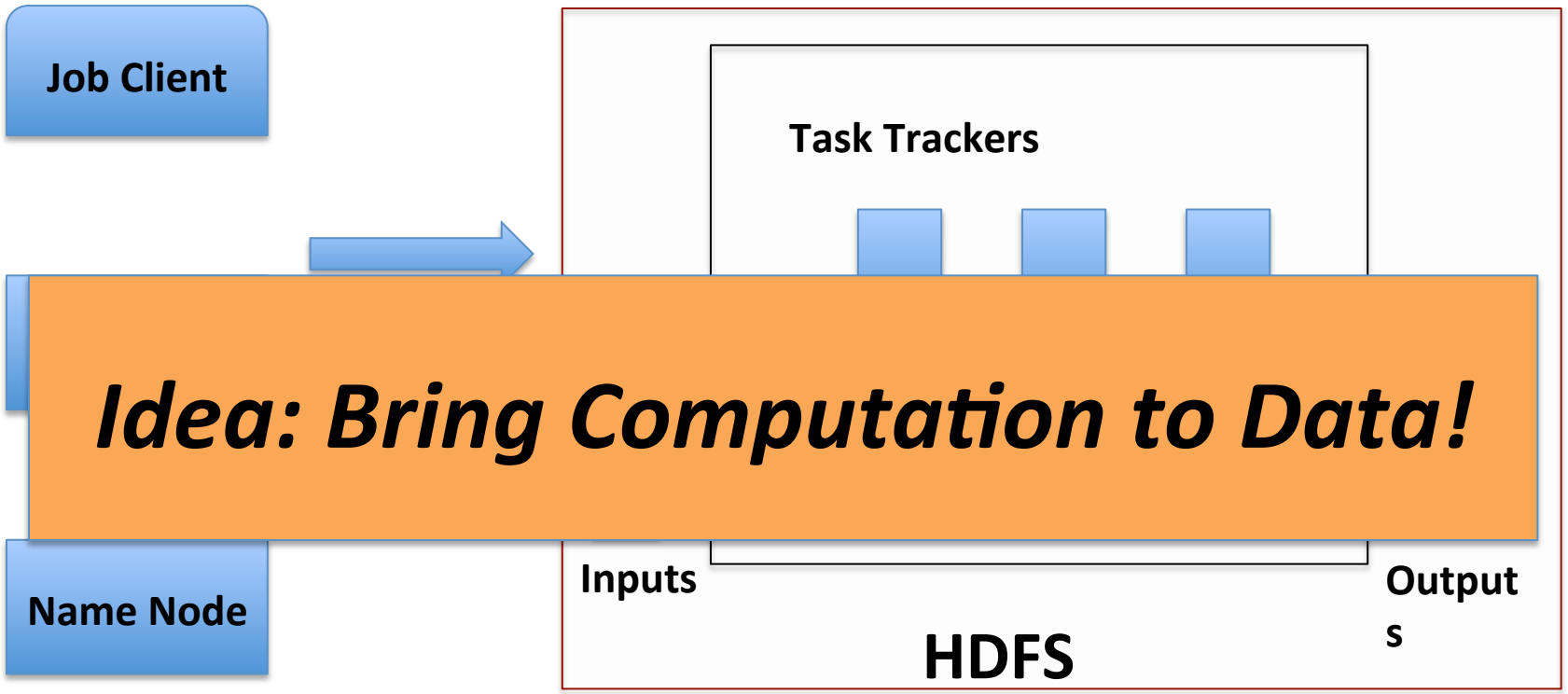
MapReduce: A Real World Analogy

Coins Deposit

Mapper: Categorize coins by their face values

Reducer: Count the coins in each face value ***in parallel***

MapReduce Architecture: Master-Slaves



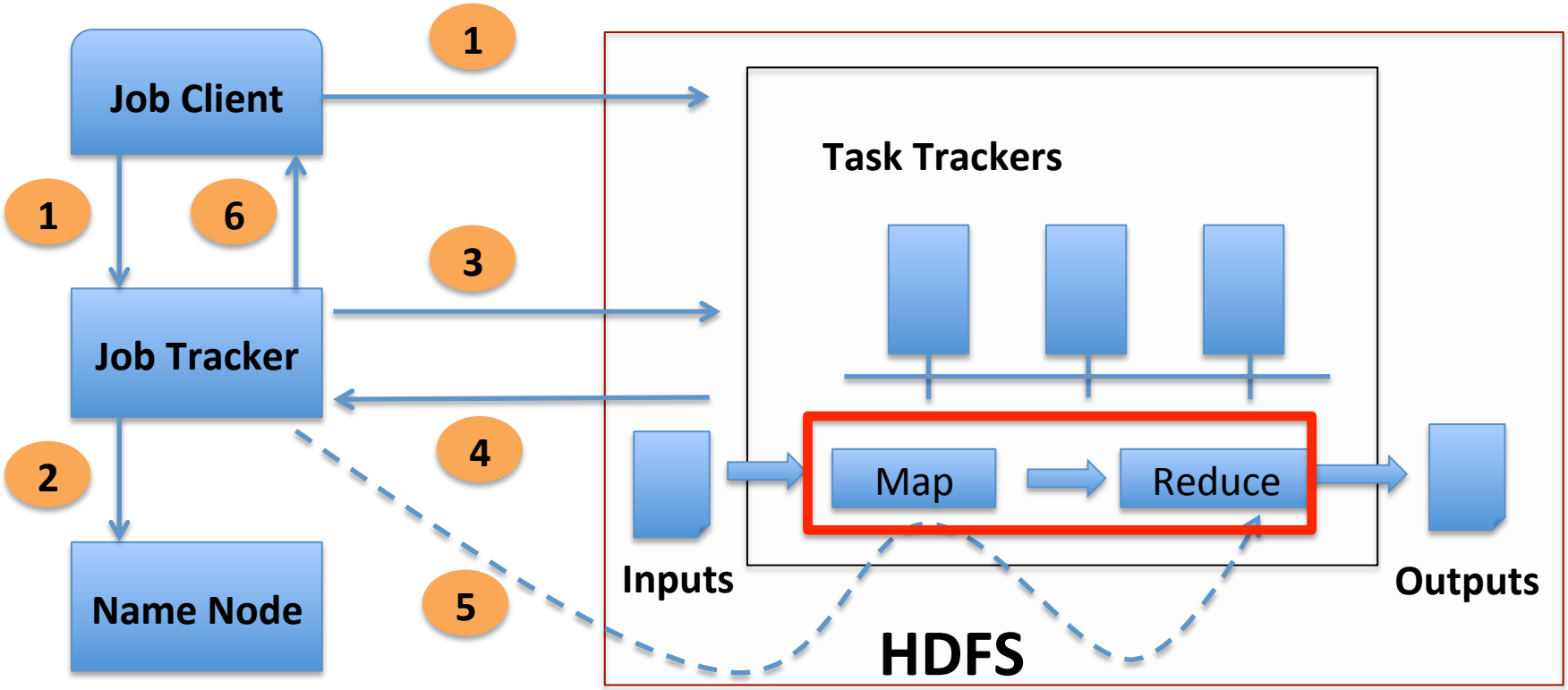
Job Client: Submit Jobs

Task Tracker: Execute Jobs

Job Tracker: Coordinate Jobs (Scheduling, Phase Coordination, etc.)

Job: MapReduce Function+ Config

MapReduce Architecture: Workflow

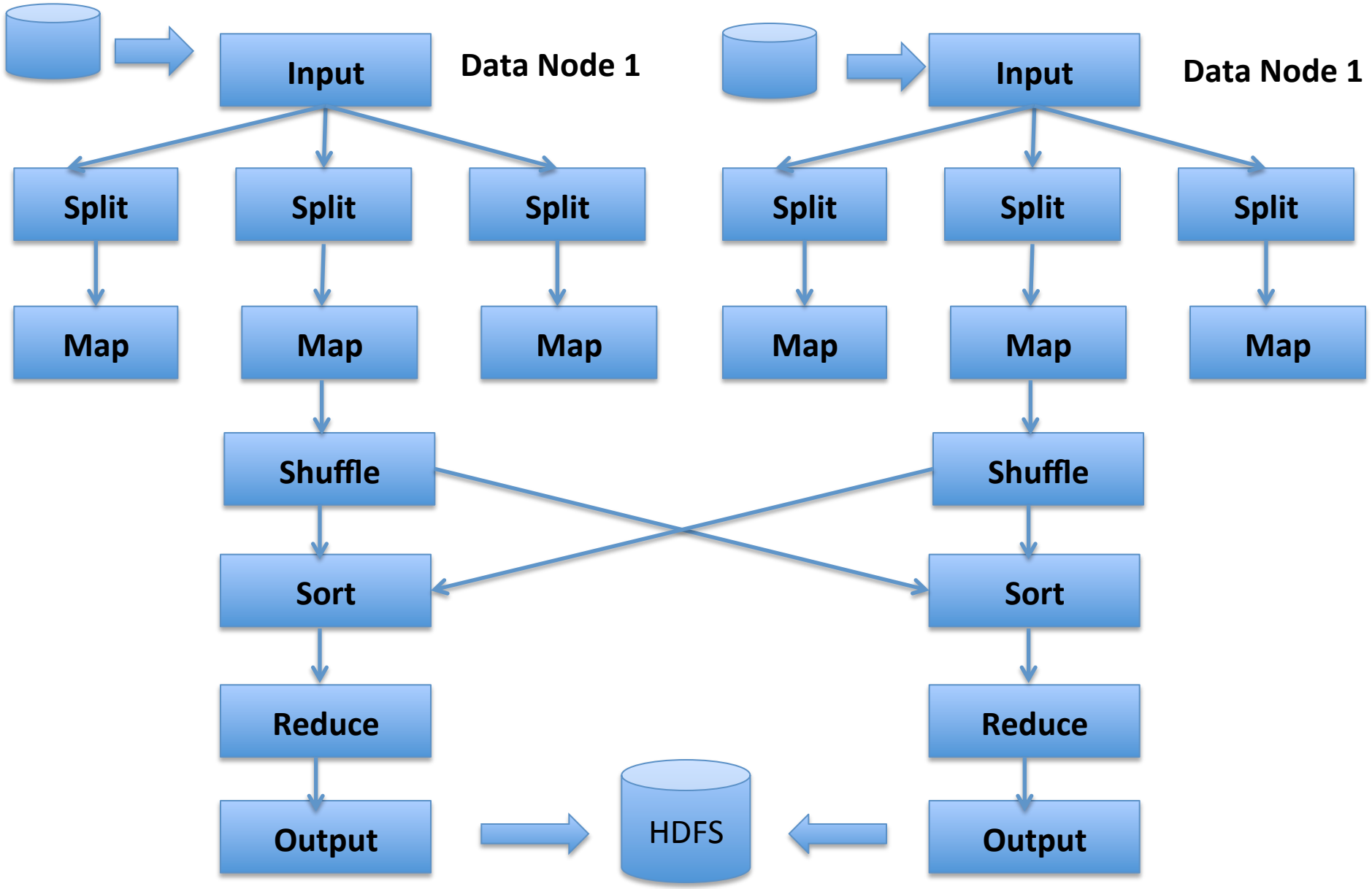


1. Client submits job to Job Tracker and copy code to HDFS
2. Job Tracker talks to NN to find data it needs
3. Job Tracker creates execution plan and submits work to Task Trackers
4. Task trackers do the job and report progress/status to Job Tracker
5. Job Tracker manages task phases
6. Job Tracker finishes the job and updates status

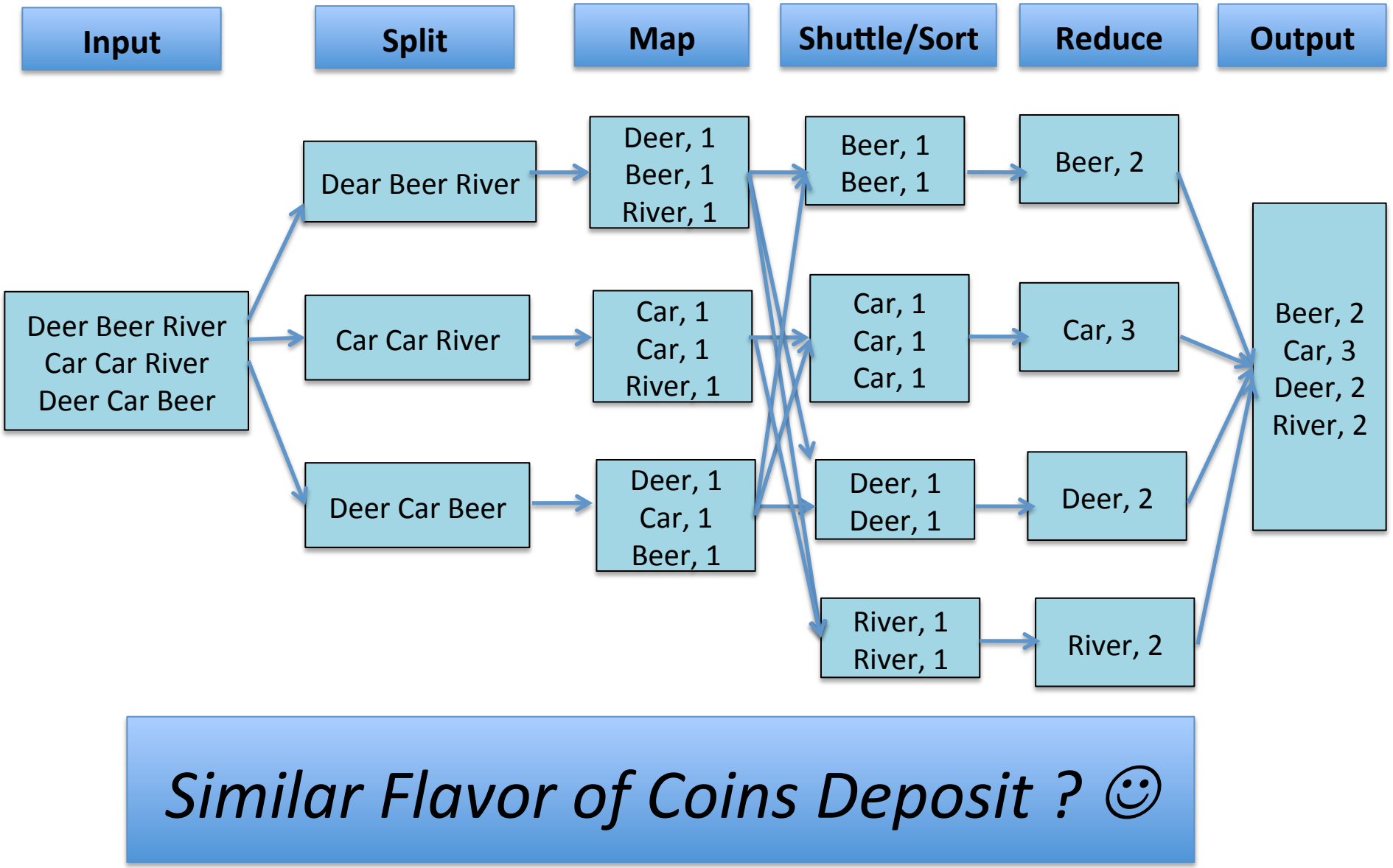
MapReduce Paradigm

- Implement two functions:
  - **Map** (k1,v1) -> list (k2, v2)
  - **Reduce**(k2, list(v2)) -> list (v3)
- Framework handles everything else
- Value with the **same key** go to the same reducer

MapReduce Internal

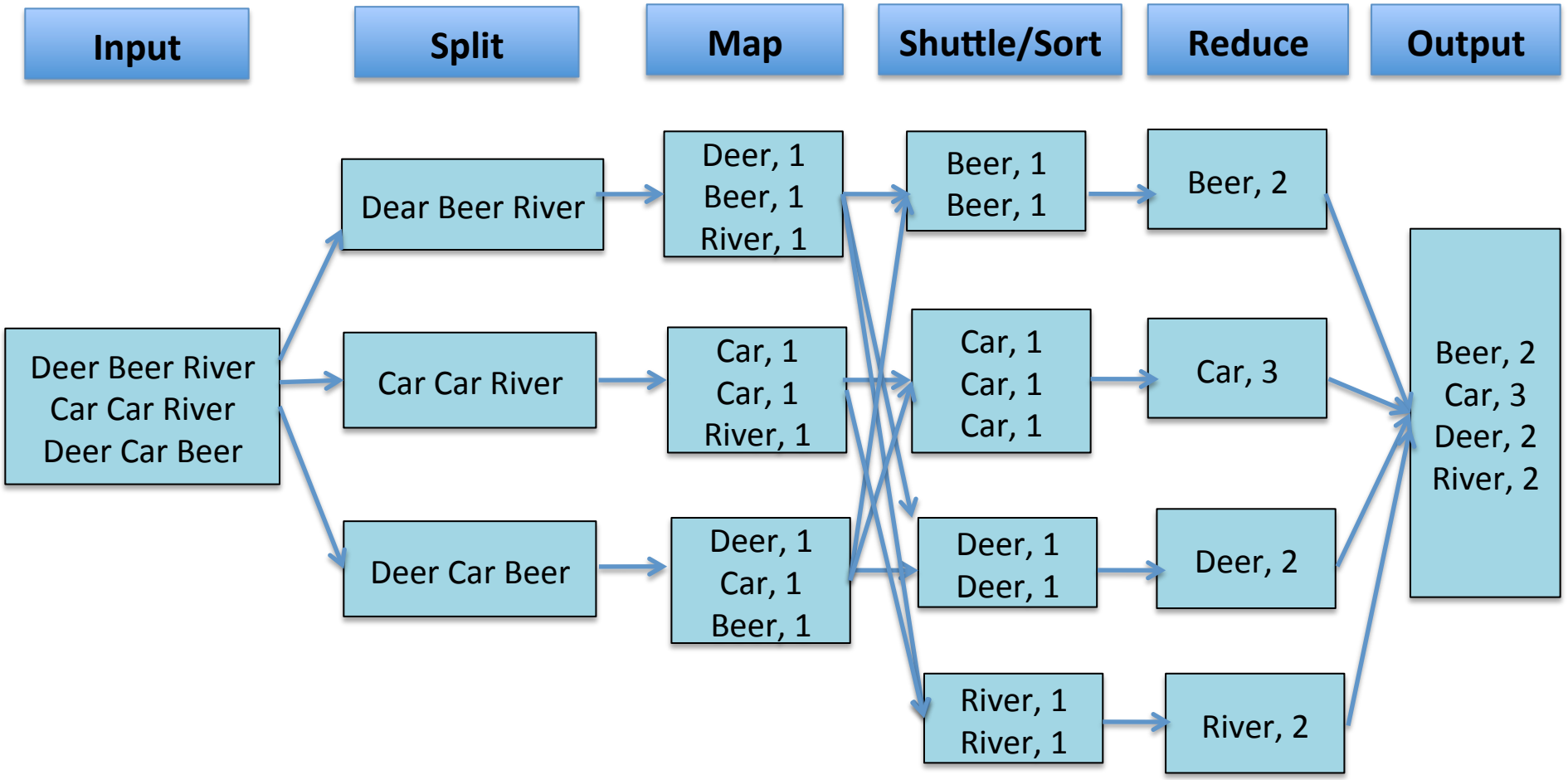


MapReduce Example: Word Count



Similar Flavor of Coins Deposit ? ☺

MapReduce Example: Word Count



Q: What are the Key and Value Pairs of Map and Reduce?

Map: Key=word, Value=1

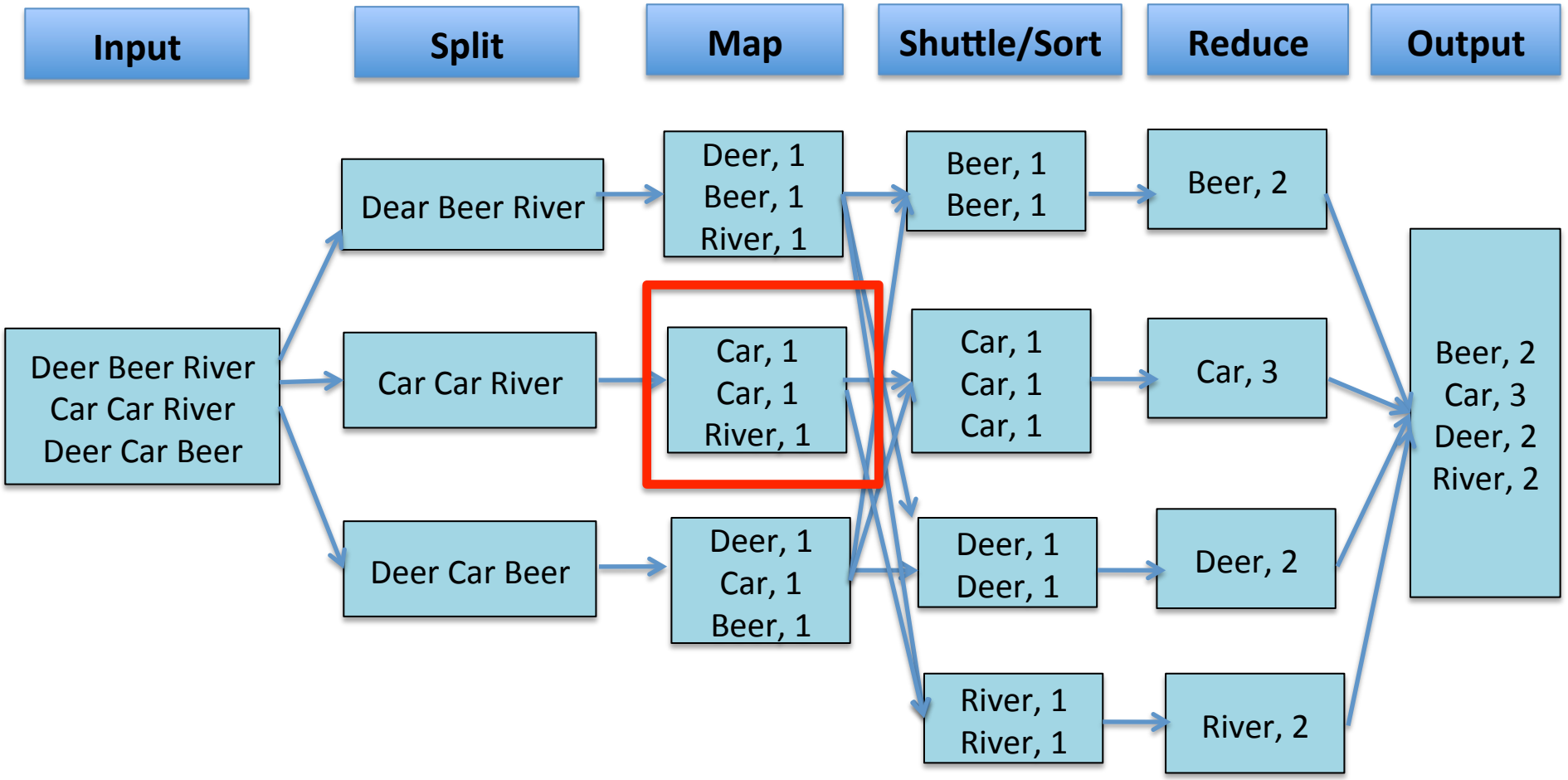
Reduce: Key=word, Value=aggregated count

Mapper and Reducer of Word Count

- Map(key, value){
  // key: line number
  // value: words in a line
  for each word w in value:
    Emit(w, "1");}
- Reduce(key, list of values){
  // key: a word
  // list of values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(key, result);}

Combiner is the same as Reducer

MapReduce Example: Word Count

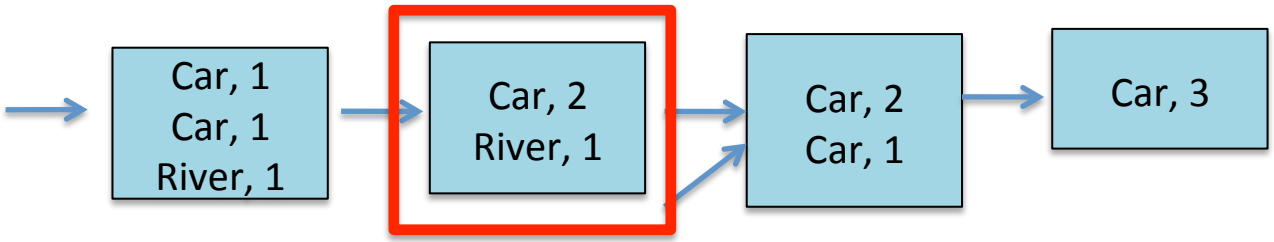


Q: Do you see any place we can improve the efficiency?

Local aggregation at mapper will be able to improve MapReduce efficiency.

MapReduce: Combiner

- **Combiner:** do local aggregation/combine task at mapper



- **Q: What are the benefits of using combiner:**
  - Reduce memory/disk requirement of Map tasks
  - Reduce network traffic
- **Q: Can we remove the reduce function?**
  - No, reducer still needs to process records with same key but from *different mappers*
- **Q: How would you implement combiner?**
  - It is the same as Reducer!

MapReduce WordCount 2

- **New Goal:** output all words sorted by their frequencies (total counts) in a document.
- **Question:** How would you adopt the basic word count program to solve it?
- **Solution:**
  - **Sort words by their counts in the reducer**
  - **Problem: what happens if we have more than one reducer?**

MapReduce WordCount 2

- **New Goal:** output all words sorted by their frequencies (total counts) in a document.
- **Question:** How would you adopt the basic word count program to solve it?
- **Solution:**
  - **Do two rounds of MapReduce**
  - **In the 2nd round, take the output of WordCount as input but *switch key and value pair*!**
  - **Leverage the sorting capability of *shuffle/sort* to do the global sorting!**

MapReduce WordCount 3

- **New Goal:** output the top K words sorted by their frequencies (total counts) in a document.
- **Question:** How would you adopt the basic word count program to solve it?
- **Solution:**
  - **Use the solution of previous problem and only grab the top K in the final output**
  - **Problem: is there a more efficient way to do it?**

MapReduce WordCount 3

- **New Goal:** output the top K words sorted by their frequencies (total counts) in a document.
- **Question:** How would you adopt the basic word count program to solve it?
- **Solution:**
  - **Add a sort function to the *reducer* in the first round and only output the top K words**
  - **Intuition: the *global top K must be a local top K* in any reducer!**

MapReduce In-class Exercise

- **Problem:** Find the maximum monthly temperature for each year from weather reports
- **Input:** A set of records with format as:

  <Year/Month, Average Temperature of that month>

  - (200707,100), (200706,90)

  - (200508, 90), (200607,100)

  - (200708, 80), (200606,80)
- **Question:** write down the Map and Reduce function to solve this problem
  - Assume we split the input by line

Mapper and Reducer of Max Temperature

- Map(key, value){

  // key: line number

  // value: tuples in a line

  for each tuple t in value:

  Emit(t->year, t->temperature);}

- Reduce(key, list of values){

  // key: year

  //list of values: a list of monthly temperature

  int max_temp = -100;

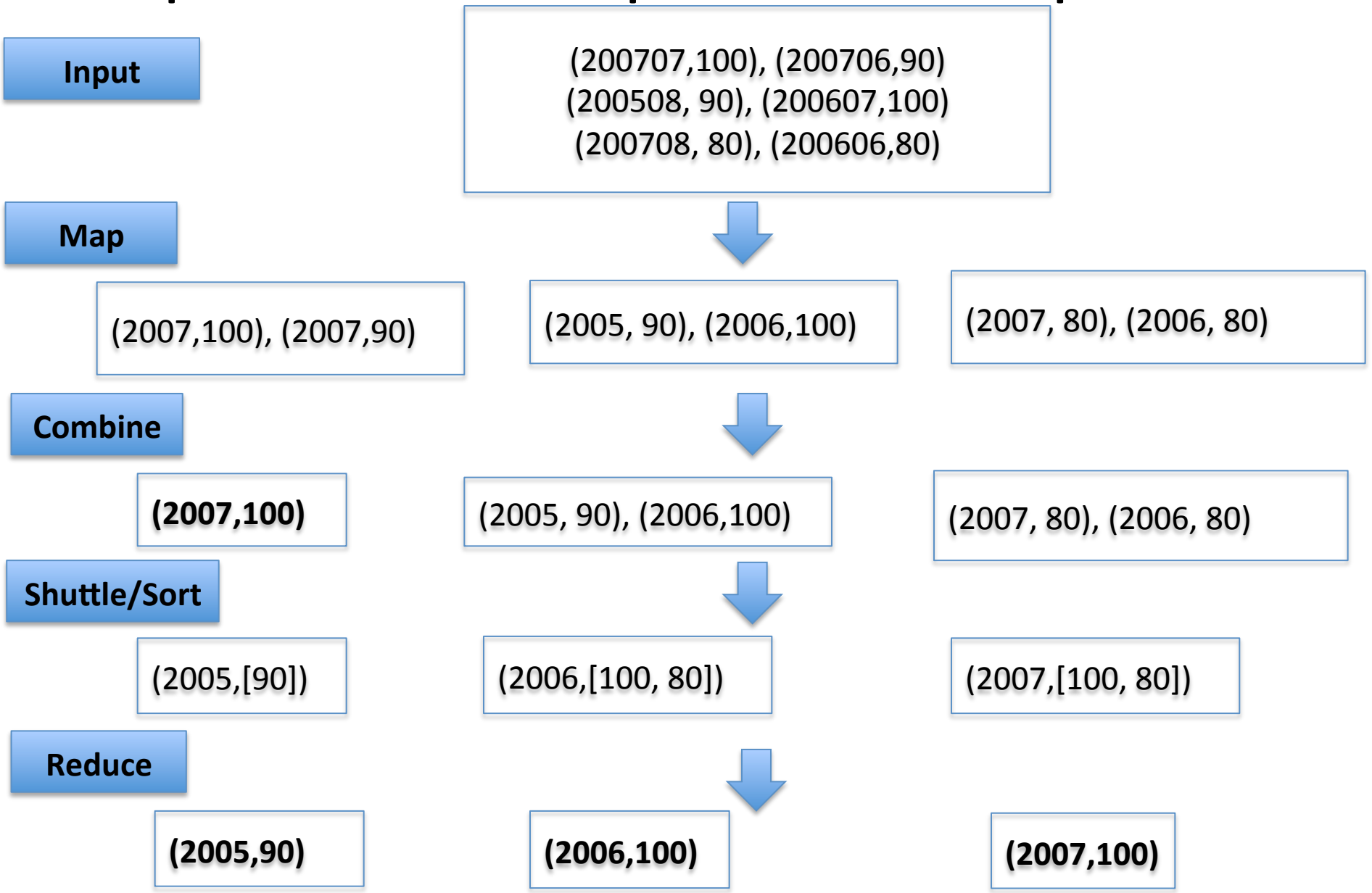
  for each v in values:

  max_temp= max(v, max_temp);

  Emit(key, max_temp);}

Combiner is the same as Reducer

MapReduce Example: Max Temperature



MapReduce In-class Exercise

- **Key-Value Pair of Map and Reduce:**
  - **Map:** (year, temperature)
  - **Reduce:** (year, maximum temperature of the year)

- **Question: How to use the above Map Reduce program *(that contains the combiner)* with slight changes to find the average monthly temperature of the year?**

Mapper and Reducer of Average Temperature

- Map(key, value){

  // key: line number

  // value: tuples in a line

  for each tuple t in value:

  Emit(t->year, t->temperature);}

- Reduce(key, list of values){

  // key: year

  // list of values: a list of monthly temperatures

  int total_temp = 0;

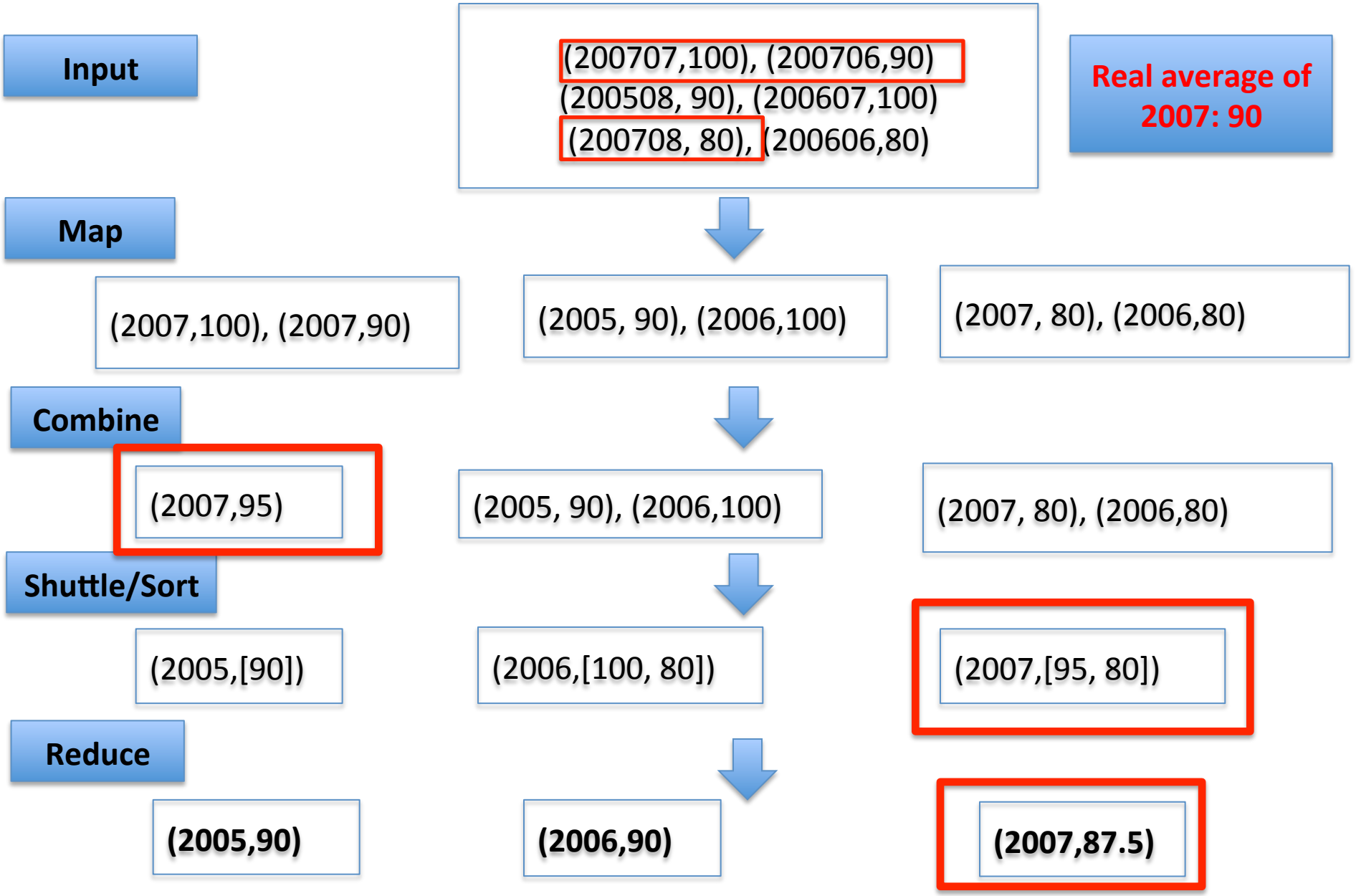
  for each v in values:

  total_temp= total_temp+v;

  Emit(key, total_temp/size_of(values));}

Combiner is the same as Reducer

MapReduce Example: Average Temperature

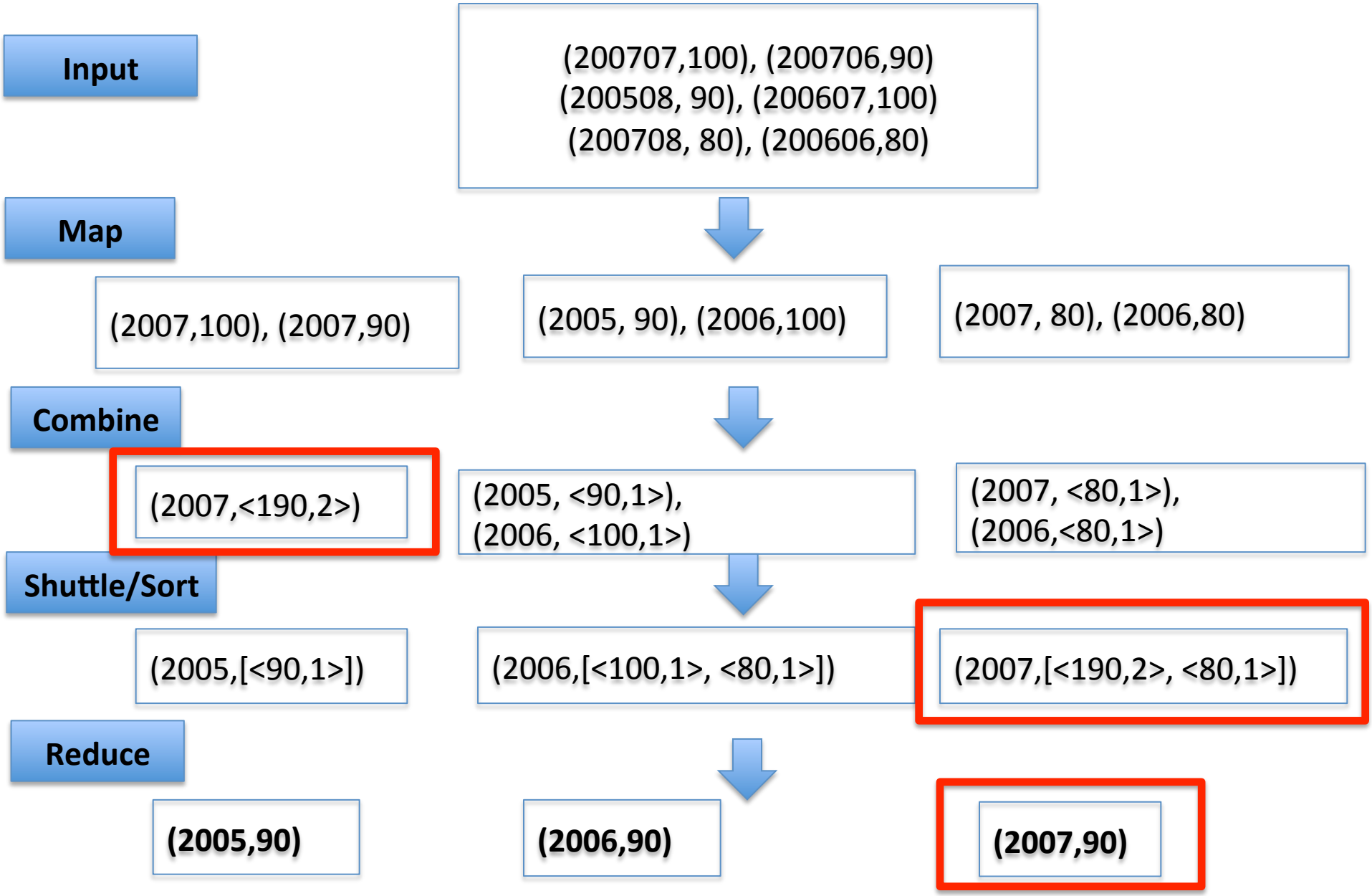


MapReduce In-class Exercise

- **The problem is with the combiner!**
- **Here is a simple counterexample:**
  - (2007, 100), (2007,90) -> (2007, 95)
    (2007,80)->(2007,80)
  - Average of the above is: (2007,87.5)
  - However, the real average is: (2007,90)

- **However, we can do a small trick to get around this**
  - Mapper: (2007, 100), (2007,90) -> (2007, <190,2>)
    (2007,80)->(2007,<80,1>)
  - Reducer: (2007,<270,3>)->(2007,90)

MapReduce Example: Average Temperature



Mapper and Reducer of Average Temperature

- Map(key, value){
  // key: line number
  // value: tuples in a line
  for each tuple t in value:
    Emit(t->year, t->temperature);}

- **Reduce (key, list of values)**{
  // key: year
  // list of values: a list of <temperature sums, counts> tuples
  int total_temp = 0;
  int total_count=0;
  for each v in values:
    total_temp= total_temp+v->sum;
    total_count=total_count+v->count;
  Emit(key,**total_temp/total_count**);}

- **Combine(key, list of values)**{
  // key: year
  // list of values: a list of monthly temperature
  int total_temp = 0;
  for each v in values:
    total_temp= total_temp+v;
  Emit(key,**<total_temp,size_of(values)>**);}

MapReduce In-class Exercise

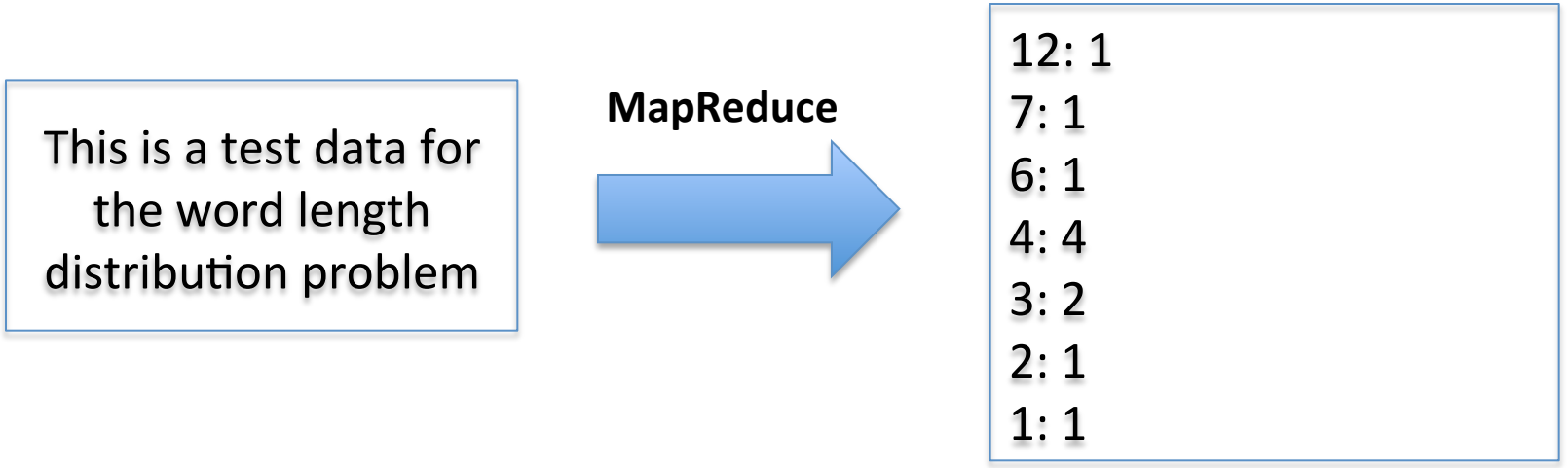
- **Functions that can use combiner are called *distributive*:**
  - Distributive: Min/Max(), Sum(), Count(), TopK()
  - Non-distributive: Mean(), Median(), Rank()

Gray, Jim*, et al. "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals." Data Mining and Knowledge Discovery 1.1 (1997): 29-53.

*Jim Gray received Turing Award in 1998

Map Reduce Problems Discussion

- **Problem 1:** Find Word Length Distribution
- **Statement:** Given a set of documents, use Map-Reduce to find the length distribution of all words contained in the documents
- **Question:**
  - What are the Mapper and Reducer Functions?



Mapper and Reducer of Word Length Distribution

- Map(key, value){
  // key: document name
  // value: words in a document
  for each word w in value:
    Emit(length(w), w);}
- Reduce(key, list of values){
  // key: length of a word
  // list of values: a list of words with the same length
  Emit(key, size_of(values));}

Map Reduce Problems Discussion

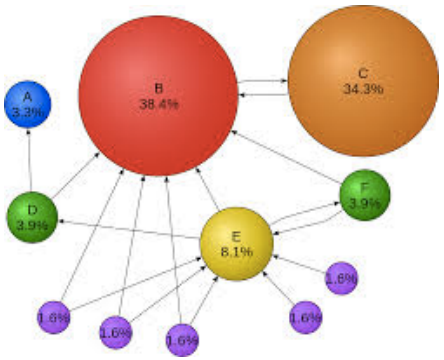
- **Problem 1:** Find Word Length Distribution
- **Mapper and Reducer:**
  - Mapper(document)

    { Emit (**Length(word), word**) }
  - Reducer(output of map)

    { Emit (Length(word), **Size of (List of words at a particular length))**}

Map Reduce Problems Discussion

- **Problem 2:** Indexing & Page Rank
- **Statement:** Given a set of web pages, each page has a **page rank** associated with it, use Map-Reduce to find, for each word, *a list of pages (sorted by rank)* that contains that word
- **Question:**
  - What are the Mapper and Reducer Functions?



MapReduce

Word 1: [page x1, page x2, ..]

Word 2: [page y1, page y2, …]

…

Mapper and Reducer of Indexing and PageRank

- Map(key, value){

  // key: a page

  // value: words in a page

  for each word w in value:

  Emit(w, <page_id, page_rank>);}

- Reduce(key, list of values){

  // key: a word

  // list of values: a list of pages containing that word

  sorted_pages=sort(values, page_rank)

  Emit(key, sorted_pages);}

Map Reduce Problems Discussion

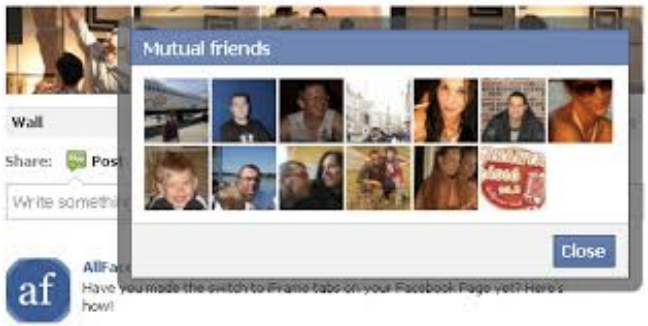
- **Problem 2:** Indexing and Page Rank
- **Mapper and Reducer:**
  - Mapper(page_id, <page_text, page_rank>)

    { Emit (**word, <page_id, page_rank>**) }
  - Reducer(output of map)

    { Emit (word, **List of pages contains the word sorted by their page_ranks)**}

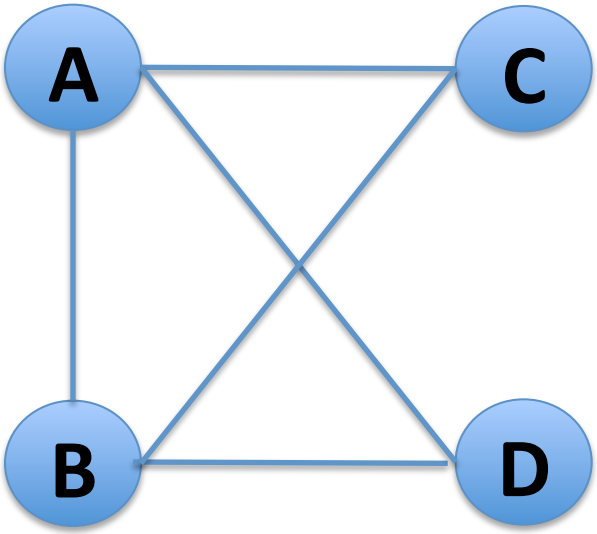
Map Reduce Problems Discussion

- **Problem 3:** Find Common Friends
- **Statement:** Given a group of people on online social media (e.g., Facebook), each has a list of friends, use Map-Reduce to find *common friends* of any two persons who are friends
- **Question:**
  - What are the Mapper and Reducer Functions?



Map Reduce Problems Discussion

- **Problem 3:** Find Common Friends
- **Simple example:**



Input:

A -> B,C,D

B-> A,C,D

C-> A,B

D->A,B

MapReduce

Output:

(A ,B) -> C,D

(A,C) -> B

(A,D) -> ..

….

Mapper and Reducer of Common Friends

- Map(key, value){

  // key: person_id

  // value: the list of friends of the person

  for each friend f_id in value:

  Emit(<person_id, f_id>, value);}

- Reduce(key, list of values){

  // key: <friend pair>

  // list of values: a set of friend lists related with the friend pair

  for v1, v2 in values:

  common_friends = v1 intersects v2;

  Emit(key, common_friends);}

Map Reduce Problems Discussion

- **Problem 3:** Find Common Friends
- **Mapper and Reducer:**
  - Mapper(friend list of a person)

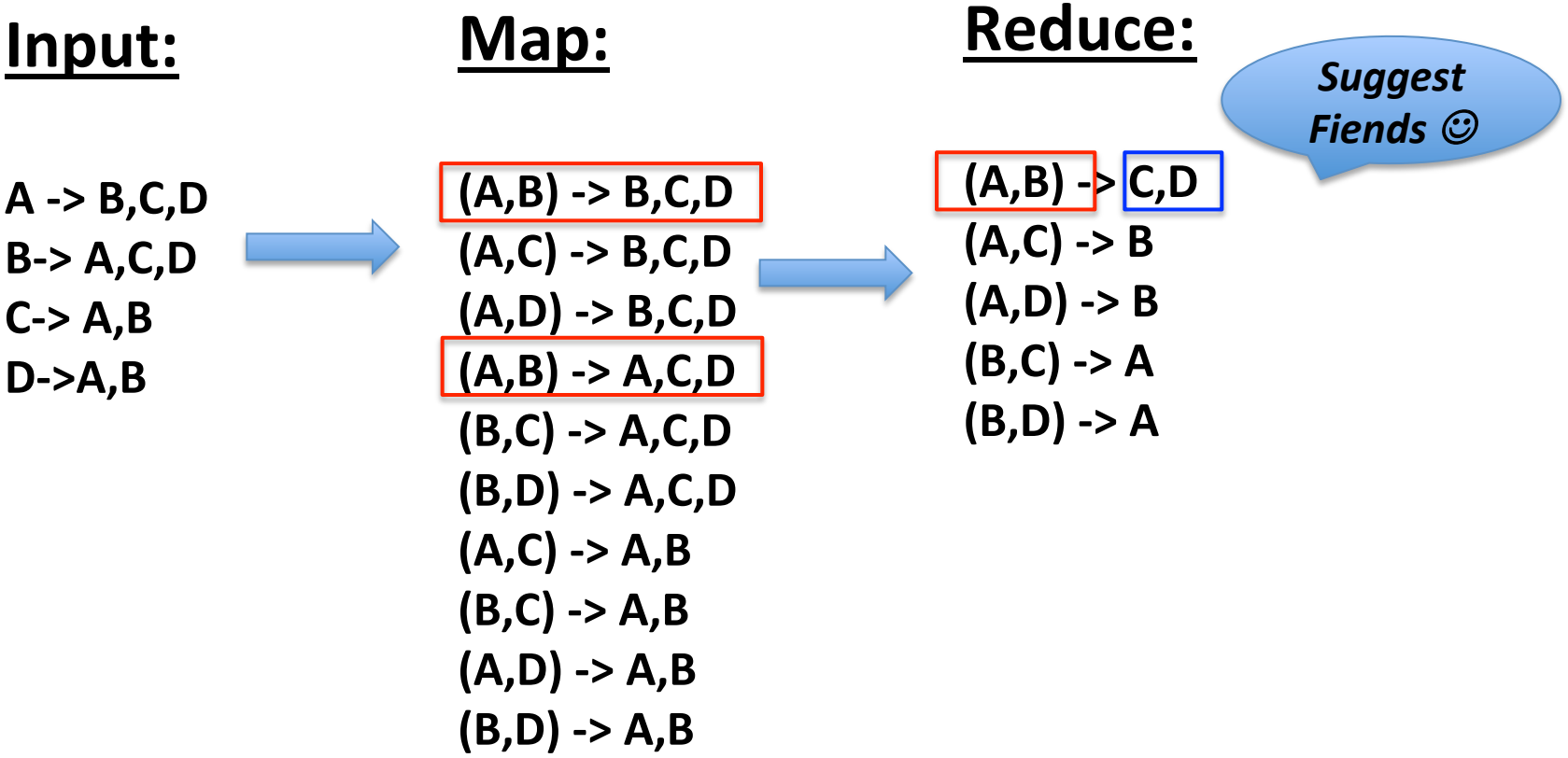
    { for each person in the friend list:

    Emit (**<friend pair>, <list of friends>**) }
  - Reducer(output of map)

    { Emit (<friend pair>, **Intersection of two (i.e, the one in friend pair) friend lists)**}

Map Reduce Problems Discussion

- **Problem 3:** Find Common Friends
- **Mapper and Reducer:**



Enjoy MR and Hadoop☺