# Quiz Logistics

Quiz time: 5:30-5:50pm

In Class:

5:15pm arrive, check-in with your USC ID at the door

Online:

5:15pm join WebEx on DEN using your phone and turn on your camera as instructed

5:30pm, use your laptop and go to DEN,

myTool->Quizzes, enter lockdown browser

type in password

# Map-Reduce (Part II)

Professor Wei-Min Shen

University of Southern California

Thanks for source slides and material to: J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets (http://www.mmds.org)

# OUTINE

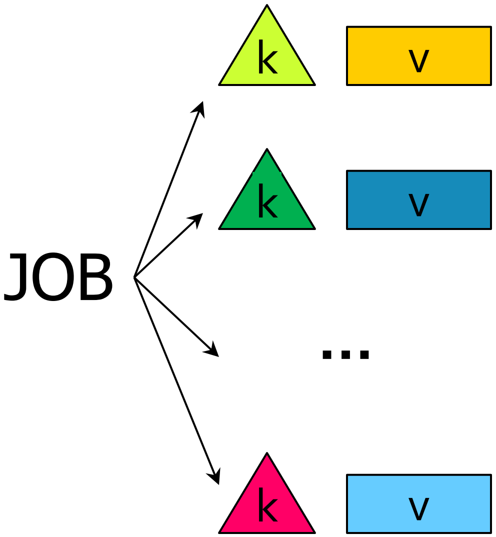
- Map-Reducer (Part II)
- Spark and Scala (Introduction)
- Quizzes and Homeworks

# Map-Reduce: Overview

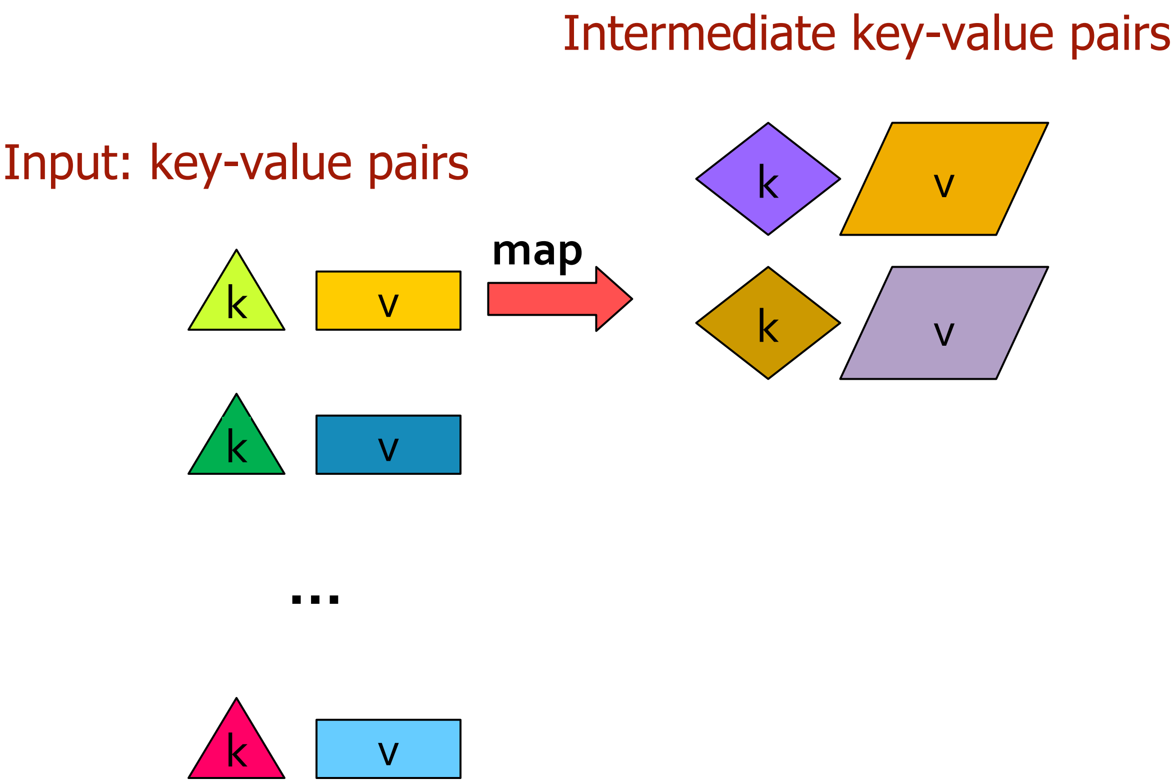
- **Map** ( dividing )
  - Divide a big job into many smaller local jobs
  - For each local job, extract an output pairs (key, values)
- **Group by key** ( gathering )
  - Sort, shuffle, group by keys
- **Reduce** **(** harvesting **)**
  - Aggregate, summarize, filter or transform
  - Output the final result

# Map-Reduce: The Map Step

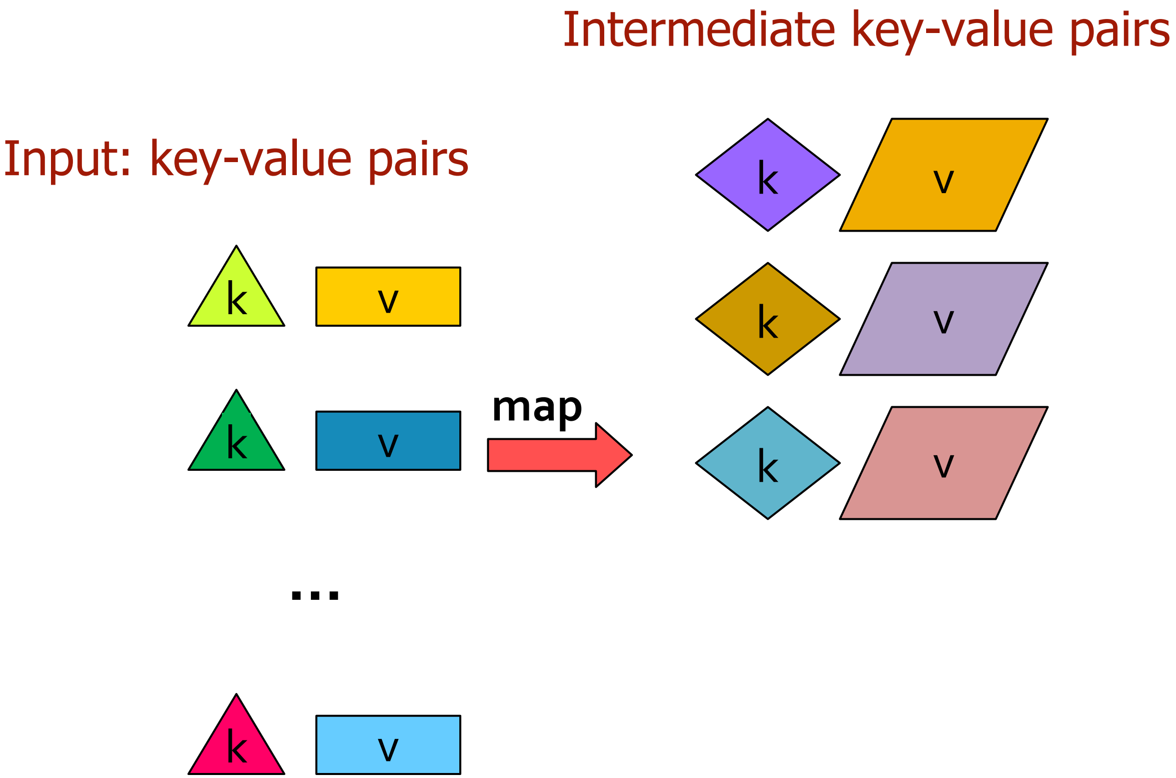
Input: key-value pairs



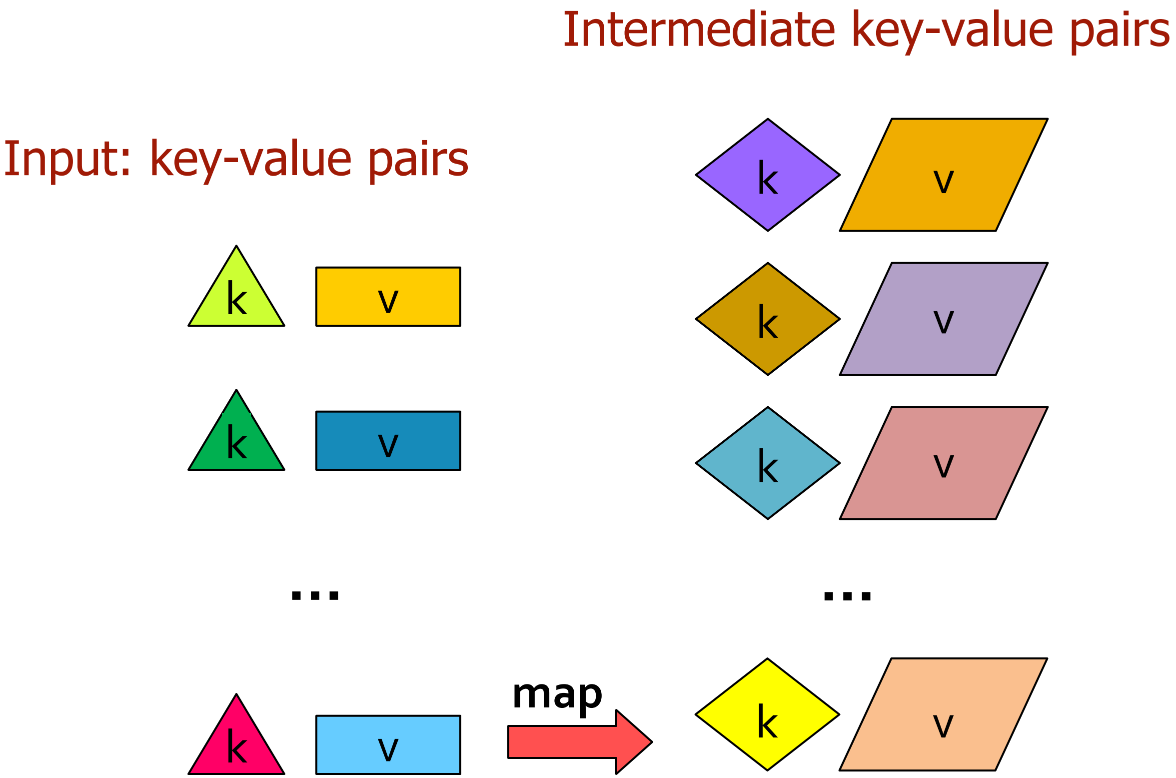
# Map-Reduce: The Map Step



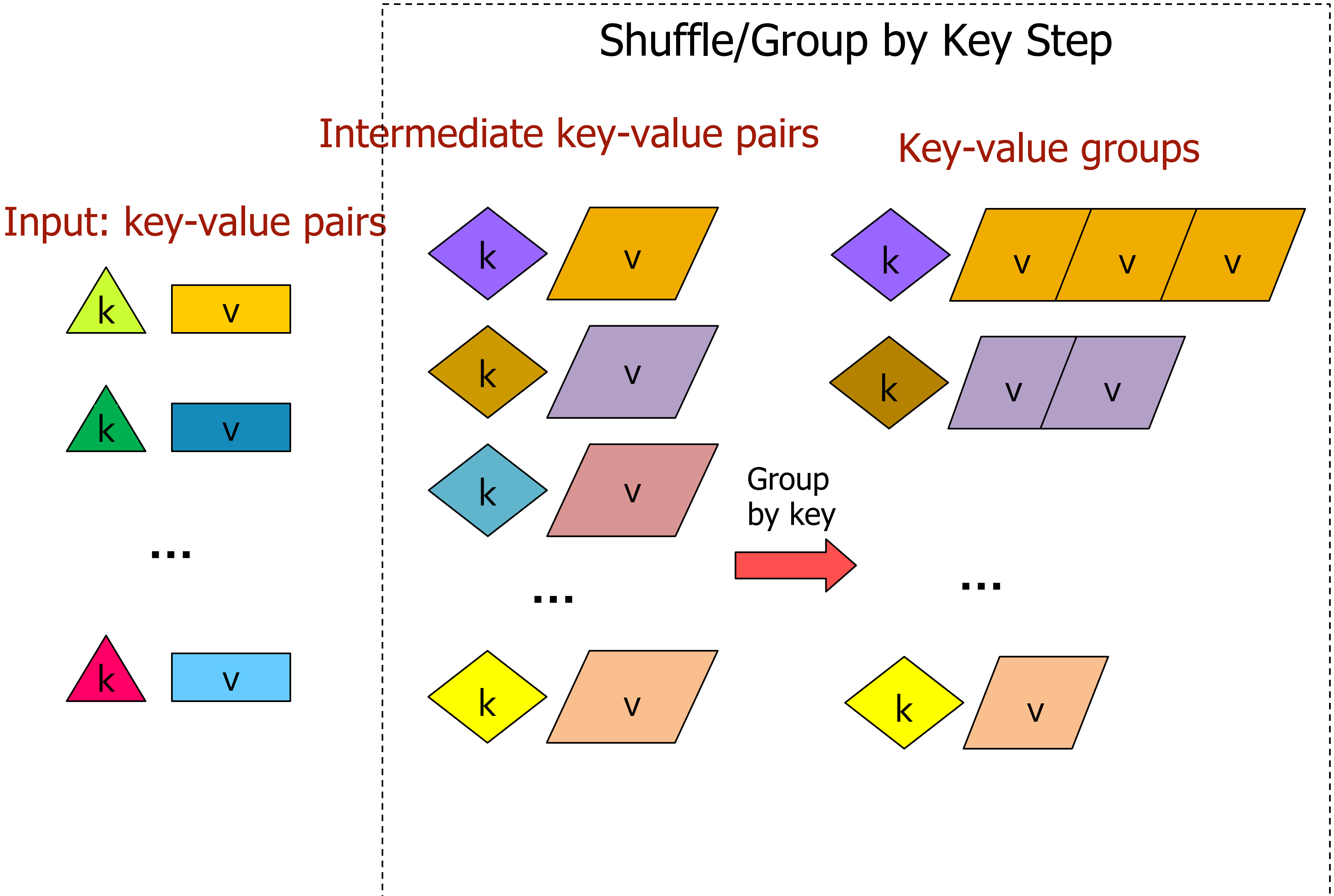
# Map-Reduce: The Map Step



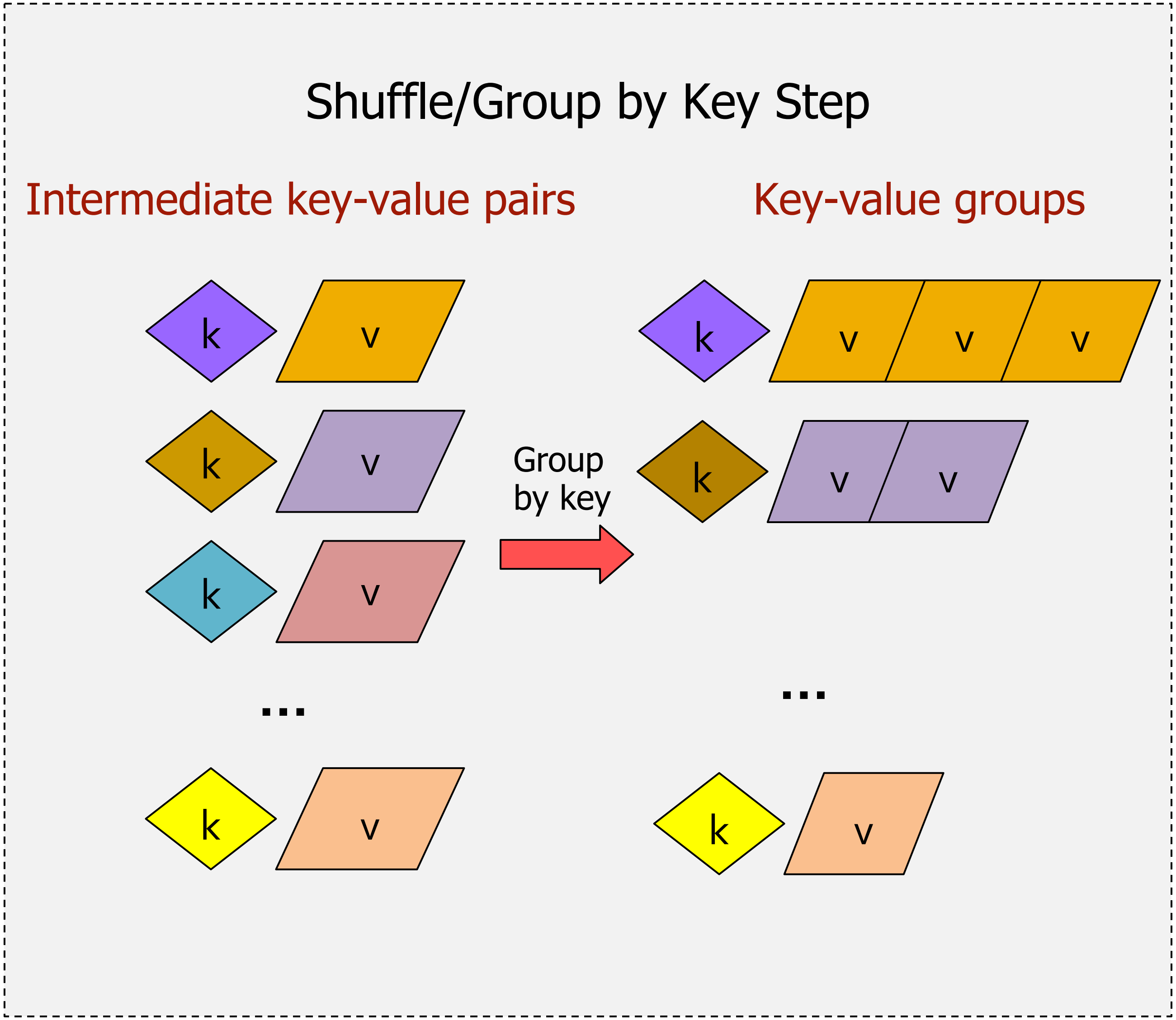
# Map-Reduce: The Map Step



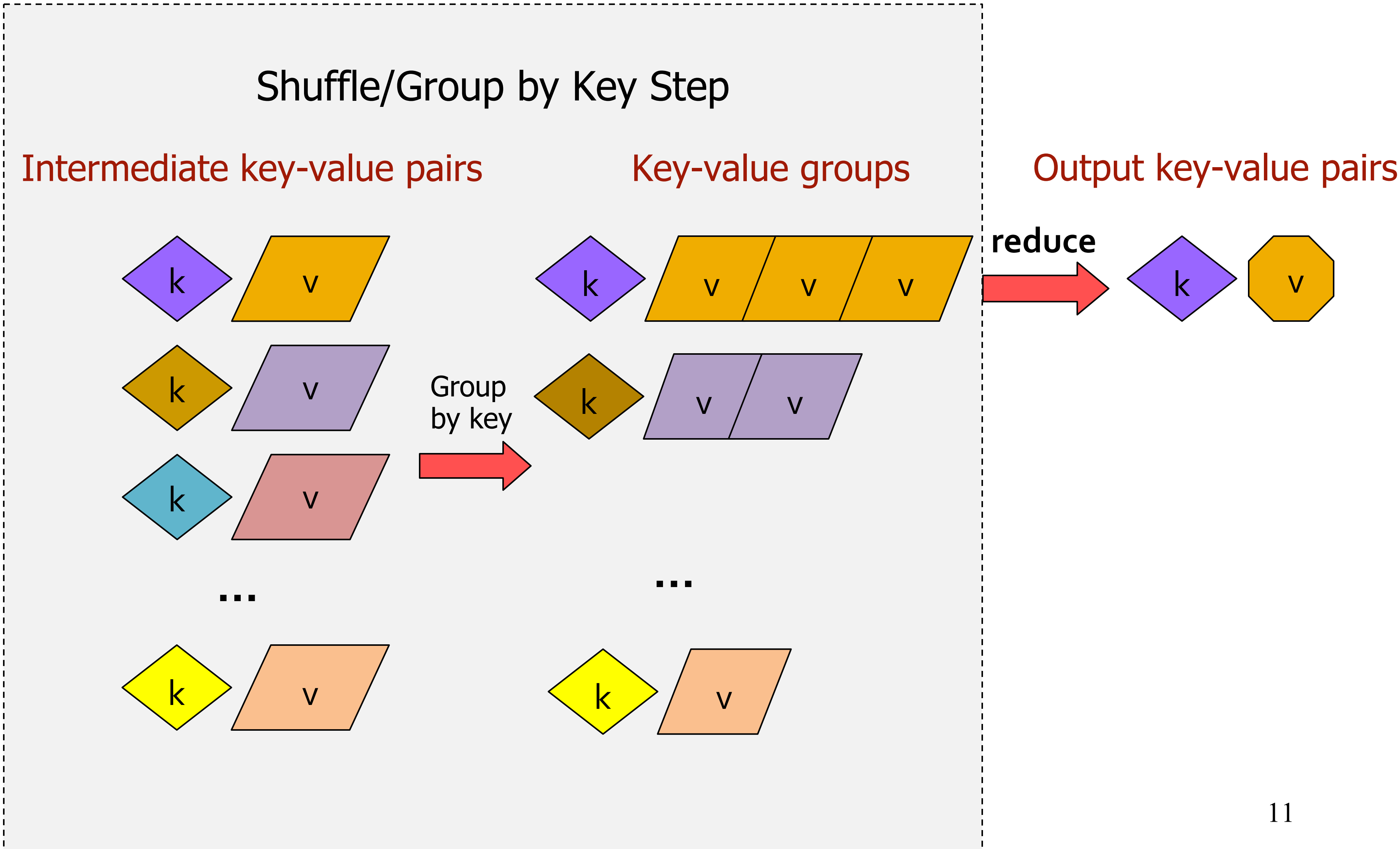
# Map-Reduce: The Reduce Step



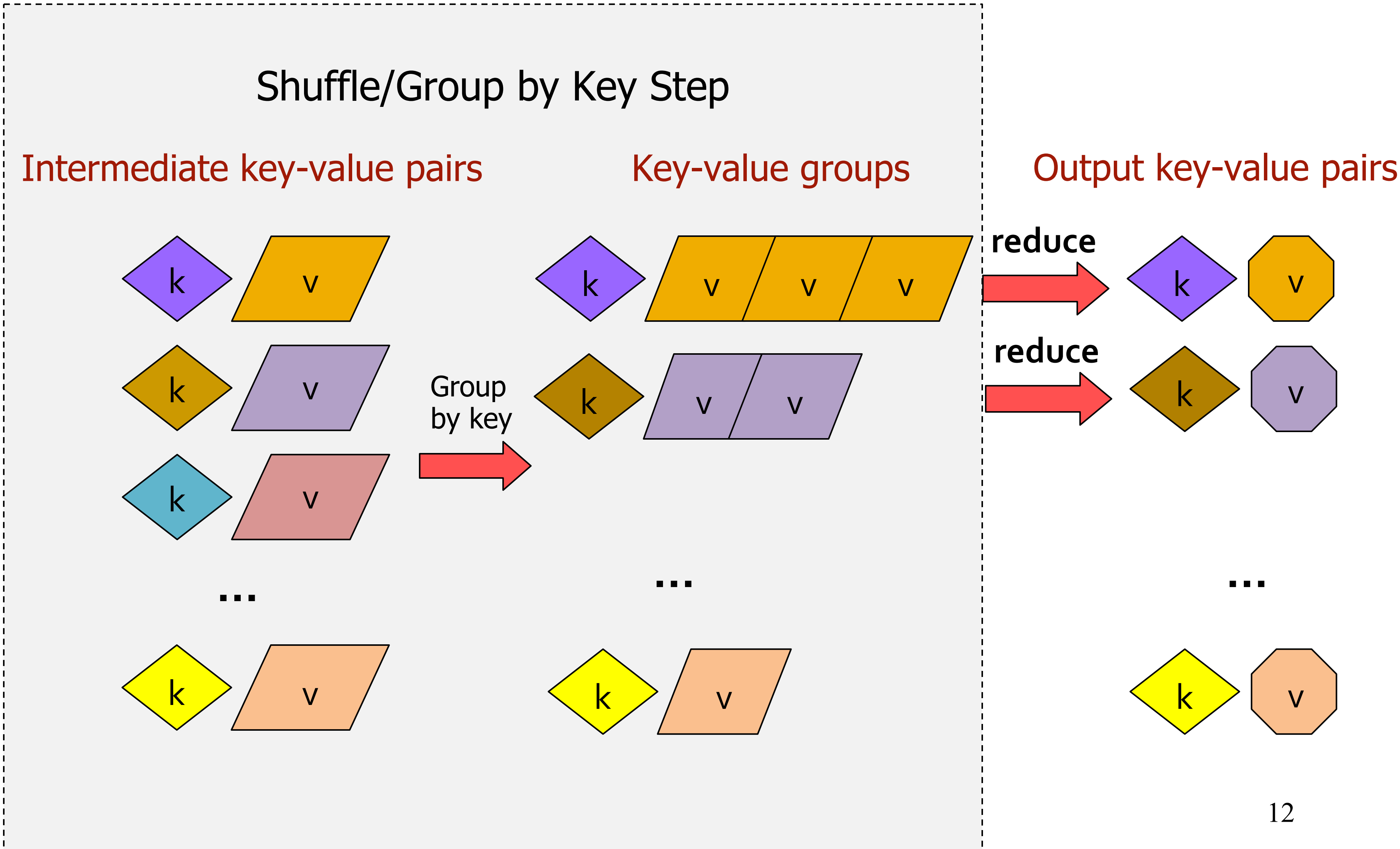
# Map-Reduce: The Reduce Step



# Map-Reduce: The Reduce Step



# Map-Reduce: The Reduce Step



# More Formally…

- **Input:** a set of key-value pairs
- Programmer need to specify two methods:
  - **Map(k, v)** → <k’, v’>*
    - Takes a key-value pair and outputs <u>a set of key-value pairs</u>

      E.g., Input: (k,v): k is the filename, v is the file

      Output: <k’,v’>*: k’ is the filename, v’ is a single line in the file

      E.g., Input: (k,v): k is the filename, v is a single line in the file

      Output: <k’,v’>*: k’ is a word, v’ is the count of that word
    - There is one Map call for every (k, v) pair

  - **Reduce(k’, <v’>*)** → <k’, v’’>*
    - **All values v’ with same key k’ are reduced together**
    - There is one Reduce function call per unique key k’

# Map-Reduce Examples (in Part I)

1. Word count: a huge file => (word, count)
2. Inverted Index: (ID, content)* => (content, List[IDs])
3. Count friends: Pair of friends (X, Y)* => (X, Nx), (Y, Ny), …
4. A huge file of integers => Unique integers divisible by 7

**Challenges**: What should the Map do? What should the Reduce do?

**Criteria**: the overall work and traffic must be as efficient as possible!

the speed is the ultimate judger!

# Map-Reduce example B1

## Find largest integer (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **the largest integer** as output
- The large file of integers cannot fit in the memory of node

**Map task:**

Each Map task gets a chunk of the file of integers and processes it …

**Reduce task:**

?

# Map-Reduce example B1
## Find largest integer (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **the largest integer** as output
- The large file of integers cannot fit in the memory of node

**Map task:**

Map task produces **(1, largest-integer)** of the largest value in the local chunk given as (key, value) pairs

(10, 1, 9, 3, 4, 5, 6, 7)
(6, 7, 19, 43, 54, 25, 61, 17)
(0, 15, 9, 43, 4, 15, 46, 7)

MAP

(1, 10)
(1, 61)
(1, 46)

# Map-Reduce example B1

## Find largest integer (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **the largest integer** as output
- The large file of integers cannot fit in the memory of node

**Group by key:**

(1, 10)
(1, 61)
(1, 46)

GROUP BY KEY

(1, (10, 61, 46))

# Map-Reduce example B1

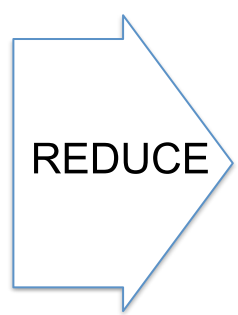
## Find largest integer (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **the largest integer** as output
- The large file of integers cannot fit in the memory of node

**Reduce task:**

Single Reduce task is needed to pick the largest integer

(1, (10, 61, 46))




(1, 61) => 61

# Map-Reduce example B2

## Get unique integers (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **a set of unique integers** as output
- The large file of integers cannot fit in the memory of node

**Map task:**

Each Map task gets a chunk of the file of integers and processes it …

**Reduce task:**

?

# Map-Reduce example B2

## Get unique integers (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **a set of unique integers** as output
- The large file of integers cannot fit in the memory of node

**Map task:**

**Emit (integer, 1)** once only for each unique integer for each chunk

e.g., maintain an array or a map to track whether have seen/emitted that integer before

# Map-Reduce example B2

## Get unique integers (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **a set of unique integers** as output
- The large file of integers cannot fit in the memory of node

**Shuffle step:**

Shuffle step will group together all values for the same integer: (integer, (1, 1, 1, 1, …))

Same integer might **appear in multiple chunks** from Map tasks and **each integer key** will only go to one Reduce task

# Map-Reduce example B2

## Get unique integers (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **a set of unique integers** as output
- The large file of integers cannot fit in the memory of node

**Reduce task:**

Each Reduce task eliminates duplicates (also ignore list of 1's) for each integer key and emit (integer)

Combine the output from multiple reduce tasks (not required to be in any order)

# Map-Reduce example B3

## Count the number of unique integers (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **the count of the number of distinct integers** as output
- The large file of integers cannot fit in the memory of node

**Map task:**

Each Map task gets a chunk of the file of integers and processes it …

**Reduce task:**

?

# Map-Reduce example B3

## Count the number of unique integers (Exercise 2.3.1 in book)

- **Two stages needed**
  - First phase: eliminate duplicates in large input file
    - Map task 1: just emit (integer, 1) for unique integers

      e.g., Mapper 1 (chunk1, (7, 7, 8, 8, …)) => (7, 1), (8,1), …

      Mapper 2 (chunk1, (7, 7, 9, …)) => (7, 1), (9,1), …
    - Reduce task 1: Eliminates duplicates (across chunks)

      e.g., (7, (1, 1)) => (7, 1) or just 7

      (8, (1)) => (8, 1) or just 8

# Map-Reduce example B3

## Count the number of unique integers (Exercise 2.3.1 in book)

- **Two stages needed**
  - Second phase: count unique numbers
    - Map task 2: each map task gets some unique integers from the previous Reduce phase as the input and output the key-value pair like **(some key, count)** (e.g., let key be 1)

      e.g., (7, 8, 9, 12, 13) => (1, 5)

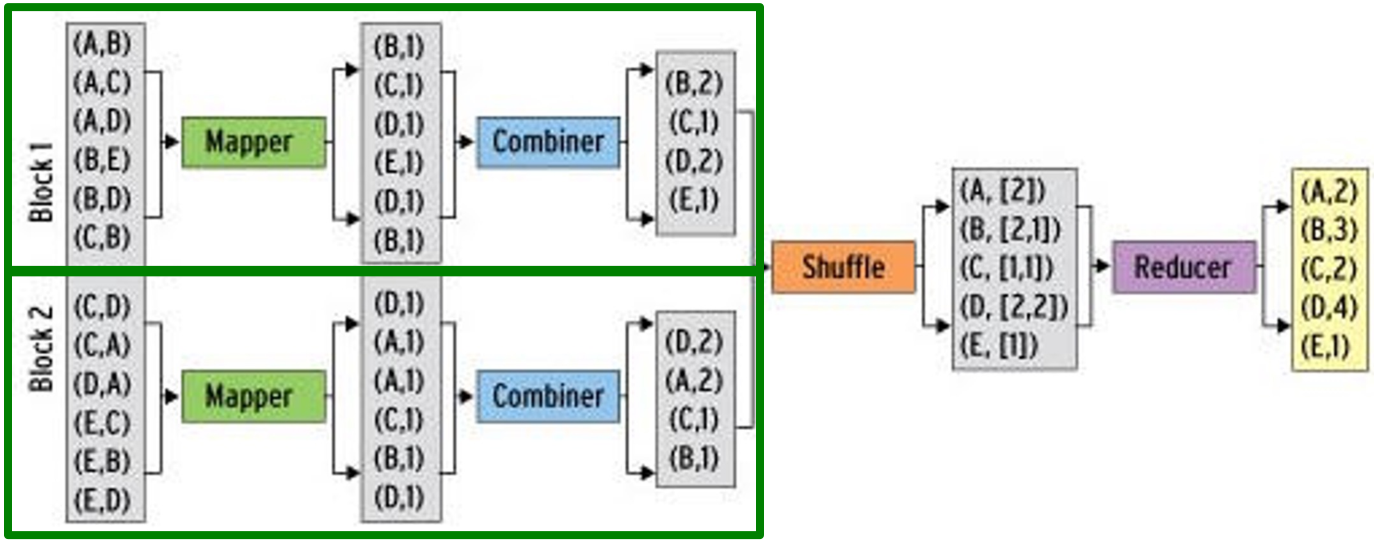
    - Reduce task 2: A single Reducer, sums all counts from map tasks and produces overall count

# Combiners: (“helpers” of map)

- Often a Map task will produce many pairs of the form $(k, v_1)$, $(k, v_2)$, … for the same key k
  - e.g., popular words “the” would appear thousands of times in the word count example, generate thousands of (“the”, 1) tuples in each mapper, and ship to reducers
  - Instead of producing many pairs (“the”, 1), (“the”, 1), … can sum the n occurrences of “the” (each word) and **emit (w, n) in Map task before shipping to reducers**
  - Now each node **only sends a single value** for each word
- Can save network time by **pre-aggregating values in the mapper**

# Refinement: Combiners

- **Combiner** combines the values of all keys of a single mapper (single node)




Much less data needs to be copied and shuffled!

# Combiners (“pre”-reducers)

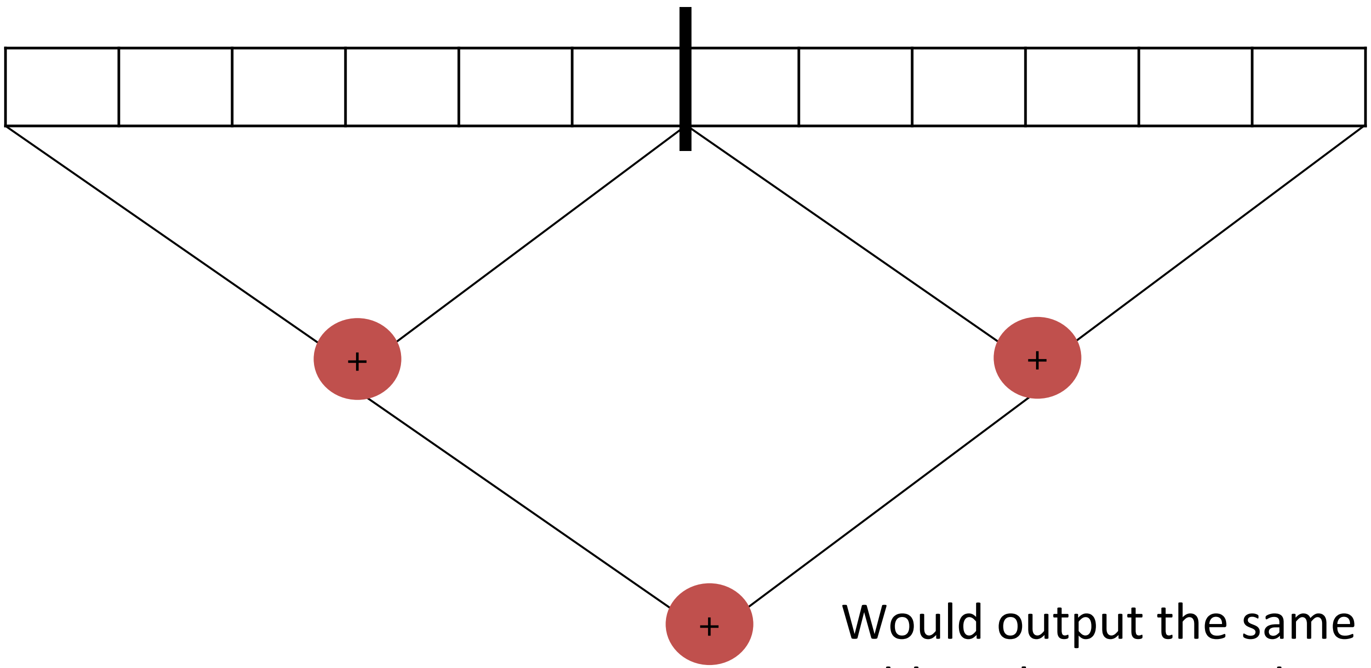
- Combiners run after the Mappers and before the Reducers
- Combiner receives all data emitted by the Mapper as input
- The output of the Combiners is then sent to the Reducers
- Usage of the Combiner is optional **(When to use ?)**
- Located at map: If combiner is suitable for the job, the instances of the Combiner are run on every node that has run map tasks (Why?)
- Perform as reducer: The Combiner is **a "mini-reduce" process** (usually the same as the reduce function)

# Combiners (when is suitable?)

- If a Reduce function is both **commutative** and **associative**, then it can be used as a Combiner
  - e.g., sum (add up all the input values)

    Commutative: a + b = b + a

    Associative: (a + b) + c = a + (b + c)




Would output the same result after adding the intermediate sum

# Combiners (when is suitable?)

- If a Reduce function is both **commutative** and **associative**, then it can be used as a Combiner
  - e.g., Sum (add up all the input values)

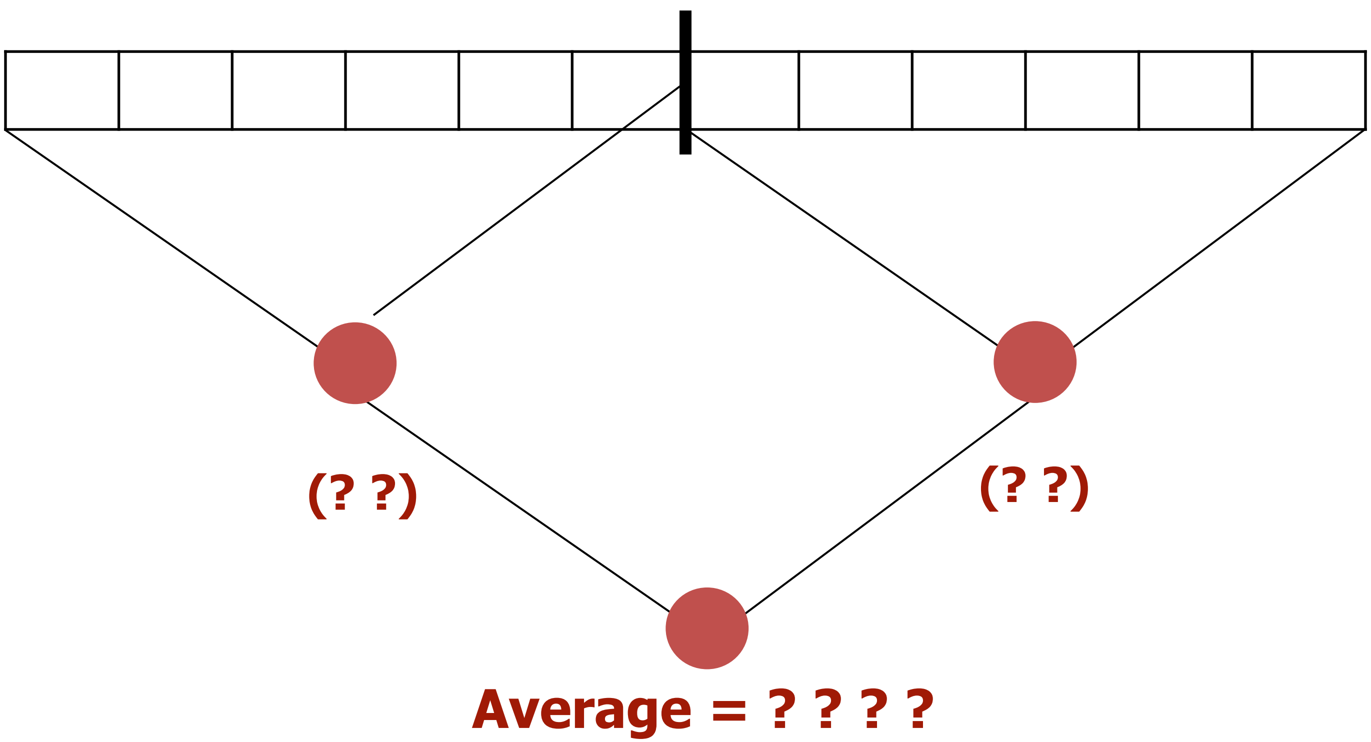
    Commutative: $a + b = b + a$

    Associative: $(a + b) + c = a + (b + c)$

- If the Reducer cannot be used directly as a Combiner because of commutativity or associativity
  - <u>You may still be able to write a third class to use as a Combiner</u>
  - e.g., Average

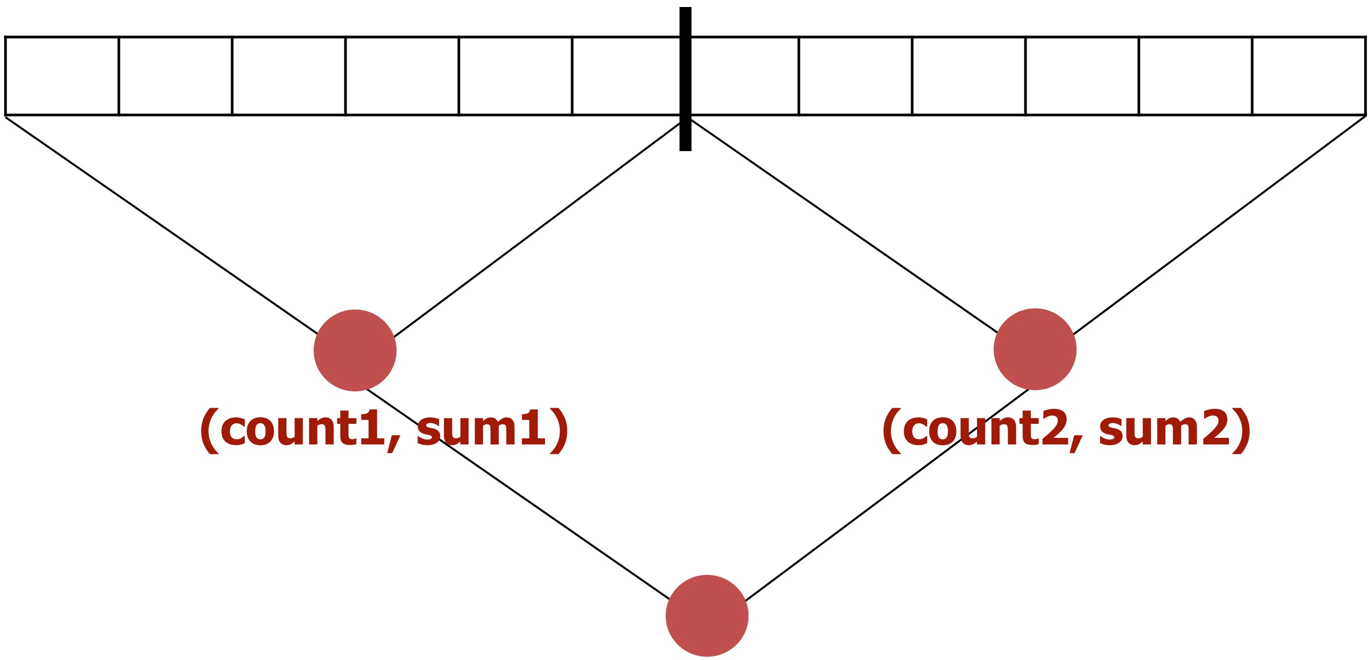
# Combiners

- If the Reducer cannot be used directly as a Combiner because of commutativity or associativity
  - <u>You may still be able to write a third class to use as a Combiner</u>
  - e.g., Average (itself is not commutative and associative)



# Combiners

- If the Reducer cannot be used directly as a Combiner because of commutativity or associativity
  - <u>You may still be able to write a third class to use as a Combiner</u>
  - e.g., Average (itself is not commutative and associative)




**Average = (sum1 + sum2) / (count1 + count2)**

# Map-Reduce Example (with combiner)
## Compute average (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **the average of all the integers** as output
- The large file of integers cannot fit in the memory of node

**Map task:**

Each Map task gets a chunk of the file of integers and processes it …

**Reduce task:**

?

# Map-Reduce example (with combiner)
## Compute average (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **the average of all the integers** as output
- The large file of integers cannot fit in the memory of node

**Map task:**

Map task produces **(key, (number of integers, sum of integers))** for each chunk as output

- Key could be set to the same value for all mappers, e.g., 1

- Value is the (number, sum) tuple

# Map-Reduce example (with combiner)
## Compute average (Exercise 2.3.1 in book)

- Design MapReduce algorithms to take a very large file of integers and produce **the average of all the integers** as output
- The large file of integers cannot fit in the memory of node

**Reduce task:**

A single Reduce task **sums all the sums of integers** and **sums number of integers**, calculate average; emit (average, 1)

Since each Map task summarizes a large chunk of data by **a single (key, (number, sum)) key-value pair**, this should be able to use **a single Reducer** even with thousands of Map tasks

# Map-Reduce Example (with combiner)
## Word length histogram

Goal: How many "big" (10+ letters), "medium" (5-9 letters), "small" (2-4 letters), and "tiny" (1 letter) words are used?

Abridged Declaration of Independence

A Declaration By the Representatives of the United States of America, in General Congress Assembled.
When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change.
We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying it's foundation on such principles and organizing it's power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood

# Map-Reduce Example (with combiner)
## Word length histogram

"Big" (Yellow) = 10+ letters

"Medium" (Red) = 5..9 letters

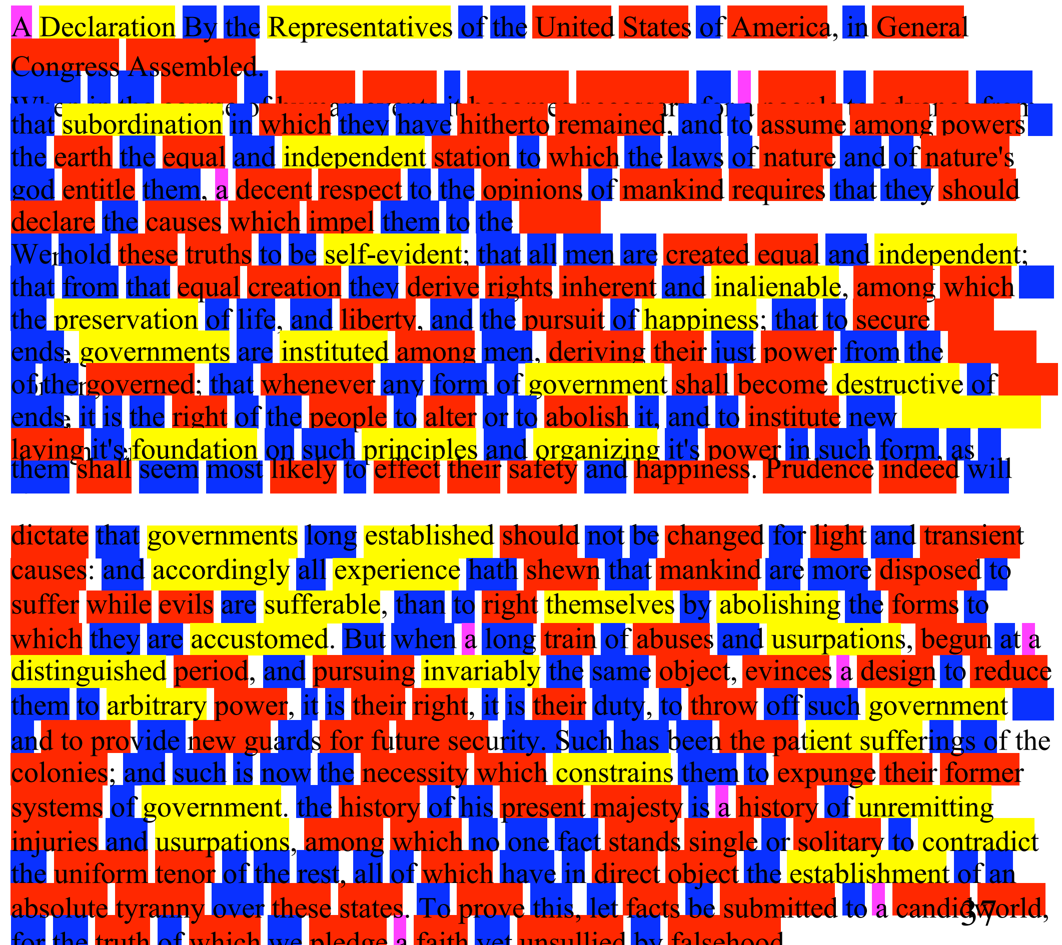
"Small" (Blue) = 2..4 letters

"Tiny" (Pink) = 1 letter

Map task: ?

Reduce task: ?

Abridged Declaration of Independence



# Map-Reduce Example (with combiner)

## Word length histogram

Abridged Declaration of Independence

Chunk 1

A Declaration By the Representatives of the United States of America, in General Congress Assembled.

When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change.

We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying it's foundation on such principles and organizing it's power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will

Chunk 2

dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.

**Split the document into multiple chunks and process each chunk on different nodes**

# Map-Reduce Example (with combiner)

## Word length histogram

Abridged Declaration of Independence

Map Task 1
(204 words)

A Declaration By the Representatives of the United States of America, in General Congress Assembled.
When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the
We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying it's foundation on such principles and organizing it's power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will

(key, value)

(yellow, 17)
(red, 77)
(blue, 107)
(pink, 3)

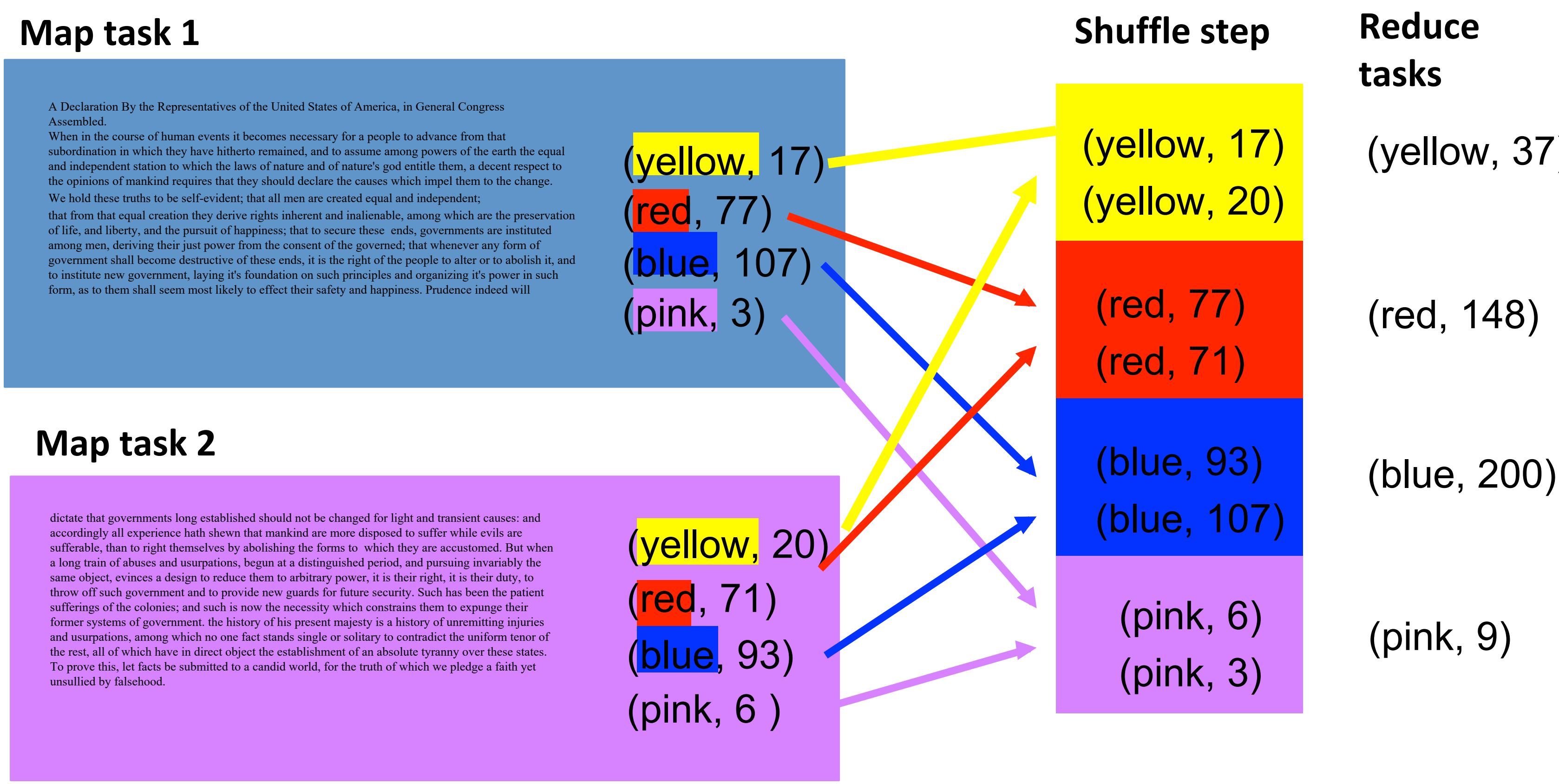
Map Task 2
(190 words)

dictate that governments long established should not be changed for light and causes and accordingly all experience hath shewn that mankind are more disposed suffer while evils are sufferable, than to right themselves by abolishing the forms which they are accustomed. But when a long train of abuses and usurpations, begun at distinguished period, and pursuing invariably the same object, evinces a design to them to arbitrary power, it is their right, it is their duty, to throw off such government to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to the uniform tenor of the rest, all of which have in direct object the establishment of absolute tyranny over these states. To prove this, let facts be submitted to a candid for the truth of which we pledge a faith yet unsullied by falsehood.

(yellow, 20)
(red, 71)
(blue, 93)
(pink, 6 )

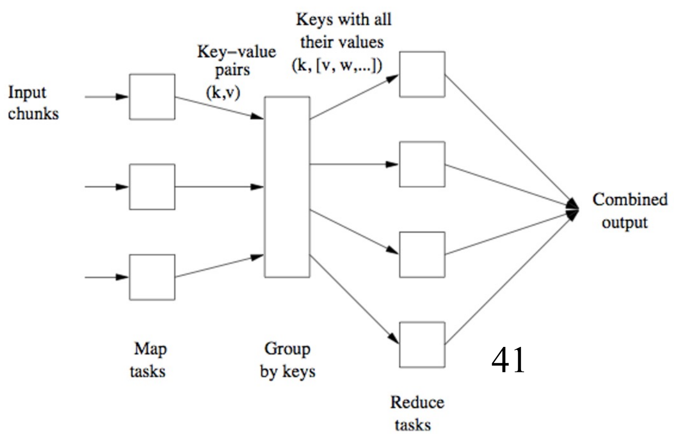
# Map-Reduce Example (with combiner)

## Word length histogram



# Map-Reduce: Summary

- **Map tasks**
  - Input is in "key-value" format, e.g., key = file locations, value = text
  - Map code written by the user
  - Processes chunks and produces sequence of key-value pairs, Notice: the "key" is not in usual sense, do not have to be unique
- **Group by Key/Shuffle**
  - Collects key-value pairs from each Map task
  - Values associated with each key are formed into a list of values
  - All key-value pairs with same key go to same Reduce task
- **Reduce task**
  - Reduce code written by user
  - Produces output key-value pairs

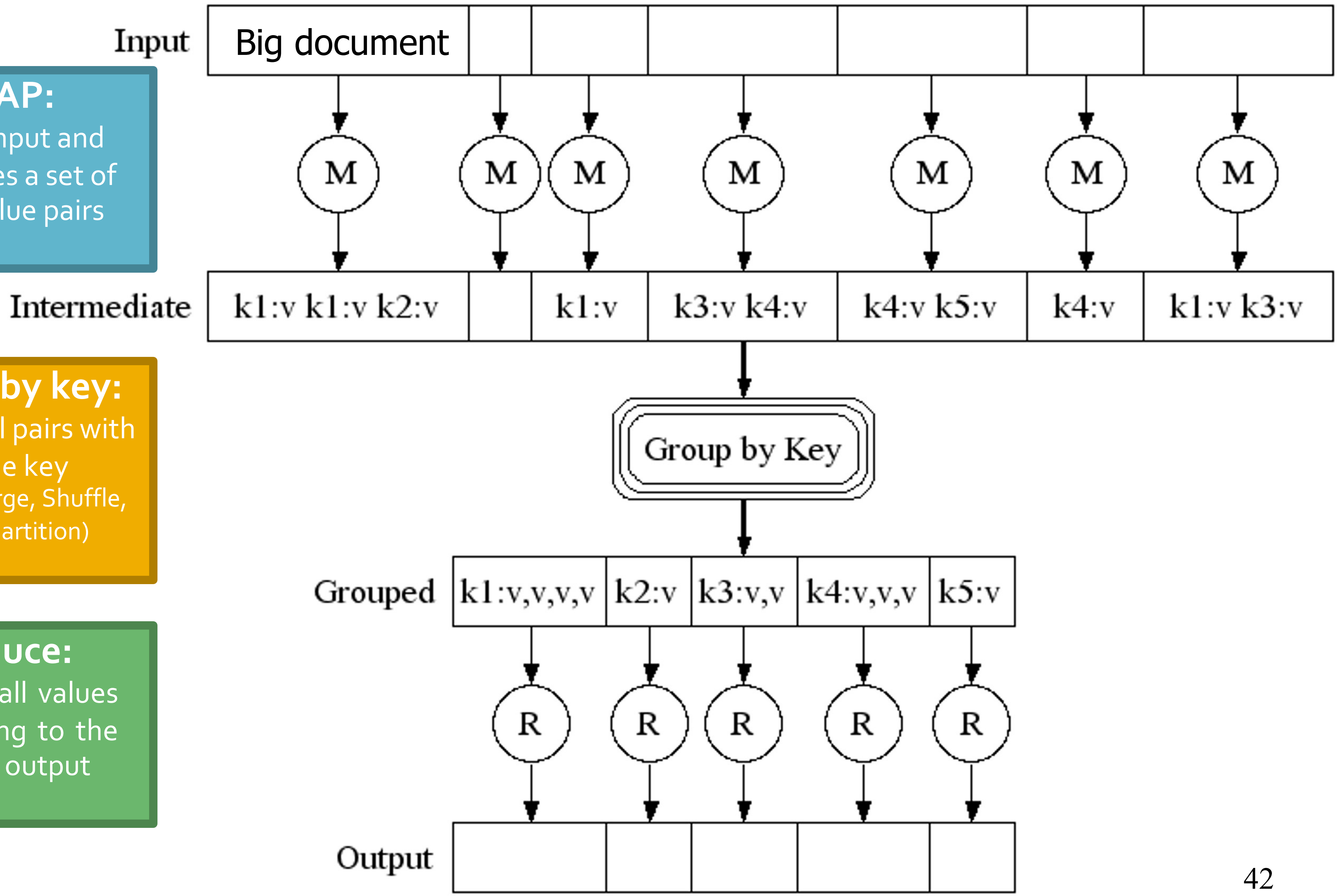


# Map-Reduce: A Diagram

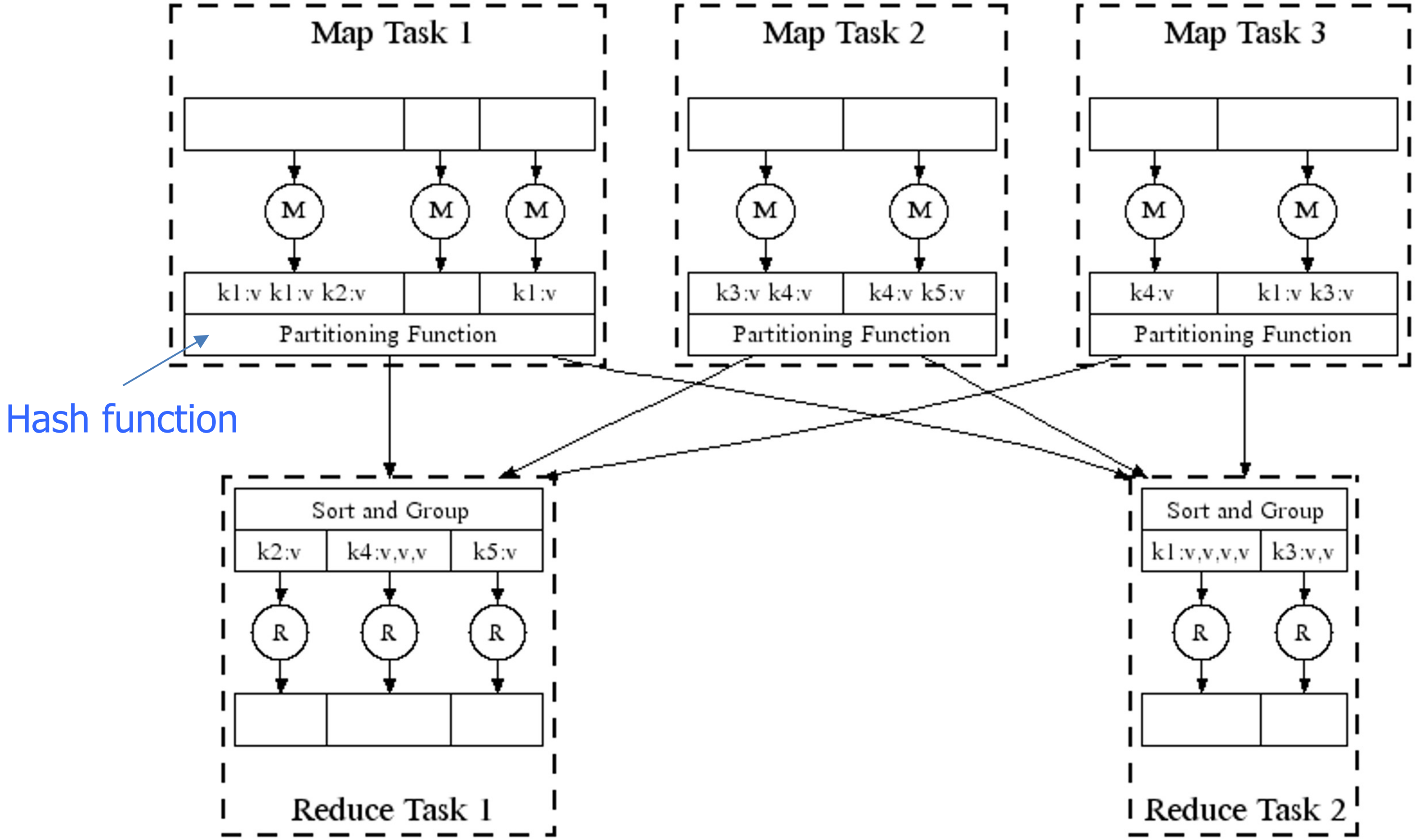
**MAP:**
Read input and produces a set of key-value pairs

**Group by key:**
Collect all pairs with same key
(Hash merge, Shuffle, Sort, Partition)

**Reduce:**
Collect all values belonging to the key and output



# Map-Reduce: In Parallel

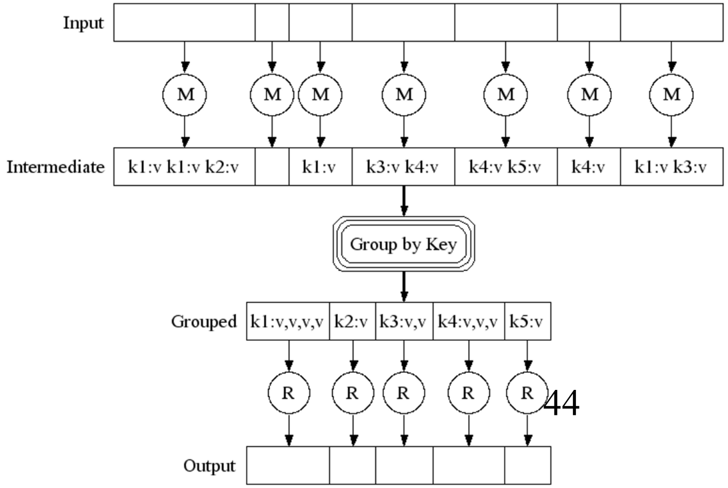



All phases are distributed with many tasks doing the work

# Map-Reduce: Environment

**MapReduce environment takes care of:**

- **Partitioning** the input data
- **Scheduling** the program's execution across a set of nodes
- Performing the **group by key** step
- Handling machine **failures**
- Managing required inter-machine **communication**



# Data Flow

- Input and final output are stored **on a distributed file system (DFS)**
  - Scheduler tries to schedule map tasks "close" to physical storage location of input data

- Intermediate results are stored **on local file system of Map workers**

  e.g., output of the map step

- Output is often input to another Map-Reduce task

# Coordination: Master

- Master node takes care of coordination:
  - **Task status:** idle, in-progress, completed
  - **Idle tasks** get scheduled as workers become available
  - When a map task completes, it sends the **master** the **location** and **sizes** of its R intermediate files, one for each reducer (R = number of reducers)
  - **Master pushes this info to reducers**
- Master pings workers periodically to detect failures

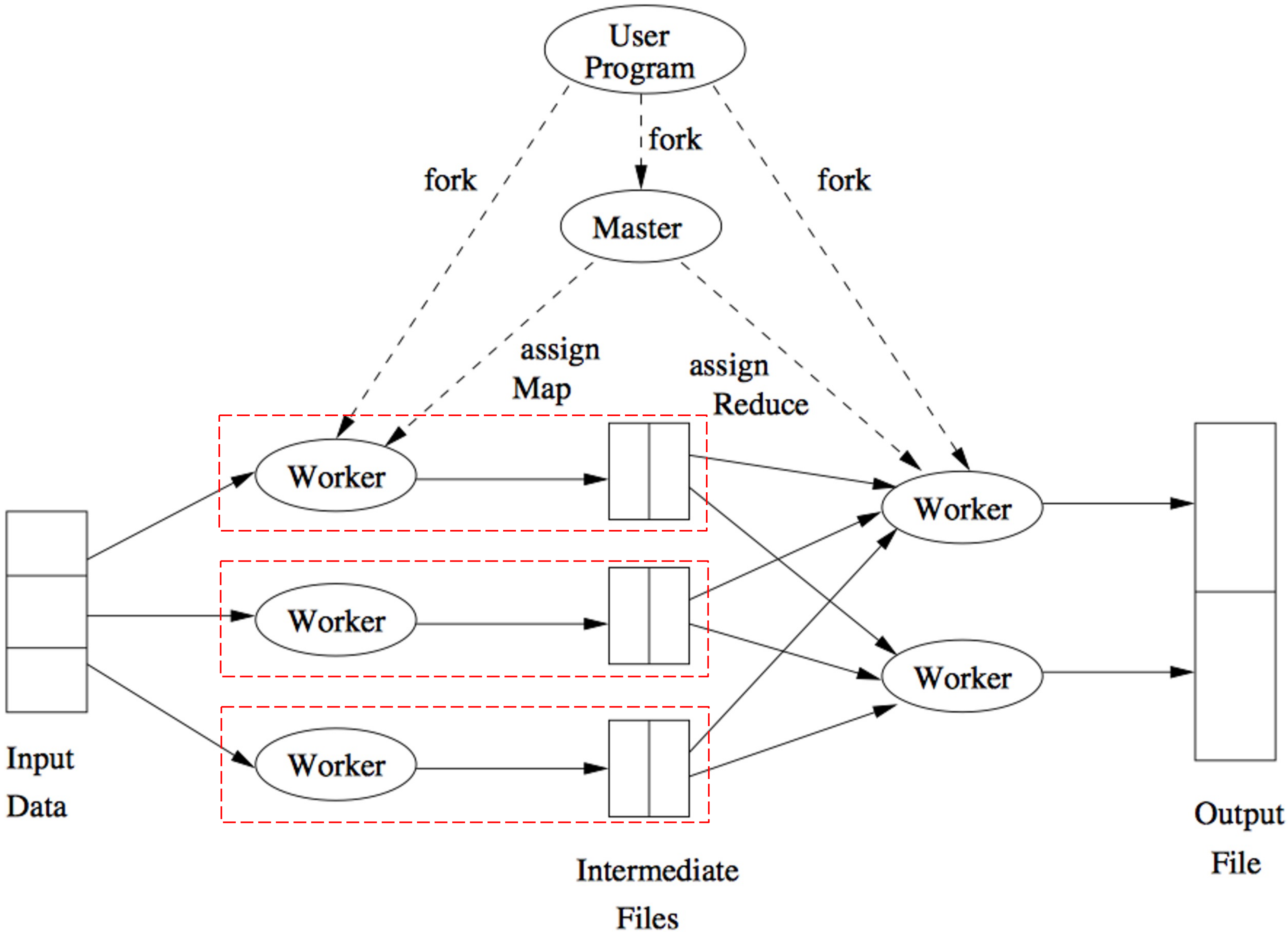



Figure 2.3: Overview of the execution of a MapReduce program

# Dealing with Failures

- **Map worker failure**
  - Map tasks **completed or in-process** at worker are reset to idle (why?)
  - Idle tasks eventually rescheduled on other worker(s)

# Dealing with Failures

- **Map worker failure**
  - Map tasks **completed or in-process** at worker are reset to idle (why?)
  - Idle tasks eventually rescheduled on other worker(s)

- **Reduce worker failure**
  - Only **in-process tasks** are reset to idle
  - Idle Reduce tasks restarted on other worker(s)

# Dealing with Failures

- **Map worker failure**
  - Map tasks **completed or in-process** at worker are reset to idle (why?)
  - Idle tasks eventually rescheduled on other worker(s)
- **Reduce worker failure**
  - Only **in-process tasks** are reset to idle
  - Idle Reduce tasks restarted on other worker(s)
- **Master failure**
  - Map-reduce task is aborted and client is notified

# How Many Map and Reduce Jobs?

- ***M* map tasks, *R* reduce tasks**

# How Many Map and Reduce Jobs?

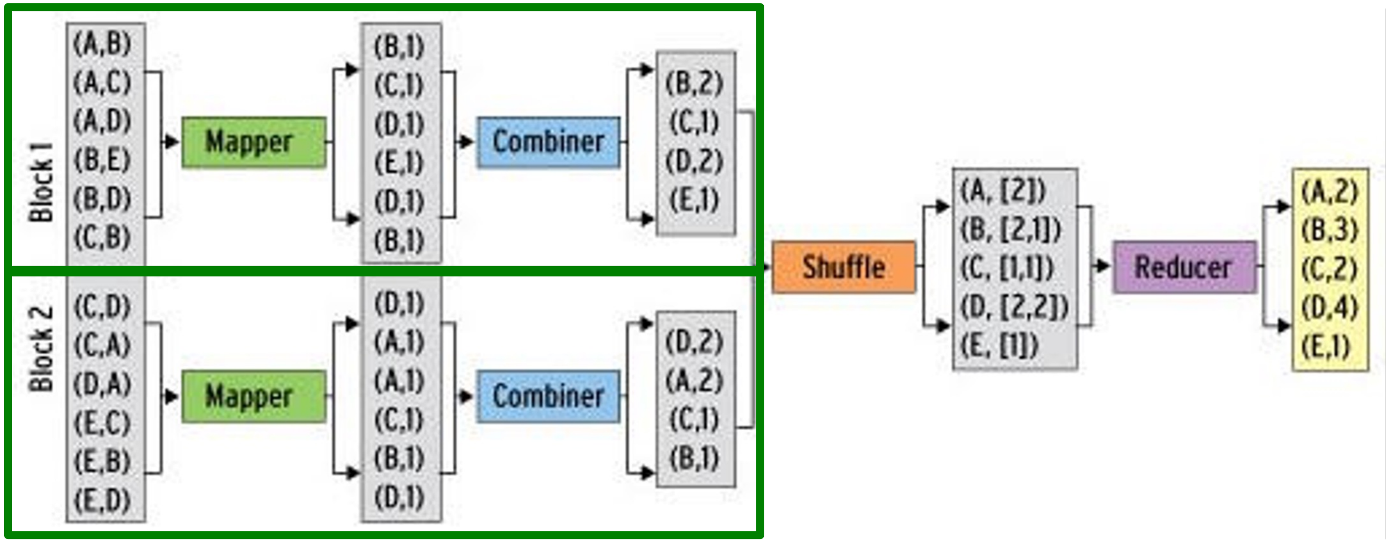
- ***M* map tasks, *R* reduce tasks**
- **Rule of a thumb:**
    - Make *M* **much larger** than the number of nodes in the cluster
    - One DFS chunk per map task is common
    - Improves dynamic load balancing and speeds up recovery from worker failures

# How Many Map and Reduce Jobs?

- ***M* map tasks, *R* reduce tasks**
- **Rule of a thumb:**
  - Make *M* **much larger** than the number of nodes in the cluster
  - One DFS chunk per map task is common
  - Improves dynamic load balancing and speeds up recovery from worker failures
- **Usually *R* is smaller than *M***
  - Output is spread across *R* files
  - Google example: Often use 200,000 map tasks, 5000 reduce tasks on 2000 machines

# Refinement: Combiners

- **Combiner** combines the values of all keys of a single mapper (single node)




Much less data needs to be copied and shuffled!

Works if reduce function is commutative and associative

# Refinement: Partition Function

- **Control how keys get partitioned (which key goes where?)**
  - Reduce needs to ensure that records with the same intermediate key end up at the same worker

- System uses a default partition function:
  - **hash(key) mod *R***

- Sometimes useful to override the hash function:
  - E.g., **hash(hostname(URL)) mod *R*** ensures URLs from the same host to end up in the same output file

# Implementations

- **Google's MapReduce**
  - Not available outside Google

- **Hadoop**
  - Open-source implementation in Java
  - Uses HDFS for stable storage
  - Download: http://hadoop.apache.org/releases.html

- **Spark** (at your hand ☺️ )

# More Examples: Relational Join

**Employee**

| Name | SSN |
|---|---|
| Sue | 999999999 |
| Tony | 777777777 |

**Assigned Departments**

| EmpSSN | DepName |
|---|---|
| 999999999 | Accounts |
| 777777777 | Sales |
| 777777777 | Marketing |

Emplyee $\bowtie$ Assigned Departments

| Name | SSN | EmpSSN | DepName |
|---|---|---|---|
| Sue | 999999999 | 999999999 | Accounts |
| Tony | 777777777 | 777777777 | Sales |
| Tony | 777777777 | 777777777 | Marketing |

# Example: Relational Join

- Map Task: Emit (key, value) pair
  - **Key = "the key used for the join"**
  - **Value is a tuple** with all fields from table (including the table name)

**Employee**

| Name | SSN |
|---|---|
| Sue | 999999999 |
| Tony | 777777777 |

key=999999999, value=(Employee, Sue, 999999999)
key=777777777, value=(Employee, Tony, 777777777)

**Assigned Departments**

| EmpSSN | DepName |
|---|---|
| 999999999 | Accounts |
| 777777777 | Sales |
| 777777777 | Marketing |

key=999999999, value=(Department, 999999999, Accounts)
key=777777777, value=(Department, 777777777, Sales)
key=777777777, value=(Department, 777777777, Marketing)

# Example: Relational Join

- **Group by Key:** groups together all values (tuples) associated with each key
- **Reduce task:** emit joined values (without table names)

key=999999999, values=[(Employee, Sue, 999999999),
(Department, 999999999, Accounts)]

Sue, 999999999, 999999999, Accounts

key=777777777, values=[(Employee, Tony, 777777777),
(Department, 777777777, Sales),
(Department, 777777777, Marketing)]

Tony, 777777777, 777777777, Sales
Tony, 777777777, 777777777, Marketing

# More Example: Distributed Sort

- Goal: Sort a very large list of (firstName, lastName) pairs by lastName followed by firstName

- Map task:

- Reduce task:

# Example: Distributed Sort

- Map task
  - Emit (lastName, firstName)
- Group by keys:
  - Group together entries with the same last name
  - **Divide into non-overlapping alphabetical ranges** (sorting)
  - Keys are sorted in alphabetical order
- Reduce task
  - Processes one key at a time
  - For each (lastName, list(firstName)), emit (lastName, firstName) in alphabetical order **(sorting)**
  - Merge output from all Reduce tasks (e.g., write)

# Example: Matrix Multiplication

- Assume two matrices A and B, and AB = C
- $A_{ij}$ is the element in **row *i*** and **column *j*** of matrix A
  - Similarly for B and C
- **$C_{ik} = \sum_j A_{ij} \times B_{jk}$**
  - $C_{ik}$ depends on
    the $i^{th}$ row of A // that is $A_{ij}$ for all j
    the $k^{th}$ column of B // that is $B_{jk}$ for all j

$$\underset{A}{\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix}} \underset{B}{\begin{bmatrix} 1 & 3 \\ 0 & 1 \\ 5 & 2 \end{bmatrix}} = \underset{C}{\begin{bmatrix} 11 & 10 \\ 9 & 14 \end{bmatrix}}$$

e.g., $C_{11} = 1\times1 + 3\times0 + 2\times5 = 11$

# Matrix Multiplication
# Map-Reduce (One phase)

C = A X B

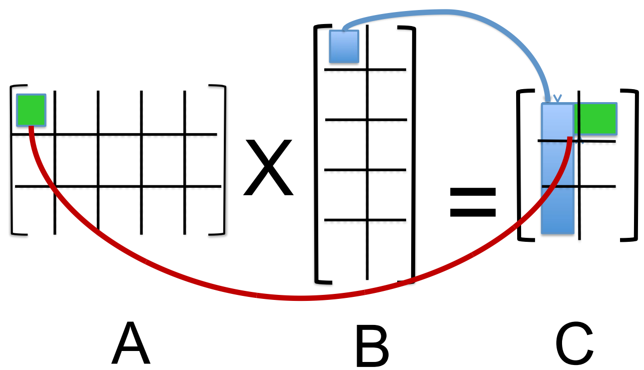
A has dimensions L x M

B has dimensions M x N

C has dimensions L x N

Matrix Multiplication:

$C[i, k] = SUM_j (A[i, j] \times B[j, k])$




**Map task:**

**Reduce task:**

# Matrix Multiplication Map-Reduce (One phase)

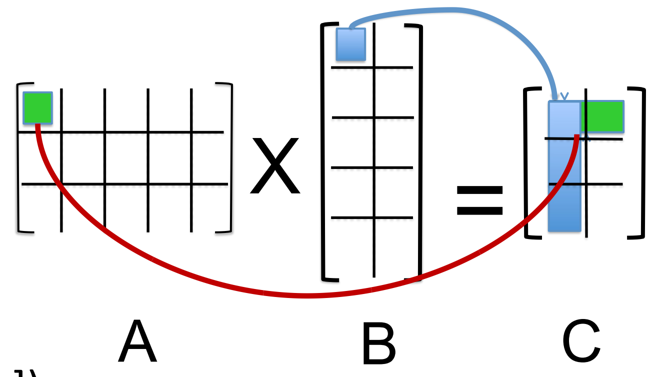
$C = A \times B$

A has dimensions L x M

B has dimensions M x N

C has dimensions L x N

Matrix Multiplication: $C[i, k] = SUM_j (A[i, j] \times B[j, k])$




**Map task:**

for each element (i,j) of A, emit **((i,k), A[i,j])** for **k in 1..N**

(i,k) is the "place" in C where A[i,j] will be needed

for each element (j,k) of B, emit **((i,k), B[j,k])** for **i in 1..L**

(i,k) is the "place" in C where B[j,k] will be needed

# Matrix Multiplication Map-Reduce (One phase)

C = A X B

A has dimensions L x M

B has dimensions M x N

C has dimensions L x N

Matrix Multiplication: $C[i, k] = SUM_j (A[i, j] \times B[j, k])$

$$\underset{A}{\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix}} \underset{B}{\begin{bmatrix} 1 & 3 \\ 0 & 1 \\ 5 & 2 \end{bmatrix}} = \underset{C}{\begin{bmatrix} 11 & 10 \\ 9 & 14 \end{bmatrix}}$$

**Map task:**

$C_{ik}$

for each element (i,j) of A, emit **((i,k), A[i,j])** for **k in 1..N** (*i* row)

e.g., For A[1, 1] emit ((1, 1), 1), ((1, 2), 1) // $A_{11}$=1 is needed for $C_{11}$, $C_{12}$

For A[1, 2] emit ((1, 1), 3), ((1, 2), 3) // $A_{12}$=3 is needed for $C_{11}$, $C_{12}$

For A[2, 1] emit ((2, 1), 4), ((2, 2), 4) // $A_{21}$=4 is needed for $C_{21}$, $C_{22}$

for each element (j,k) of B, emit **((i,k), B[j,k])** for **i in 1..L** (*k* column)

e.g., For B[1, 1] emit ((1, 1), 1), ((2, 1), 1) // $B_{11}$=1 is needed for $C_{11}$, $C_{21}$

For B[2, 1] emit ((1, 1), 0), ((2, 1), 0) // $B_{21}$=0 is needed for $C_{11}$, $C_{21}$

For B[1, 2] emit ((1, 2), 3), ((2, 2), 3) // $B_{12}$=3 is needed for $C_{12}$, $C_{22}$

# Matrix Multiplication
# Map-Reduce (One phase)

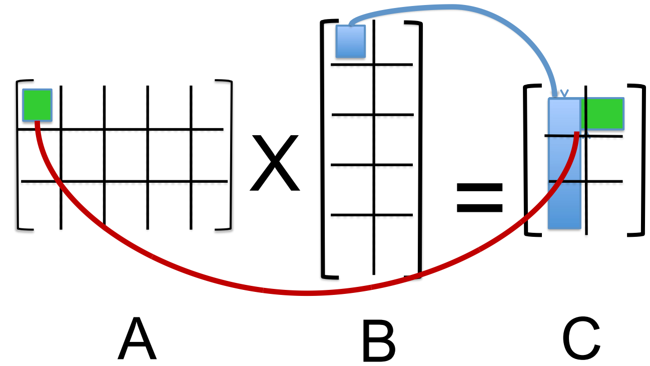
C = A X B

A has dimensions L x M

B has dimensions M x N

C has dimensions L x N

Matrix Multiplication: $C[i, k] = SUM_j (A[i, j] \times B[j, k])$




**Map task:**

$C_{ik}$

for each element (i,j) of A, emit **((i,k), A[i,j])** for **k in 1..N**

**Better: emit ((i,k), (‘A’, i, j, A[i,j])) for k in 1..N**

**Or just emit ((i,k), (‘A’, j, A[i,j])) for k in 1..N**

for each element (j,k) of B, emit **((i,k), B[j,k])** for **i in 1..L**

**Better: emit ((i,k), (‘B’, j, k, B[j,k])) for i in 1..L**

**Or just emit ((i,k), (‘B’, j, B[j,k])) for i in 1..L**

# Matrix Multiplication Map-Reduce (One phase)

C = A X B

A has dimensions L x M

B has dimensions M x N

C has dimensions L x N

Matrix Multiplication: $C[i, k] = SUM_j (A[i, j] \times B[j, k])$

$$\underset{A}{\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix}} \underset{B}{\begin{bmatrix} 1 & 3 \\ 0 & 1 \\ 5 & 2 \end{bmatrix}} = \underset{C}{\begin{bmatrix} 11 & 10 \\ 9 & 14 \end{bmatrix}}$$

**Map task:**

$C_{ik}$

for each element (i,j) of A, emit **((i,k), ('A', i, j, A[i,j]))** for k in 1..N

e.g., For A[1, 1] emit ((1, 1), ('A', 1, 1, 1)), ((1, 2), ('A', 1, 1, 1)) // for $C_{11}$, $C_{12}$

For A[1, 2] emit ((1, 1), ('A', 1, 2, 3)), ((1, 2), ('A', 1, 2, 3)) // for $C_{11}$, $C_{12}$

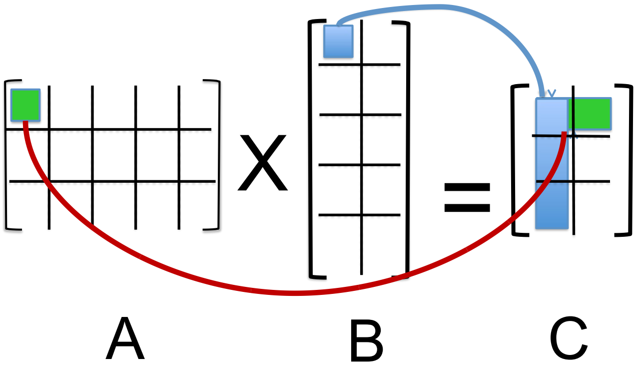
For A[2, 1] emit ((2, 1), ('A', 2, 1, 4)), ((2, 2), ('A', 2, 1, 4)) // for $C_{21}$, $C_{22}$

for each element (j,k) of B, emit **((i,k), ('B', j, k, B[j,k]))** for i in 1..L

e.g., For B[1, 1] emit ((1, 1), ('B', 1, 1, 1)), ((2, 1), ('B', 1, 1, 1)) // for $C_{11}$, $C_{21}$

For B[2, 1] emit ((1, 1), ('B', 2, 1, 0)), ((2, 1), ('B', 2, 1, 0)) // for $C_{11}$, $C_{12}$

For B[1, 2] emit ((1, 2), ('B', 1, 2, 3)), ((2, 2), ('B', 1, 2, 3))




**C[i,k] = $\text{Sum}_j$ (A[i,j] x B[j,k]), C is L x N**

In the map phase:

Cik

- for each element (i,j) of A, emit **((i,k), (‘A’, i, j, A[i,j]))** for k in 1..N
- for each element (j,k) of B, emit **((i,k), (‘B’, j, k, B[j,k]))** for i in 1..L

e.g.,

C[1,1] = A[1,1] * B[1,1] + **A[1,2]** * B[2,1] + A[1,3] * **B[3,1]** + A[1,4] * B[4,1] + A[1,5] * B[5,1]

C[1,2] = A[1,1] * B[1,2] + **A[1,2]** * B[2,2] + A[1,3] * B[3,2] + A[1,4] * B[4,2] + A[1,5] * B[5,2]

C[2,**1**] = A[2,1] * B[1,1] + A[2,2] * B[2,1] + A[2,3] * **B[3,1]** + A[2,4] * B[4,1] + A[2,5] * B[5,1]

C[3,**1**] = A[3,1] * B[1,1] + A[3,2] * B[2,1] + A[3,3] * **B[3,1]** + A[3,4] * B[4,1] + A[3,5] * B[5,1]

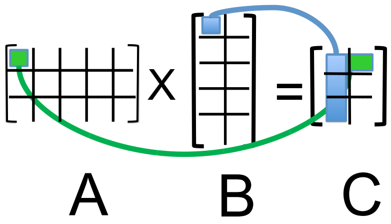
Map phase: For **A[1,2]**, emit ((1, k), (‘A’, 1, 2, A[1,2])) for k in 1..2

**emit ((1,1)(‘A’, 1, 2, A[1,2])) ((1,2)(‘A’, 1, 2, A[1,2]))**

For **B[3,1]**, emit ((i, 1), (‘B’, 3, 1, B[3,1])) for i in 1..3

**emit ((1,1), (‘B’, 3, 1,B[3,1])), ((2,1)(‘B’, 3, 1,B[3,1])), ((3,1)(‘B’, 3, 1,B[3,1]))**

# Matrix Multiplication
# Map-Reduce (One phase)




## A simple exercise:

Suppose the data of matrices are distributed among 5 workers:

W1 has A-row-1, W2 has A-row-2, W3 has A-row-3

W4 has B-column-1, W5 has B-column-2

Now who sends out what?

For A[1, 1], W? emits ((1, 1), ('A', 1, 1, 1)), ((1, 2), ('A', 1, 1, 1))

For A[1, 2], W? emits ((1, 1), ('A', 1, 2, 3)), ((1, 2), ('A', 1, 2, 3))

For A[2, 1], W? emits ((2, 1), ('A', 2, 1, 4)), ((2, 2), ('A', 2, 1, 4))

For B[1, 1], W? emits ((1, 1), ('B', 1, 1, 1)), ((2, 1), ('B', 1, 1, 1))

For B[2, 1], W? emits ((1, 1), ('B', 2, 1, 0)), ((2, 1), ('B', 2, 1, 0))

For B[1, 2], W? emits ((1, 2), ('B', 1, 2, 3)), ((2, 2), ('B', 1, 2, 3))

# Matrix Multiplication Map-Reduce (One phase)

C = A X B

A has dimensions L x M

B has dimensions M x N

C has dimensions L x N

Matrix Multiplication: $C[i, k] = SUM_j (A[i, j] \times B[j, k])$

$$\underset{A}{\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix}} \underset{B}{\begin{bmatrix} 1 & 3 \\ 0 & 1 \\ 5 & 2 \end{bmatrix}} = \underset{C}{\begin{bmatrix} 11 & 10 \\ 9 & 14 \end{bmatrix}}$$

**Reduce task:**

key = (i,k)

value = $Sum_j$ (A[i,j] x B[j,k]) // this step does both Sum and Multiply
// We may break this step into more steps

# Matrix Multiplication
# Map-Reduce (Two Phases)

## Main Ideas:

Phase 1: <u>Multiply</u> the appropriate values in 1st MapReduce phase
Phase 2: <u>Sum up</u> in 2nd MapReduce phase

Try this at home!

# Matrix Multiplication
# Map-Reduce (Two phase)

**Idea: 1, Multiply the appropriate values in 1st MapReduce phase**
**2, Sum up in 2nd MapReduce phase**

**C[i,k] = Sum$_j$ (A[i,j]** x **B[j,k]), C is L** x **N**

e.g.,

C[1,1] = A[**1**,1] * B[1,**1**] + A[**1**,2] * B[2,**1**] + A[**1**,3] * B[3,**1**] + A[**1**,4] * B[4,**1**] + A[**1**,5] * B[5,**1**]

C[1,2] = A[**1**,1] * B[1,**2**] + A[**1**,2] * B[2,**2**] + A[**1**,3] * B[3,**2**] + A[**1**,4] * B[4,**2**] + A[**1**,5] * B[5,**2**]

1st Map Task:

For each matrix element **A[i,j] : emit( j , ('A', i, A[i,j]))**
For each matrix element **B[j,k] : emit( j , ('B', k, B[j,k]))**

e.g., (all will go to same reducer with key 2):

For A[1,2]: emit (2, ('A', 1, A[1,2]))
For B[2,1]: emit (2, ('B', 1, B[2,1]))
For B[2,2]: emit (2, ('B', 2, B[2,2]))

# Matrix Multiplication
# Map-Reduce (Two phase)

1st Reduce Task: (does the multiplication)

- For each key j, produce all possible products
- For each value of (i,k) which comes from A and B,
  i.e., (‘A’, i, A[i, j]) and (‘B’, k, B[j, k]): **emit ((i,k), (A[i, j] * B[j, k]))**

e.g., (from map task)
    For A[1,2]: emit (2, (‘A’, 1, A[1,2]))
    For B[2,1]: emit (2, (‘B’, 1, B[2,1]))
    For B[2,2]: emit (2, (‘B’, 2, B[2,2]))

**Reduce task for key j: (j=2)**
    **emit ((1,1), A[1,2] * B[2,1])**
    **emit ((1,2), A[1,2] * B[2,2])**

# Matrix Multiplication Map-Reduce (Two phase)

2$^{nd}$ Map Task:

- The input would be the (key, value) from 1$^{st}$ Reduce task
- Let the pair of (( (i,k), (A[i, j] * B[j, k])) pass through

2$^{nd}$ Reduce Task: (does the summation)

- For each (i,k), add up the values, **emit ((i,k), SUM(values))**

e.g., C[1,1] = A[1,1]*B[1,1] + A[1,2]*B[2,1] + A[1,3]*B[3,1] + A[1,4]*B[4,1] + A[1,5]*B[5,1]

(Note: every term has key (1,1), will go to same reducer)

# General Characteristic of Good Problem for Map-Reduce

Data set is truly "big"

- Terabytes, not tens of gigabytes
- Hadoop/MapReduce designed for terabyte/petabyte scale computation
- Most real-world problems process less than 100 GB of input
  - Microsoft, Yahoo: median job under 14 GB
  - Facebook: 90% of jobs under 100 GB

Source: "To Hadoop or Not to Hadoop" by Anand Krishnaswamy 8/13/2013

# General Characteristic of Good Problem for Map-Reduce

Don't need fast response time

- When submitting jobs, Hadoop latency can be 1 min
- Not well-suited for problems that require faster response time
  - online purchases, transaction processing
- A good pre-computation engine
  - E.g., pre-compute related items for every item in inventory

Source: "To Hadoop or Not to Hadoop" by Anand Krishnaswamy 8/13/2013

# General Characteristic of Good Problem for Map-Reduce

- Good for applications that work in **batch mode**
- Runs over entire data set
  - Takes time to initiate, run;
  - Shuffle step can be time-consuming;
- Does not provide good support for random access to datasets
  - Extensions: Hive, Dremel, Shark, Amplab

Source: "To Hadoop or Not to Hadoop" by Anand Krishnaswamy 8/13/2013

# General Characteristic of Good Problem for Map-Reduce

- Best suited for data that can be expressed as key-value pairs without losing context, dependencies
  - Graph data is hard to process using Map-Reduce
    - Implicit relationships: edges, sub-trees, child/parent relationships, weights, etc.
  - Graph algorithms need information about the entire graph for each iteration
    - Hard to break into independent chunks for Map tasks
  - Alternatives: Google's Pregel, Apache Giraph

Source: "To Hadoop or Not to Hadoop" by Anand Krishnaswamy 8/13/2013

# General Characteristic of Good Problem for Map-Reduce

Other problems/data NOT suited for MapReduce

- Tasks that need results of intermediate steps to compute results of current step
  - Interdependencies among tasks
  - Map tasks must be independent
- Some machine learning algorithms
  - Gradient-based learning

Source: "To Hadoop or Not to Hadoop" by Anand Krishnaswamy 8/13/2013

# General Characteristic of Good Problem for Map-Reduce

**Summary: Good candidates for Map-Reduce:**

- Jobs that process huge quantities of data and either summarize or transform the content
- Collected data has elements that can easily be captured with an identifier (key) and corresponding value

Source: "To Hadoop or Not to Hadoop" by Anand Krishnaswamy 8/13/2013