

# Map-Reduce

## Computational Model Examples

Mining of Massive Datasets  
Leskovec, Rajaraman, and Ullman  
Stanford University



# Programming Model: MapReduce

## Warm-up task:

- We have a huge text document
- Count the number of times each distinct word appears in the file
- Sample applications
  - Analyze web server logs to find popular URLs
  - Term statistics for search

# Task: Word Count

## Case 1:

- File too large for memory, but all <word, count> pairs fit in memory

# Word Count (2)

## Case 2:

- Even the <word,count> pairs don't fit in memory
- `words (doc.txt) | sort | uniq -c`
  - where **words** takes a file and outputs the words in it, one per a line
- Case 2 captures the essence of MapReduce
  - Great thing is that it is naturally parallelizable

# MapReduce: Overview

```
words (doc.txt) | sort | uniq -c
```

## ■ Map

- Scan input file record-at-a-time
- Extract something you care about from each record (keys)

## ■ Group by key

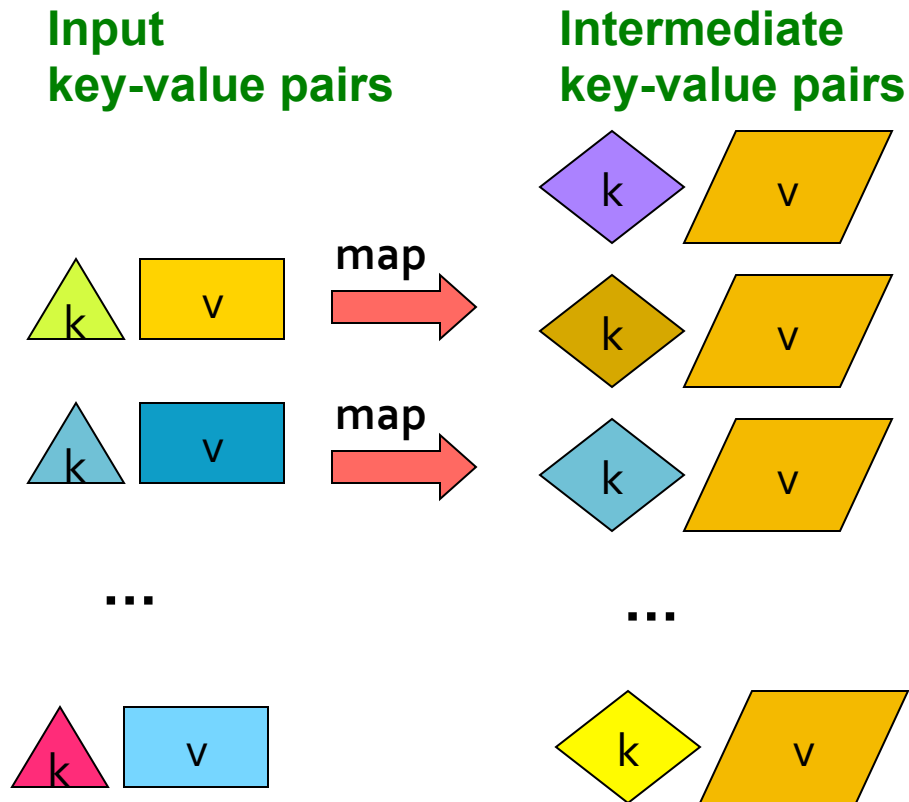
- Sort and Shuffle

## ■ Reduce

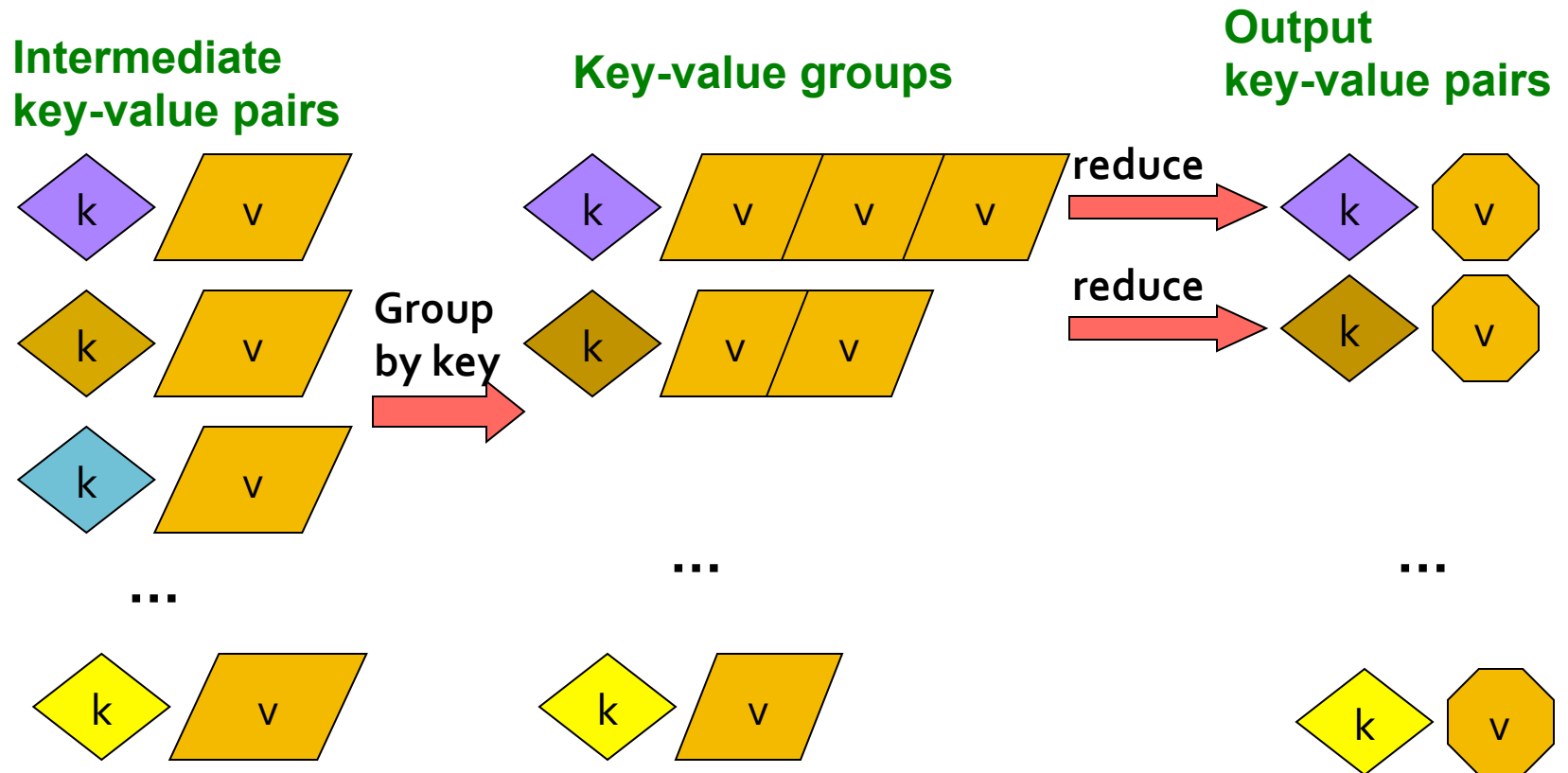
- Aggregate, summarize, filter or transform
- Write the result

Outline stays the same, **Map** and **Reduce**  
change to fit the problem

# MapReduce: The Map Step



# MapReduce: The Reduce Step



# More formally...

- **Input:** a set of key-value pairs
- Programmer specifies two methods:
  - **Map( $k, v$ )**  $\rightarrow \langle k', v' \rangle^*$ 
    - Takes a key-value pair and outputs a set of key-value pairs
    - There is one Map call for every  $(k, v)$  pair
  - **Reduce( $k', \langle v' \rangle^*$ )**  $\rightarrow \langle k', v'' \rangle^*$ 
    - **All values  $v'$  with same key  $k'$  are reduced together**
    - There is one Reduce function call per unique key  $k'$



# MapReduce: Word Counting

Provided by the  
programmer

## MAP:

Read input and  
produces a set of  
key-value pairs

(The, 1)

(crew, 1)

(of, 1)

(the, 1)

(space, 1)

(shuttle, 1)

(Endeavor, 1)

(recently, 1)

....

(key, value)

## Group by key:

Collect all pairs  
with same key

(crew, 1)

(crew, 1)

(space, 1)

(the, 1)

(the, 1)

(the, 1)

(shuttle, 1)

(recently, 1)

...

(key, value)

Provided by the  
programmer

## Reduce:

Collect all values  
belonging to the  
key and output

(crew, 2)

(space, 1)

(the, 3)

(shuttle, 1)

(recently, 1)

...

(key, value)

Only sequential reads

The crew of the space shuttle Endeavor recently returned to Earth as ambassadors, harbingers of a new era of space exploration. Scientists at NASA are saying that the recent assembly of the Dextre bot is the first step in a long-term space-based man/machine partnership. "The work we're doing now -- the robotics we're doing -- is what we're going to need .....

Big document

# Word Count Using MapReduce

**map(key, value) :**

```
// key: document name; value: text of the document
  for each word w in value:
    emit(w, 1)
```

**reduce(key, values) :**

```
// key: a word; value: an iterator over counts
  result = 0
  for each count v in values:
    result += v
  emit(key, result)
```

# Example: Host size

- Suppose we have a large web corpus with a metadata file formatted as follows:
  - Each record of the form: (URL, size, date, ...)
- For each host, find the total number of bytes
- Map
  - For each record, output (hostname(URL), size)
- Reduce
  - Sum the sizes for each host

# Example: Language Model

- Count number of times each 5-word sequence occurs in a large corpus of documents
- Map
  - Extract (5-word sequence, count) from document
- Reduce
  - Combine the counts