Introduction to Data Science

(Lecture 22)

Dr. Mohammad Pourhomayoun

Assistant Professor

Computer Science Department

California State University, Los Angeles


ALSO
BIG
DATA
TOOLS
DISK
TARGET
APPLIED
SENSOR
SHARED
DEFINITION
CURRENT
MAY
MOVING
WITHIN
EXAMPLES
EVERY
RECONSIDER
PRACTITIONERS
CAPTURE
ZETTABYTES
THOUGHT
INTERNET
BUSINESS
SETS
GENOMICS
DISTRIBUTED
DESCRIBING
RADIO-FREQUENCY
COMPLEXITY
MANAGEMENT
TERABYTES
CASE
ABILITY
PETABYTES
INCLUDE
ELAPSED
SYSTEMS
TOLERABLE
SIZE
STORAGE
MPP
PARALLEL
SAN
QUALITIES
SINCE
MASSIVELY
GROW
CONNECTOMICS
ORGANIZATIONS
SET
USE
RELATIONAL
LARGER
CONTINUES
SOCIAL
INDEXING
TENS
COMPLEX
CITATION
ANALYTICS
NOW
BIOLOGICAL
PROCESSING
UBIQUITOUS
CAPACITY
BURIED
WORLD'S
HUNDREDS
DESKTOP
RECORDS
NETWORKS
DATABASES
SOLID
TYPES
CURRENTLY
GARTNER
WORKING
AMOUNT
OPPORTUNITIES
SEARCH
DIFFICULTY

Map-Reduce

One Example to Start …

- What is the amount of digital data stored in **a single human body cell**?
- Answer: 1.5×10^9 bytes = 1.5 Gbytes !!!!!

Example: Genomics

- **Genomics** is a discipline in **genetics** that applies recombinant DNA, DNA sequencing methods, and **bioinformatics** to assemble, sequence, and analyze the function and structure of **genomes** [1].
- **Genome** is the complete set of DNA within a single cell of an organism.
- **Bioinformatics** is an interdisciplinary field that develops methods and software tools for understanding biological data. Bioinformatics combines **computer science (machine learning & big data), statistics, mathematics,** and **engineering** to analyze and interpret biological data [1].

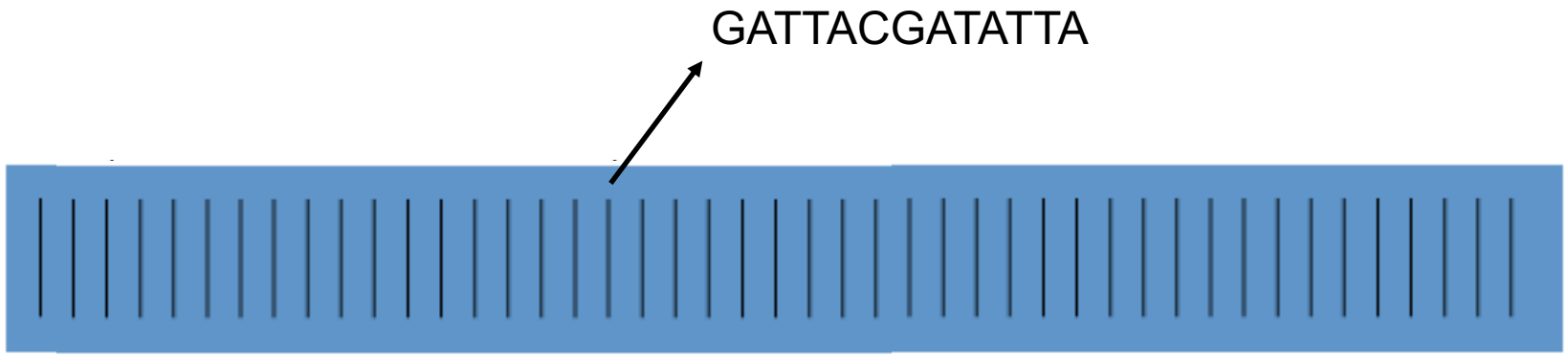
1: Wikipedia

One Example to Start …

- **Genomics:**
  - What is DNA and How Does it Work?
    - https://www.youtube.com/watch?v=ZaJ64lyqrIs
  - What is Genome?
    - https://www.youtube.com/watch?v=rvryNYlbfKA
  - How to sequence the human genome?
    - https://www.youtube.com/watch?v=MvuYATh7Y74

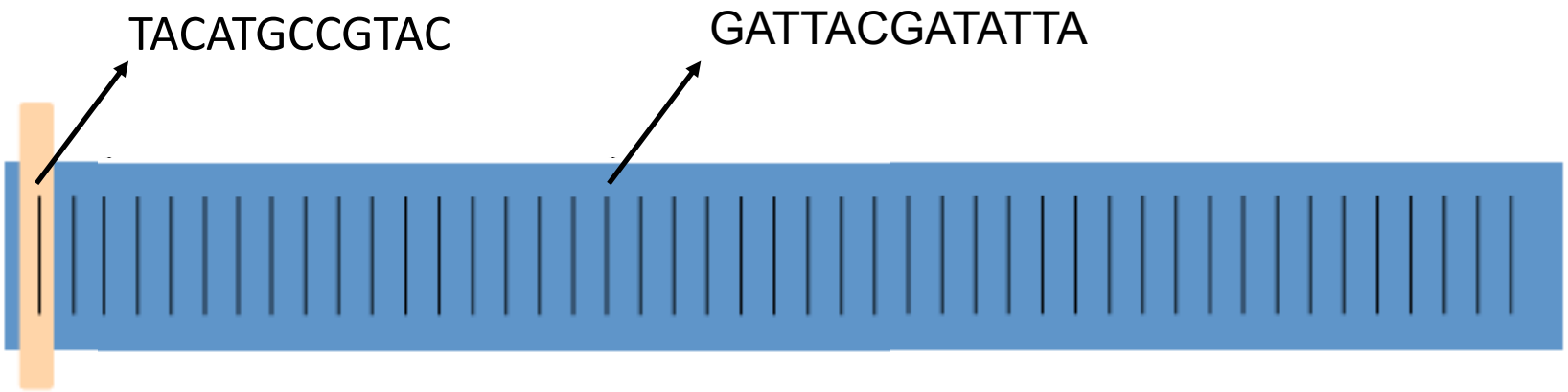
Example*: Find Matching DNA Sequences

- Given a set of sequences, Find all sequences equal to:
  - "GATTACGATATTA"



* Example from Bill Howe, University of Washington

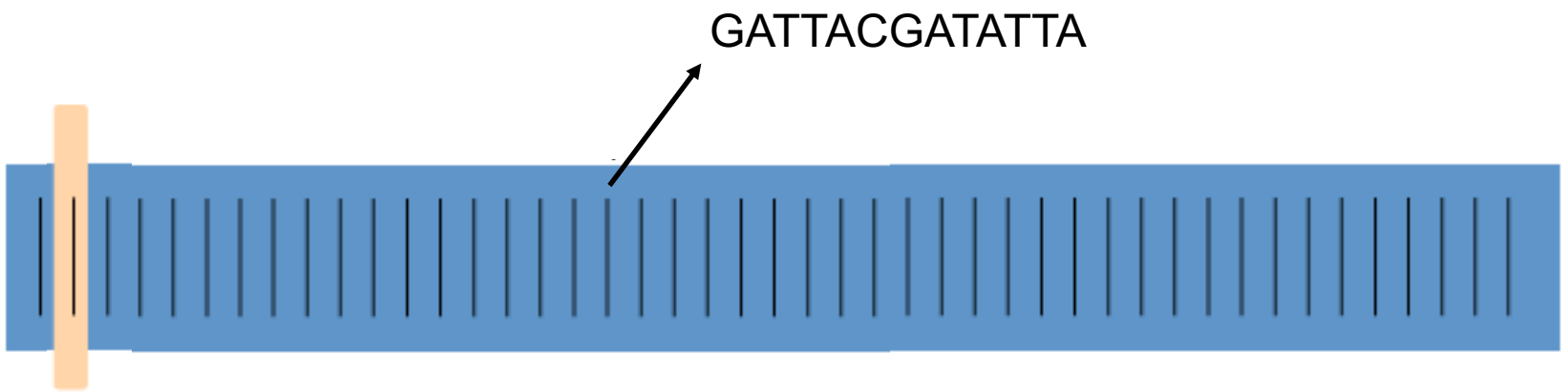
Example: Find Matching DNA



Time 0:

Is (TACATGCCGTAC = GATTACGATATTA)? → No

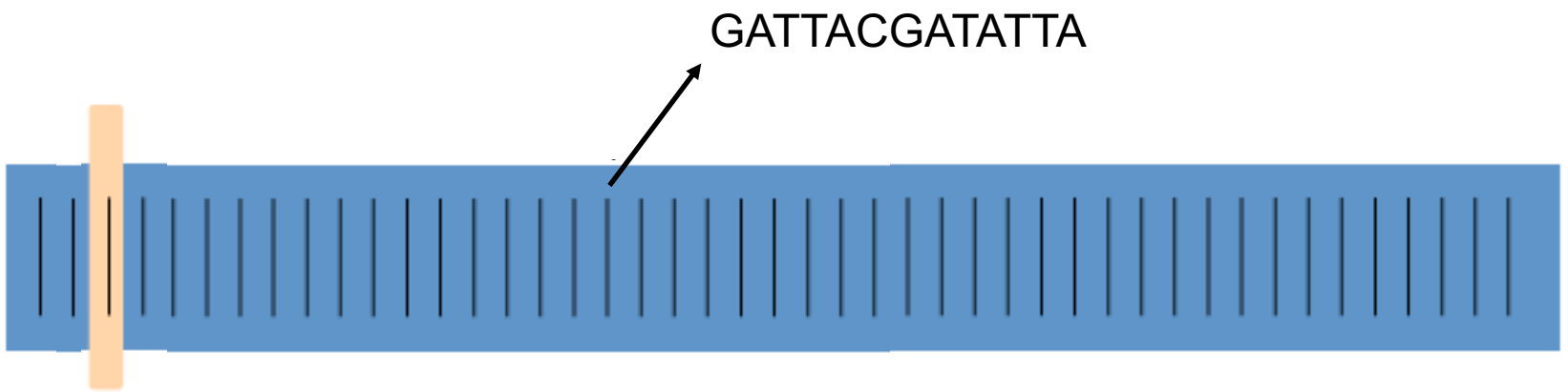
Example: Find Matching DNA



Time 1:

Is (TAATGCCGTACGC = GATTACGATATTA)? → No

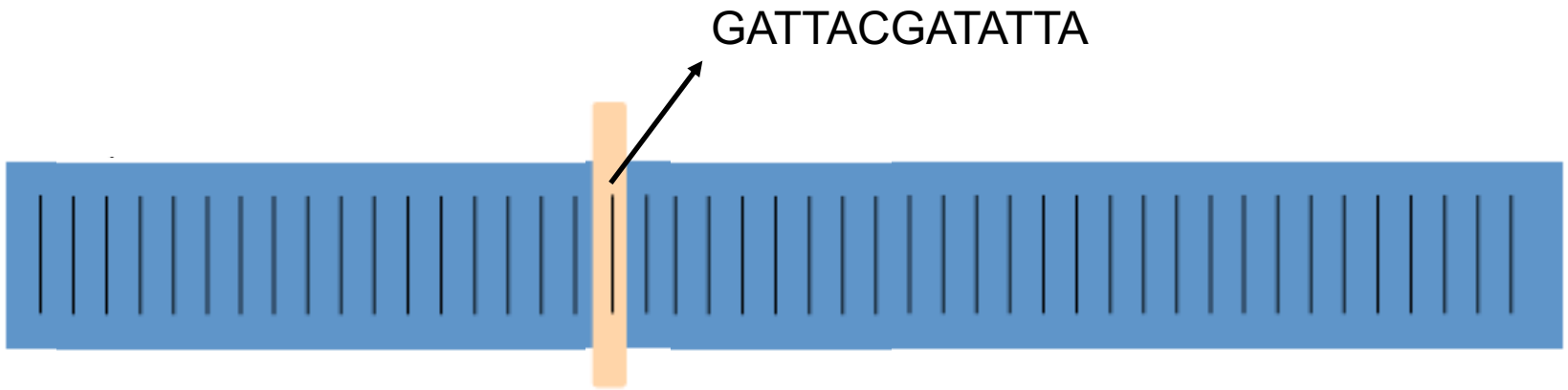
Example: Find Matching DNA



Time 2:

Is (TACGTAGCATGC = GATTACGATATTA)? → No

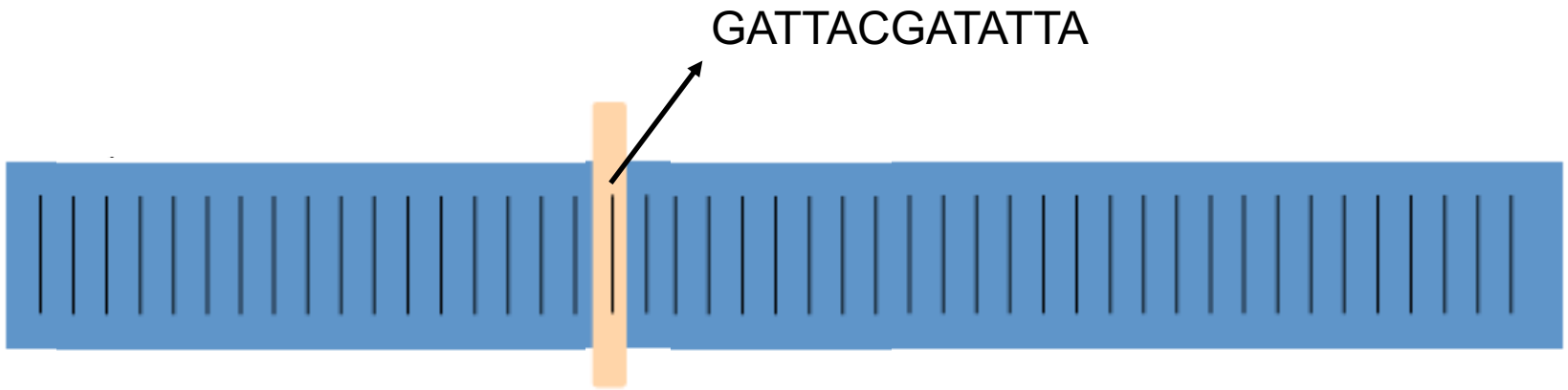
Example: Find Matching DNA



Time 17:

Is (ATTACGATATTA = GATTACGATATTA)? → Yes

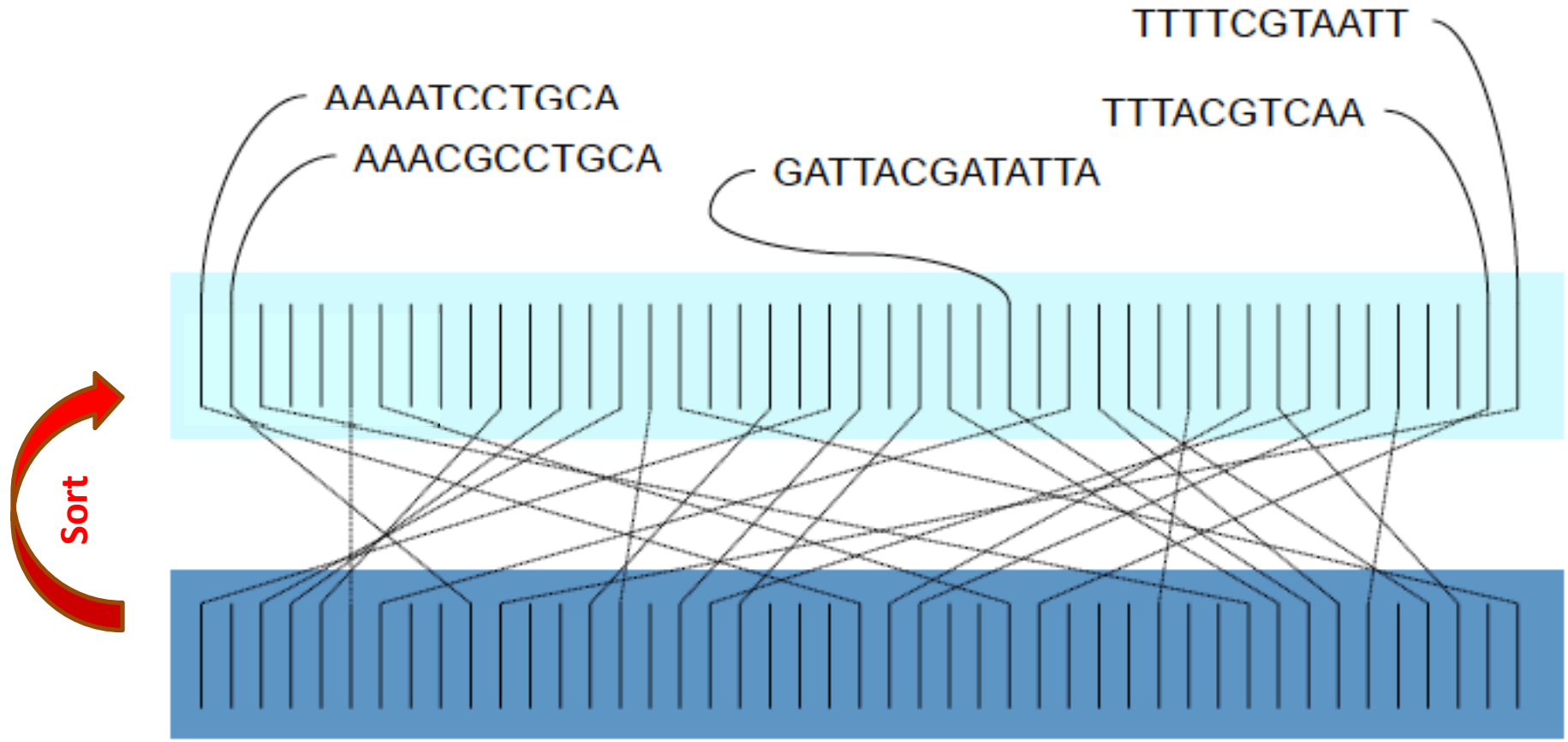
Example: Find Matching DNA



N records, N comparisons

The algorithmic complexity is order N (linear): O(N)

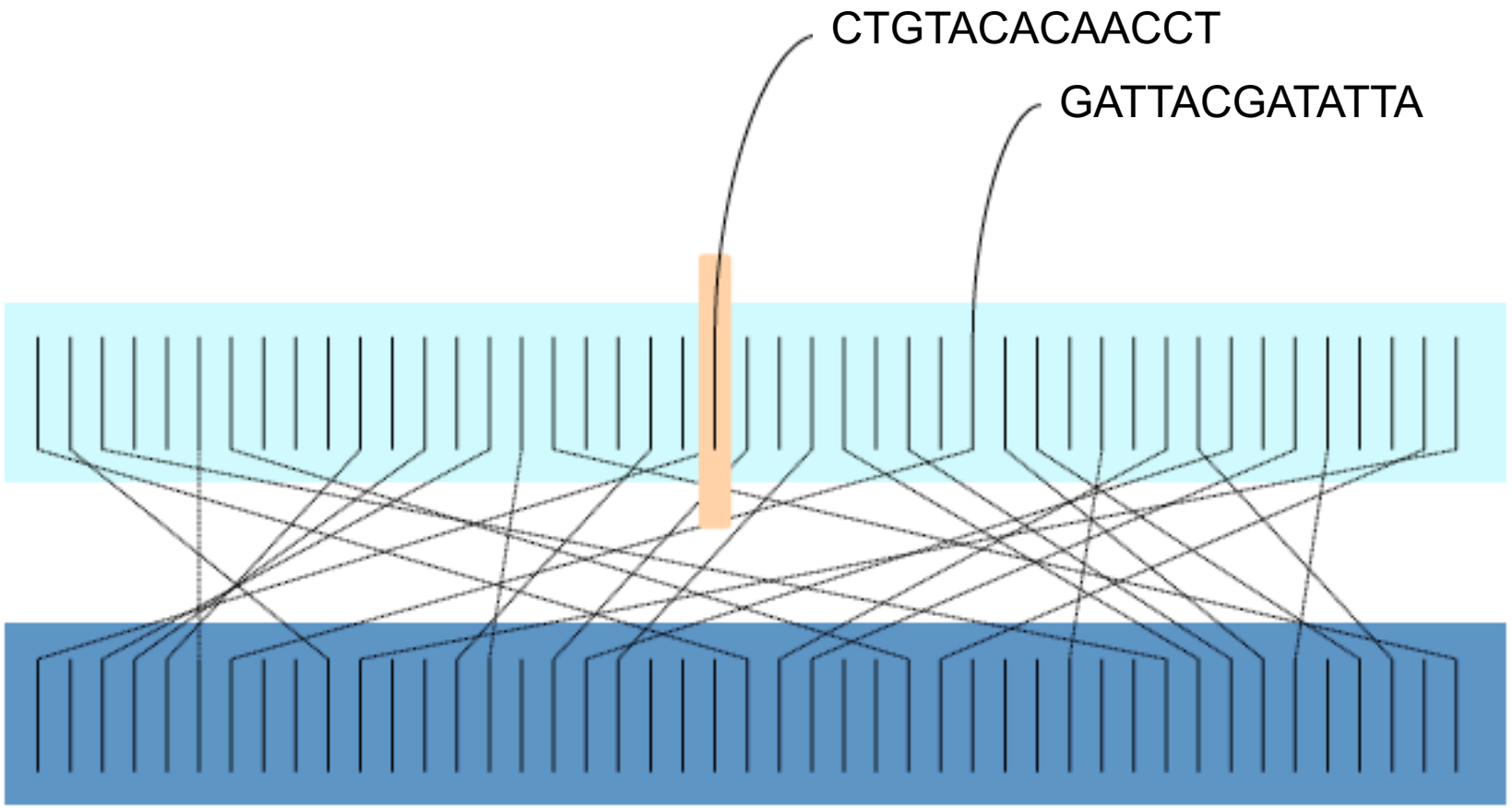
- What if we want to do this search Many times?
- Then, a better approach is Binary Search:
  - Let's Sort the set:



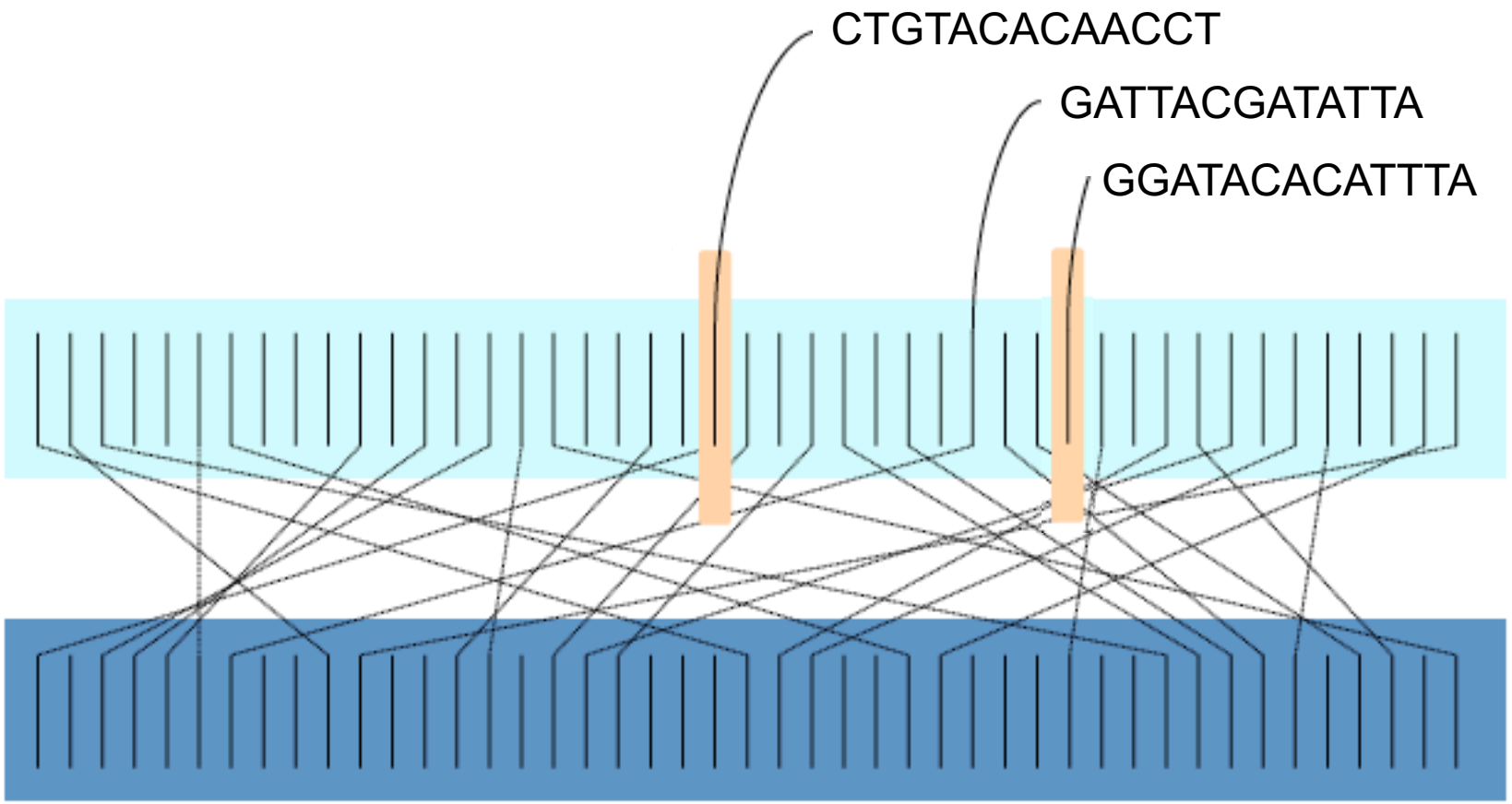
* Example from Bill Howe, University of Washington

- Start at the middle:

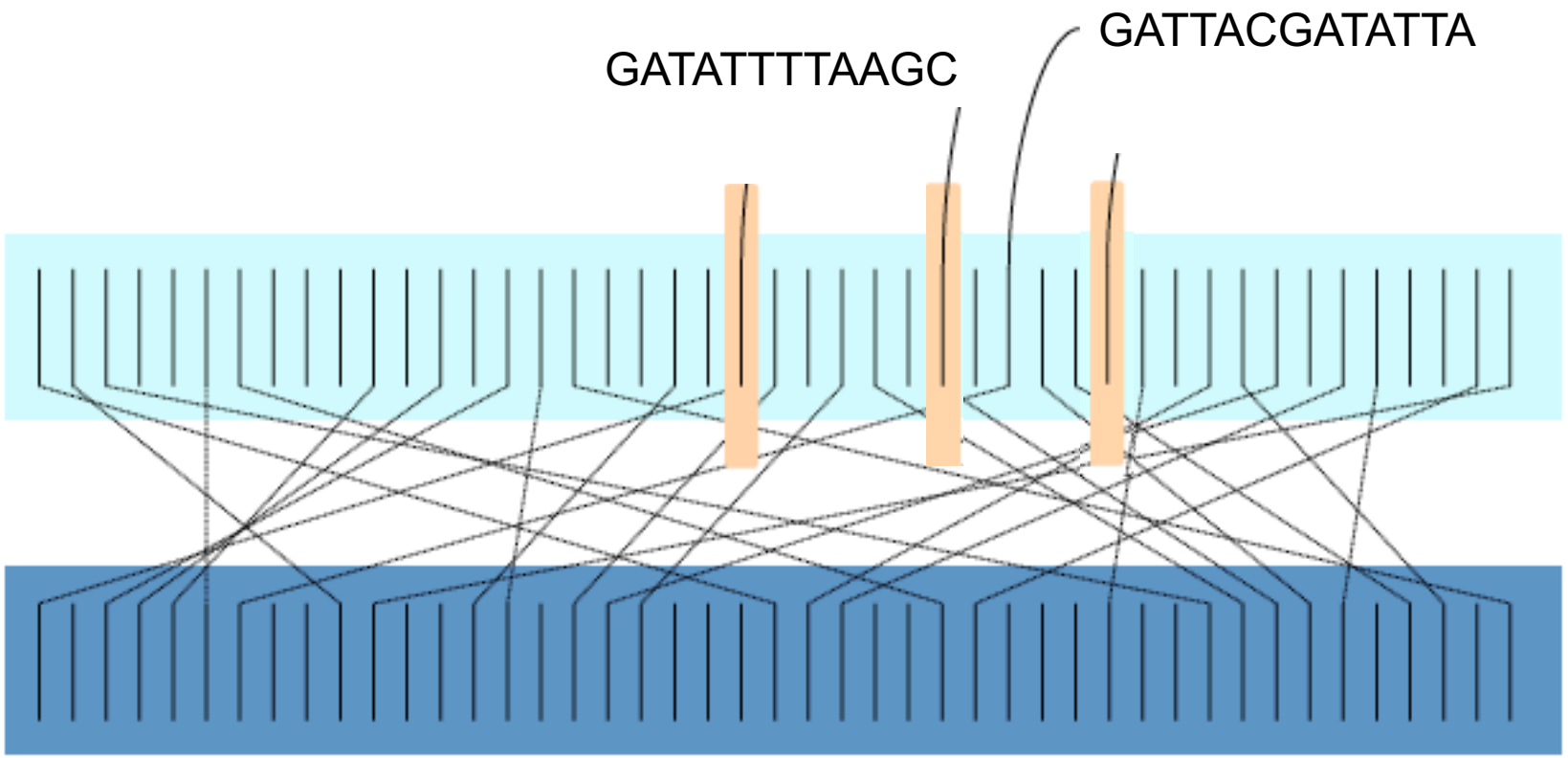
  CTGTACACAACCT < **GATTACGATATTA** → Right Side



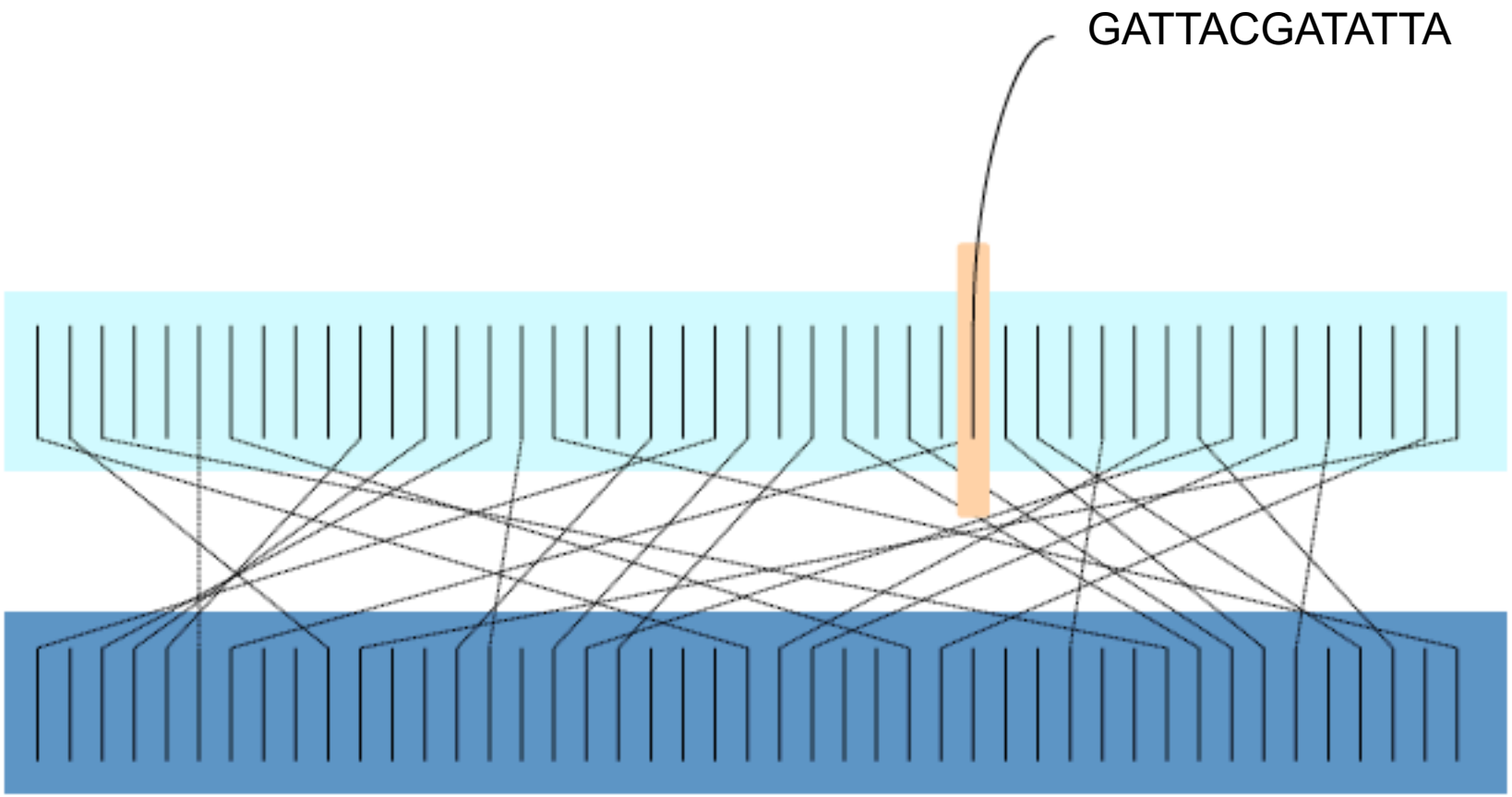
- GGATACACATTTA > **GATTACGATATTA** → Left Side



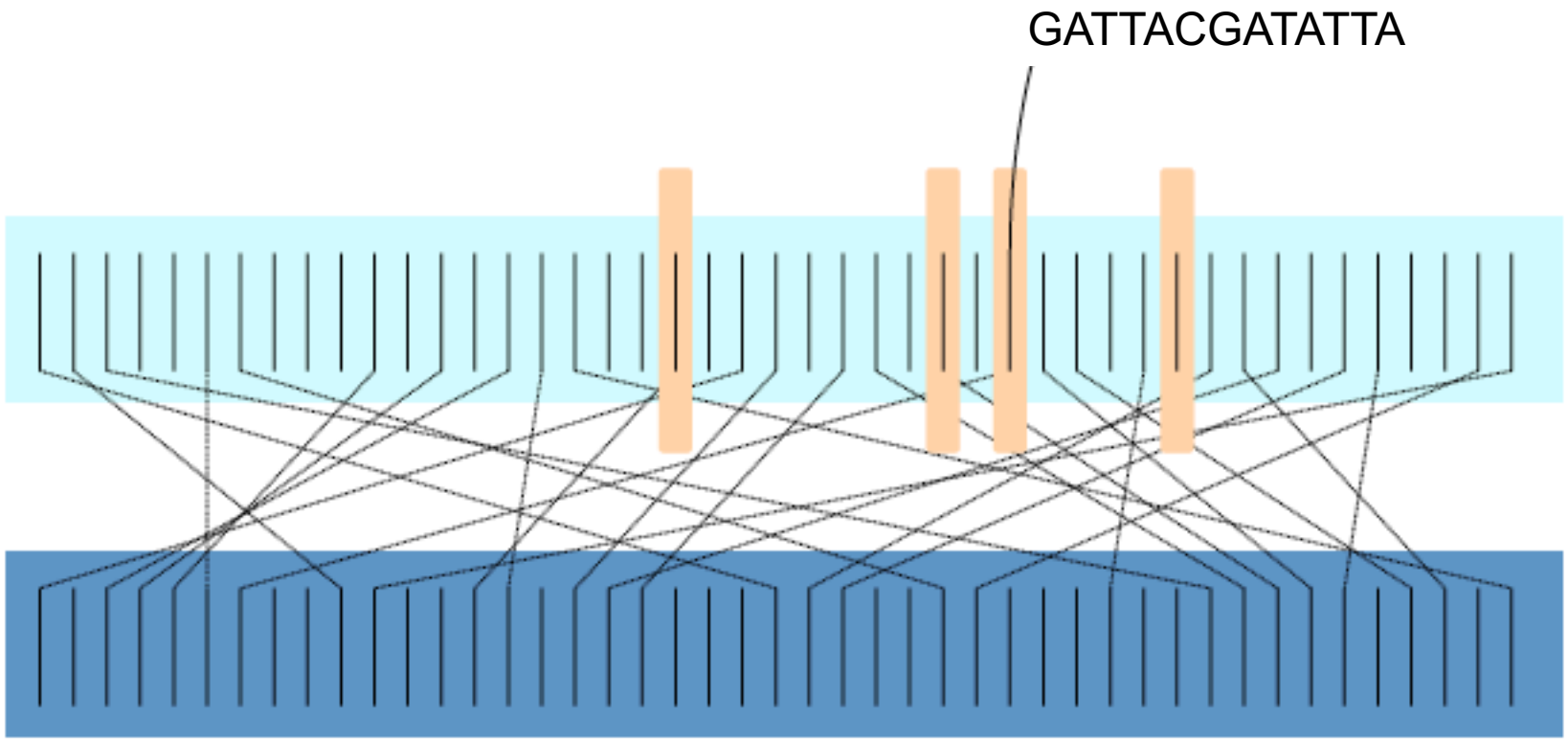
- GATATTTTAAGC < **GATTACGATATTA** → Right Side



- GATTACGATATTA = **GATTACGATATTA** → Match!!!

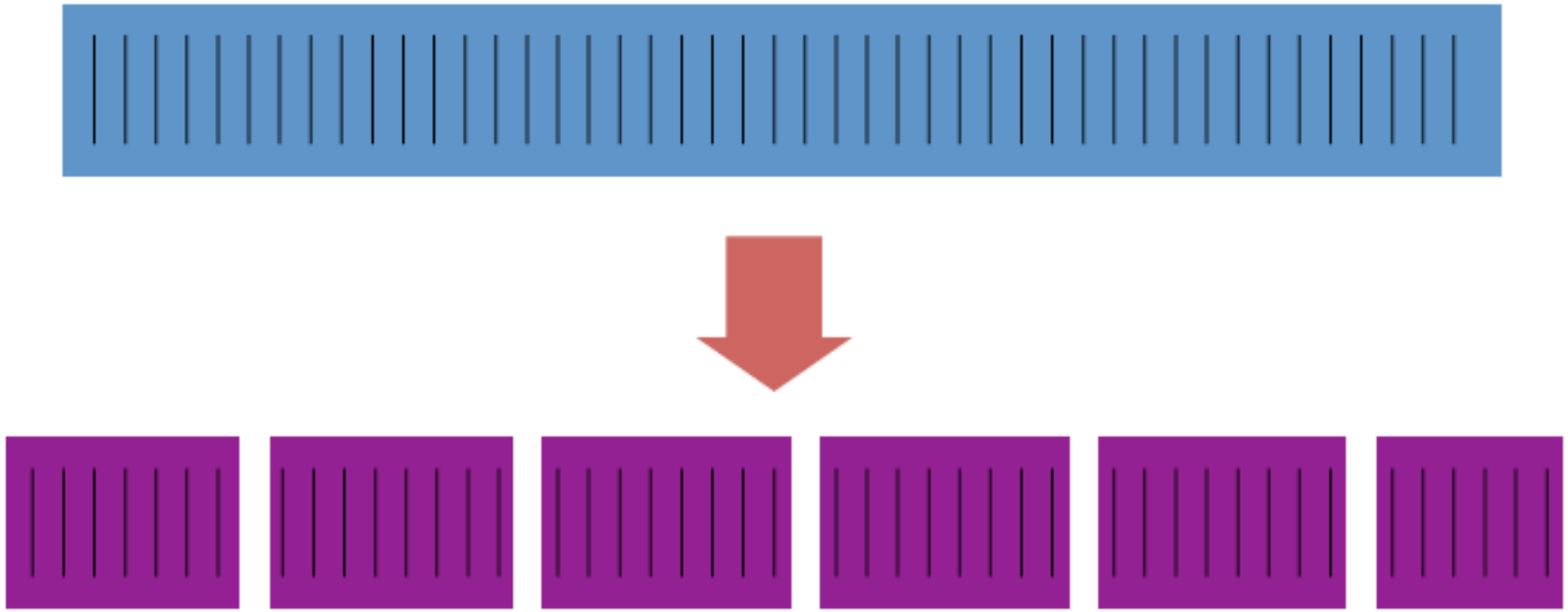


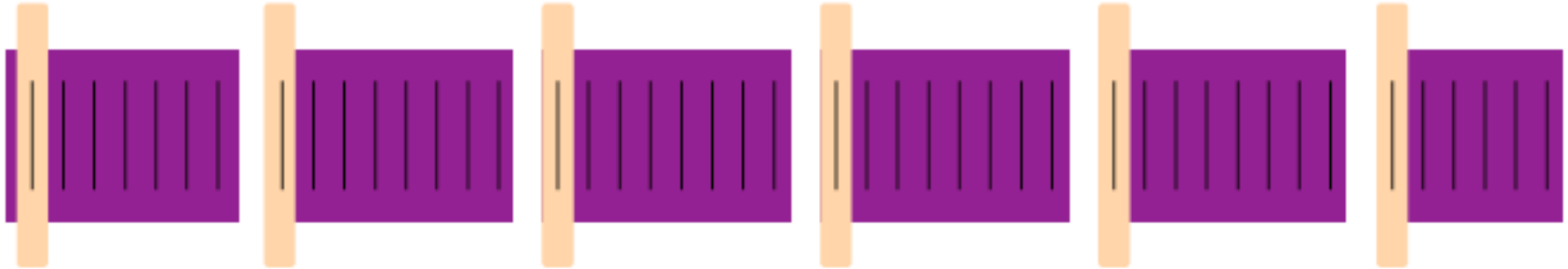
- 40 records, only 4 comparisons
- N records, **log(N)** comparisons
- This algorithm is **O(log(N))** → **Much Better Scalability!**



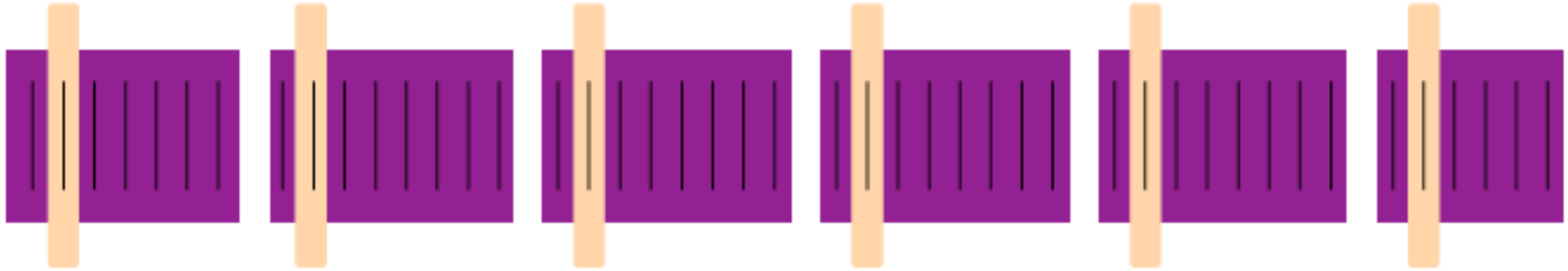
Can we do any better?

- **Can we do any better?**
- Yes! Because the processing of each comparison is independent from others, so we can split a big task to several smaller tasks and process them in parallel!

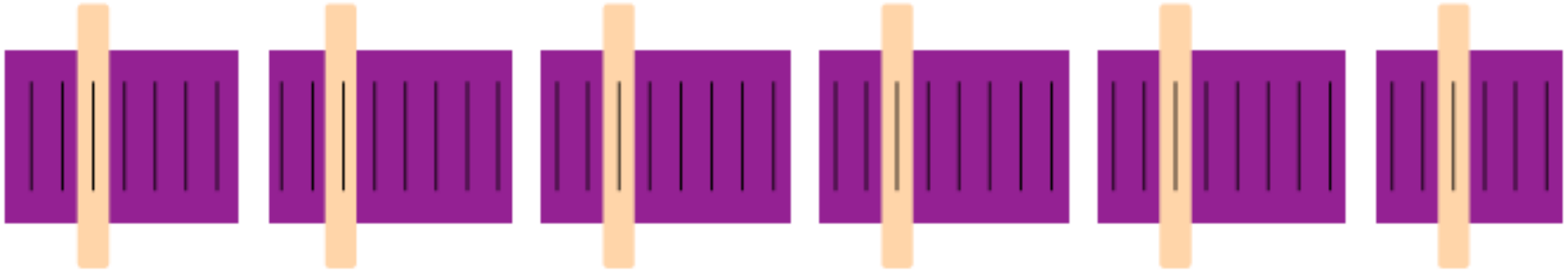




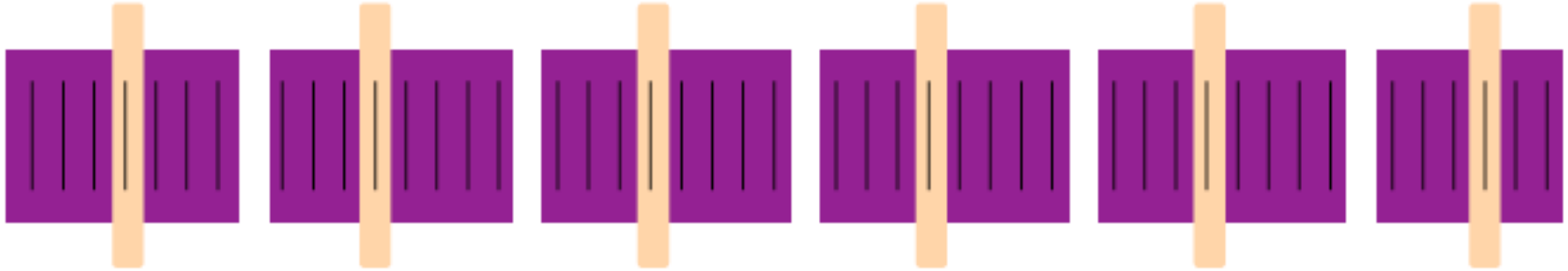
Time 0



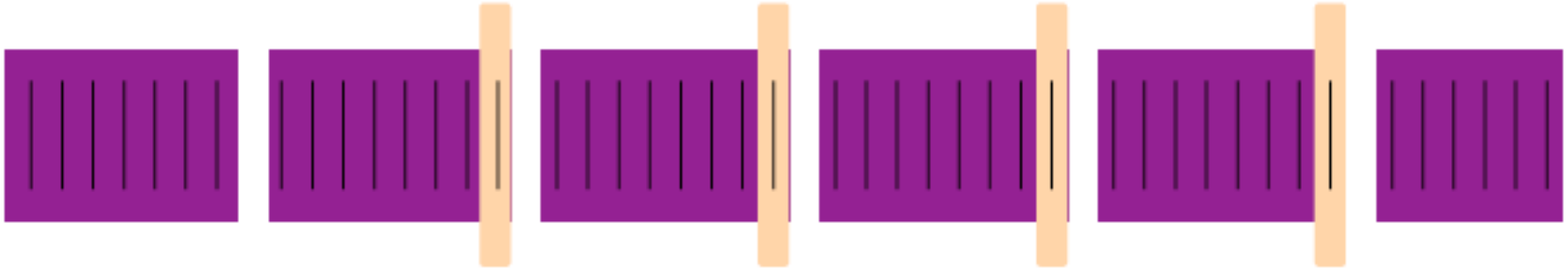
Time 1



Time 2



Time 3



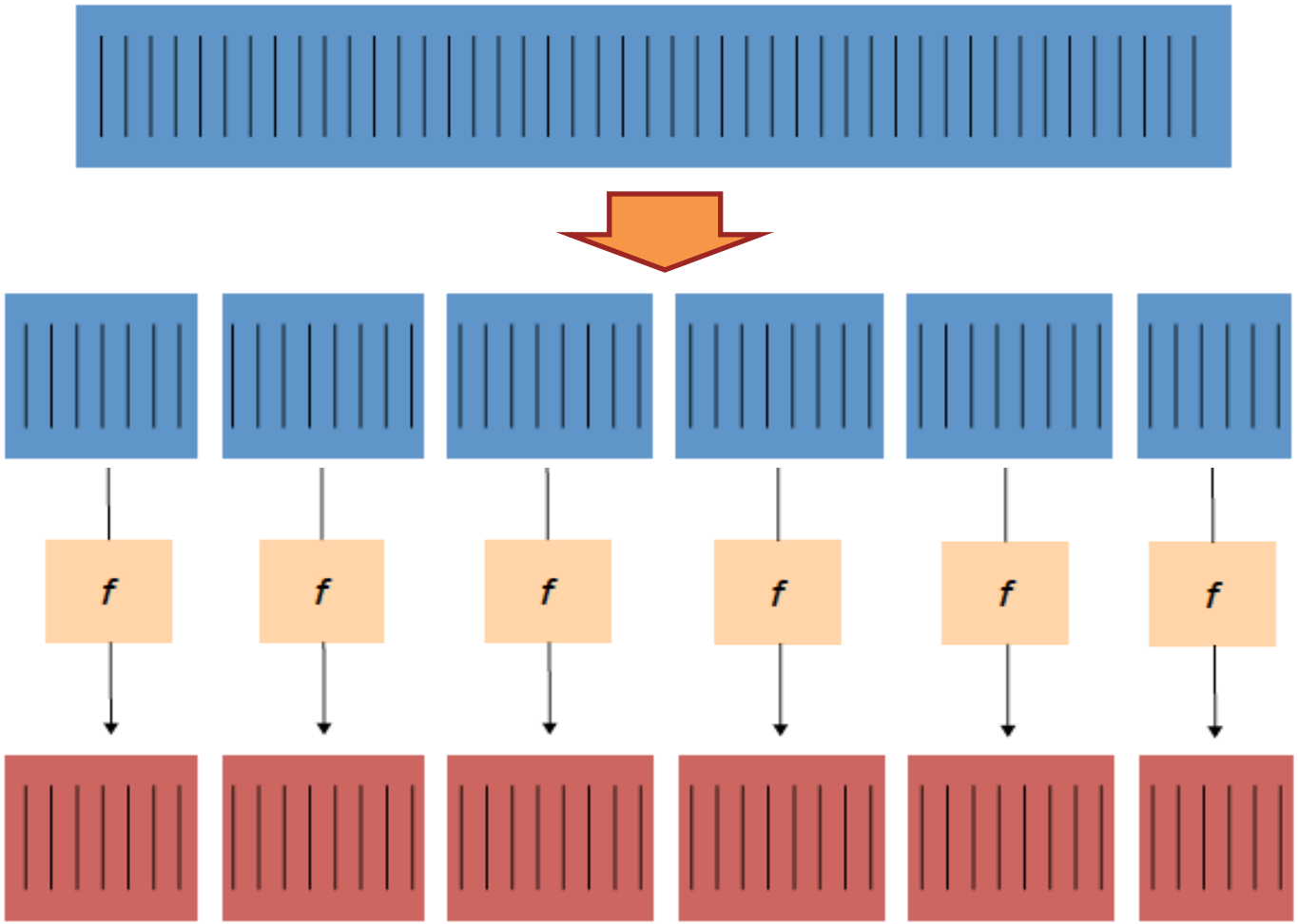
Time 7

- How much time did this take?
- 7 cycles 40 records, 6 workers (6 processors) → O(N/k)

- Although O(N/k) is same as O(N) in theory of complexity, but it can be extremely helpful in practice!

How does Map Reduce work?

Example[1]: Converting 405k TIFF Images to PNG



You are given TIFF images

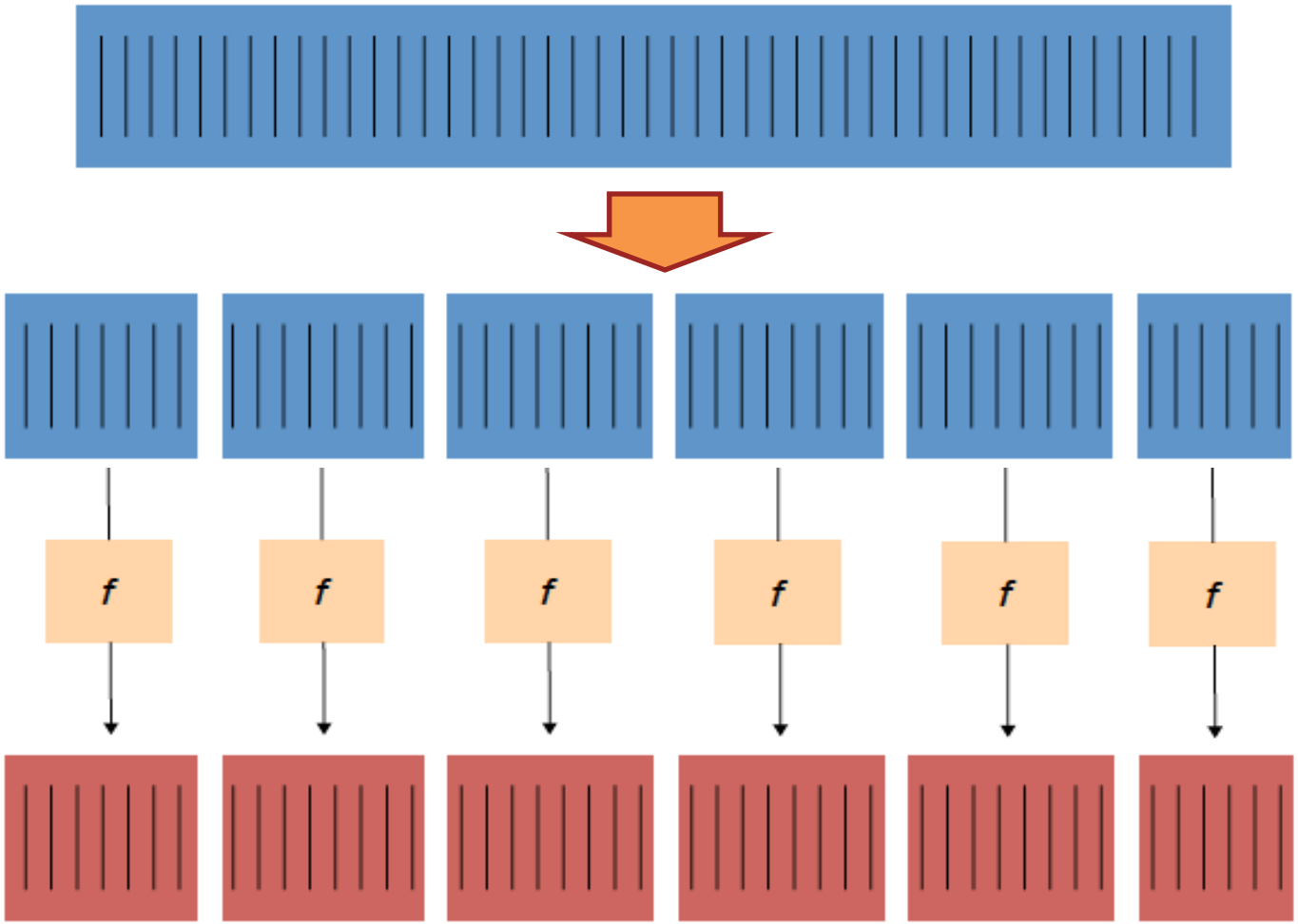
Distribute the images among k computers

f is a function to convert TIFF to PNG; apply it to every item

Now we have a big distributed set of converted images

* Example from blogs.nytimes.com, and Bill Howe, University of Washington

Example: Find the most common word in each document



You have millions of documents

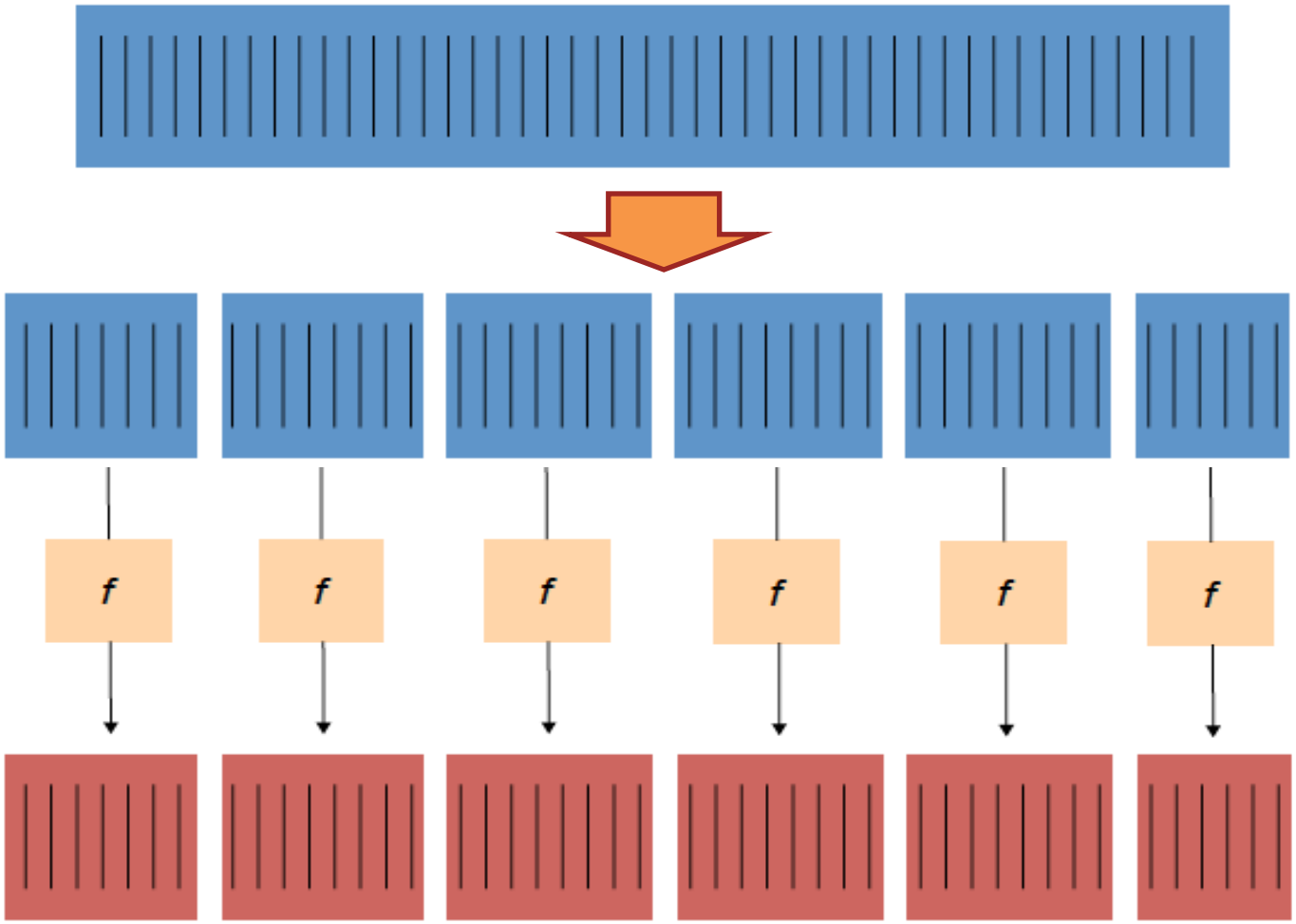
Distribute the documents among k computers

f finds the most common word in each document

Now we have a big distributed list of **(doc_id, word)** pairs

* Example from Bill Howe, University of Washington

Example: Compute overall word frequency across 5M docs



You have millions of documents

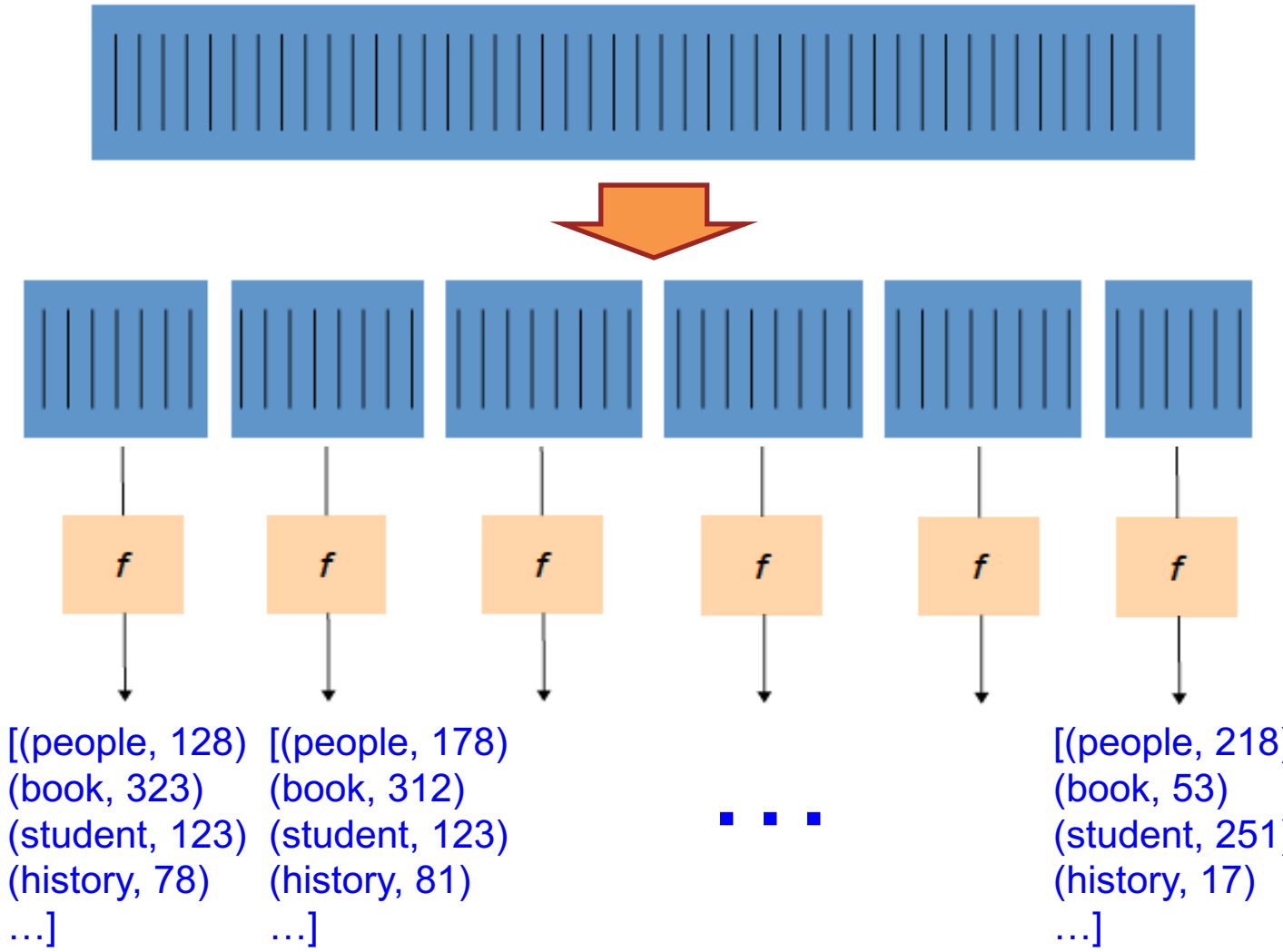
Distribute the documents among k computers

For each document, ***f*** returns a set of (word, freq) pairs

Now we have a big distributed list of **(word,freqs)** pairs for each set of docs

* Example from Bill Howe, University of Washington

Continue: Compute overall word frequency across 5M docs



You have millions of documents

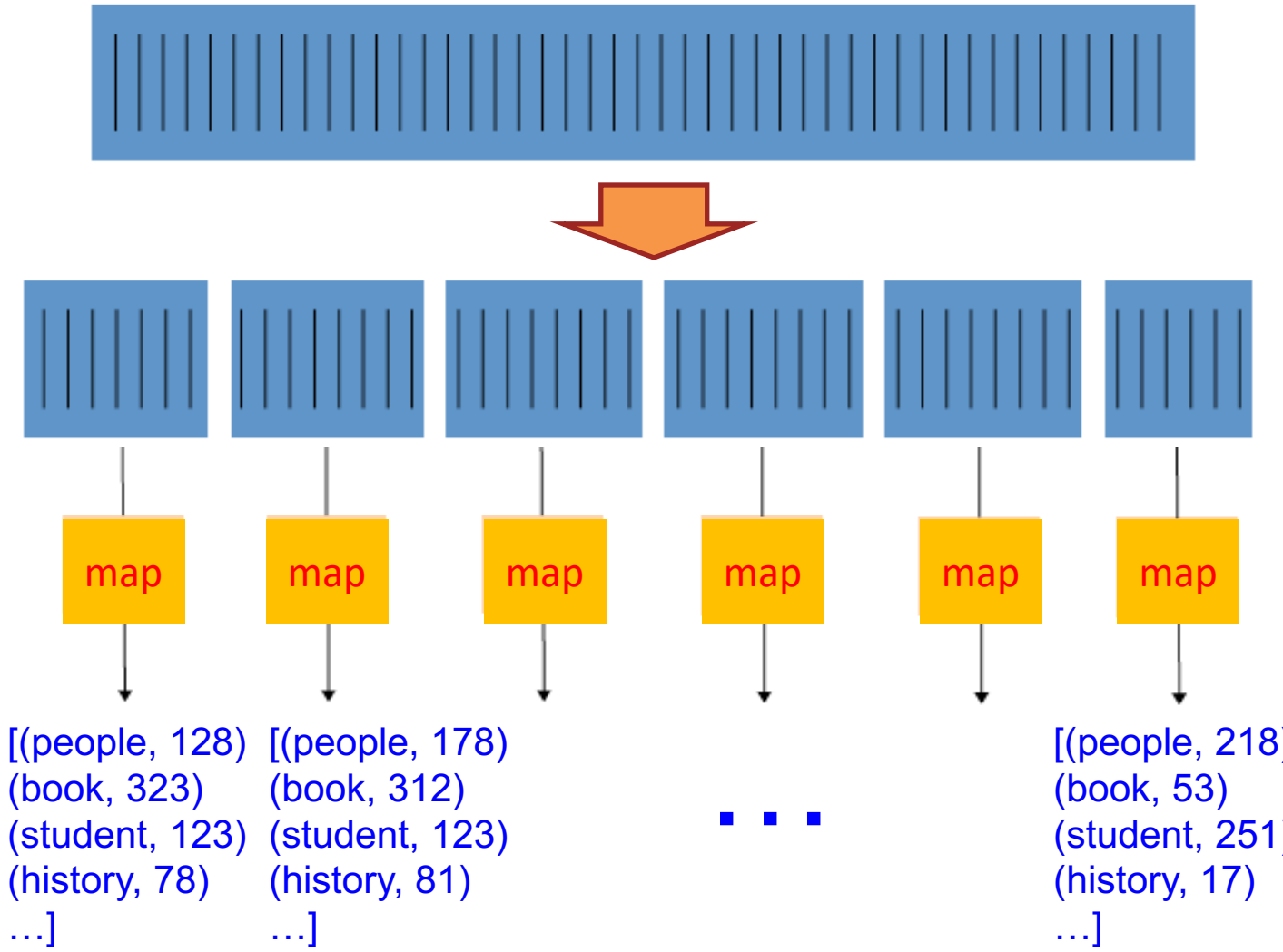
Distribute the documents among k computers

For each document, f returns a set of **(word, freq)** pairs

MAP

- In the 1st example, function “f” **maps** a **TIFF image to a PNG image**.
- In the 2nd example, function “f” **maps** a **document to its most common word**.
- In the 3rd example, function “f” **maps** a **set of documents to its word frequencies**.

Continue: Compute overall word frequency across 5M docs

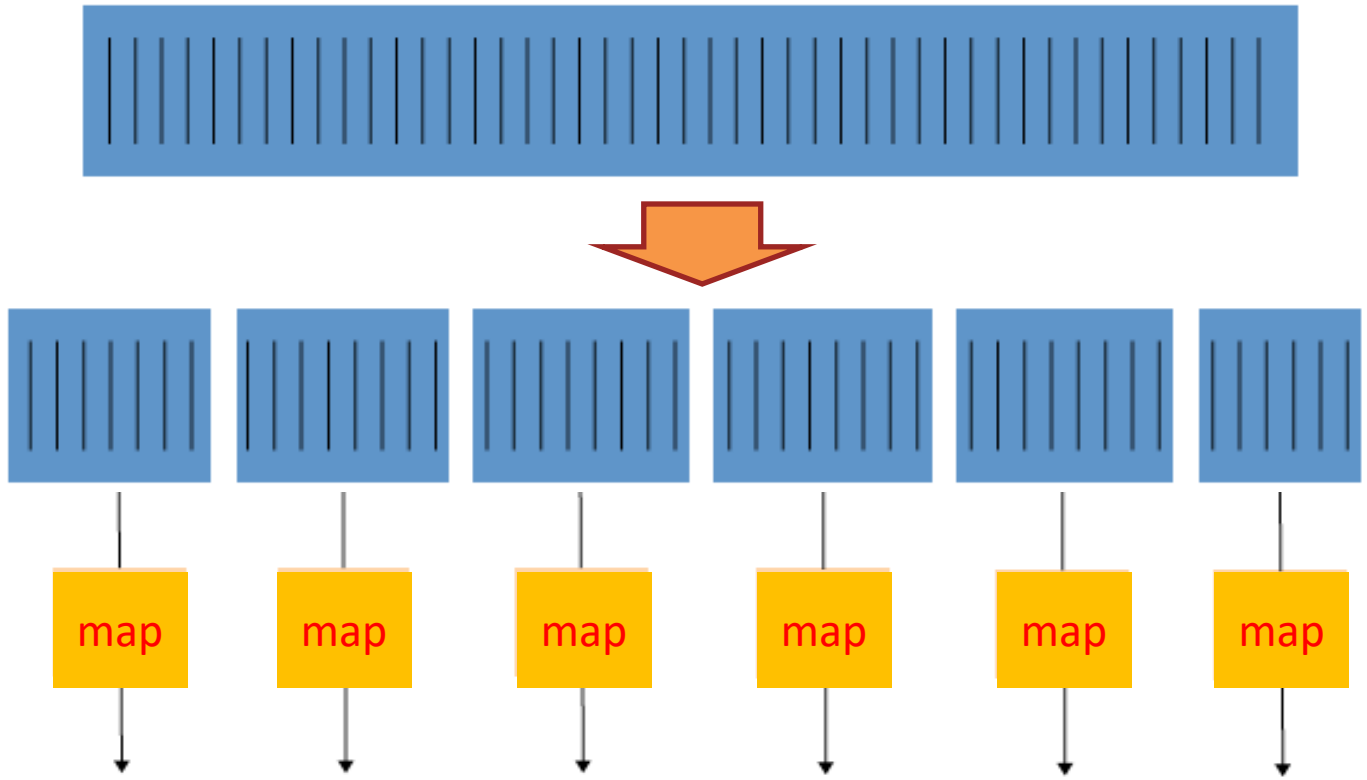


You have millions of documents

Distribute the documents among k computers

For each document f returns a set of **(word, freq)** pairs

Continue : Compute overall word frequency across 5M docs

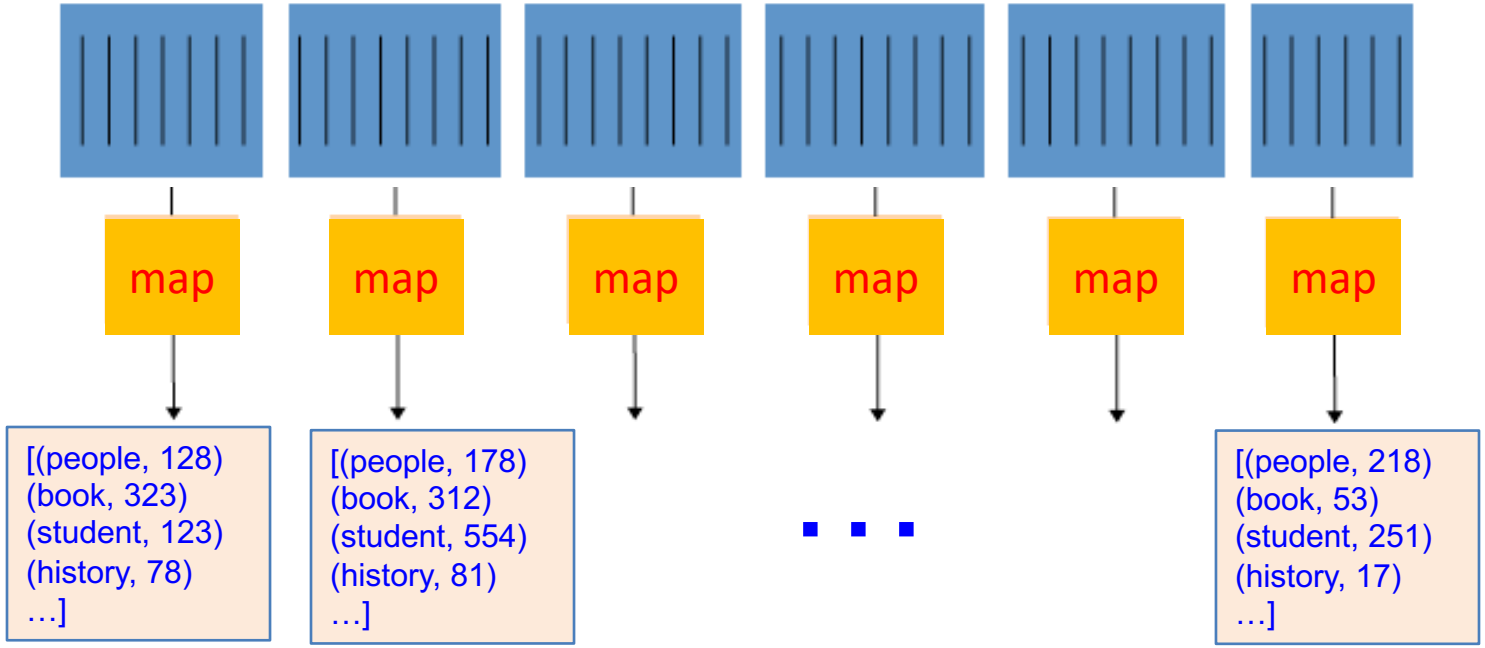


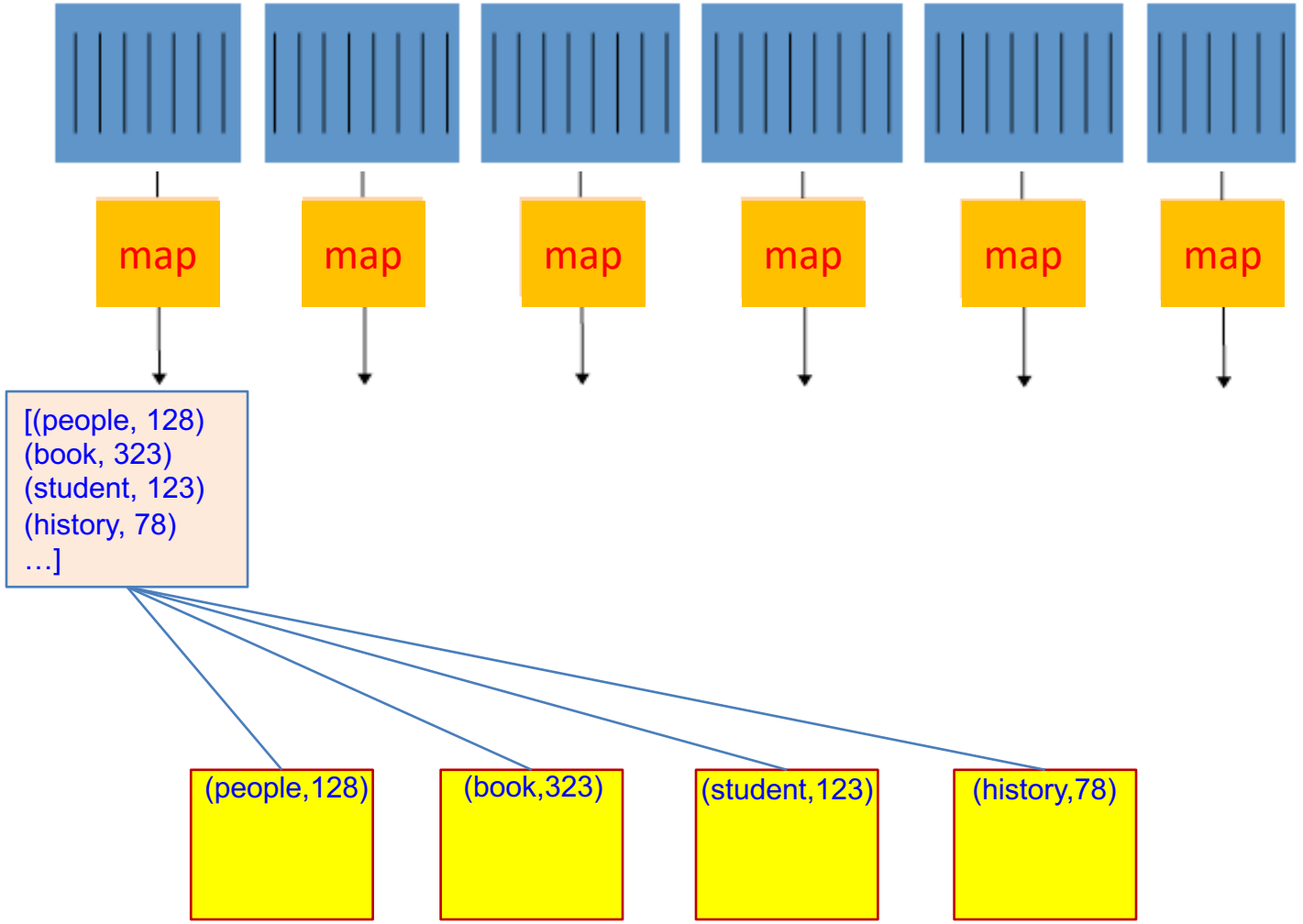
You have millions of documents

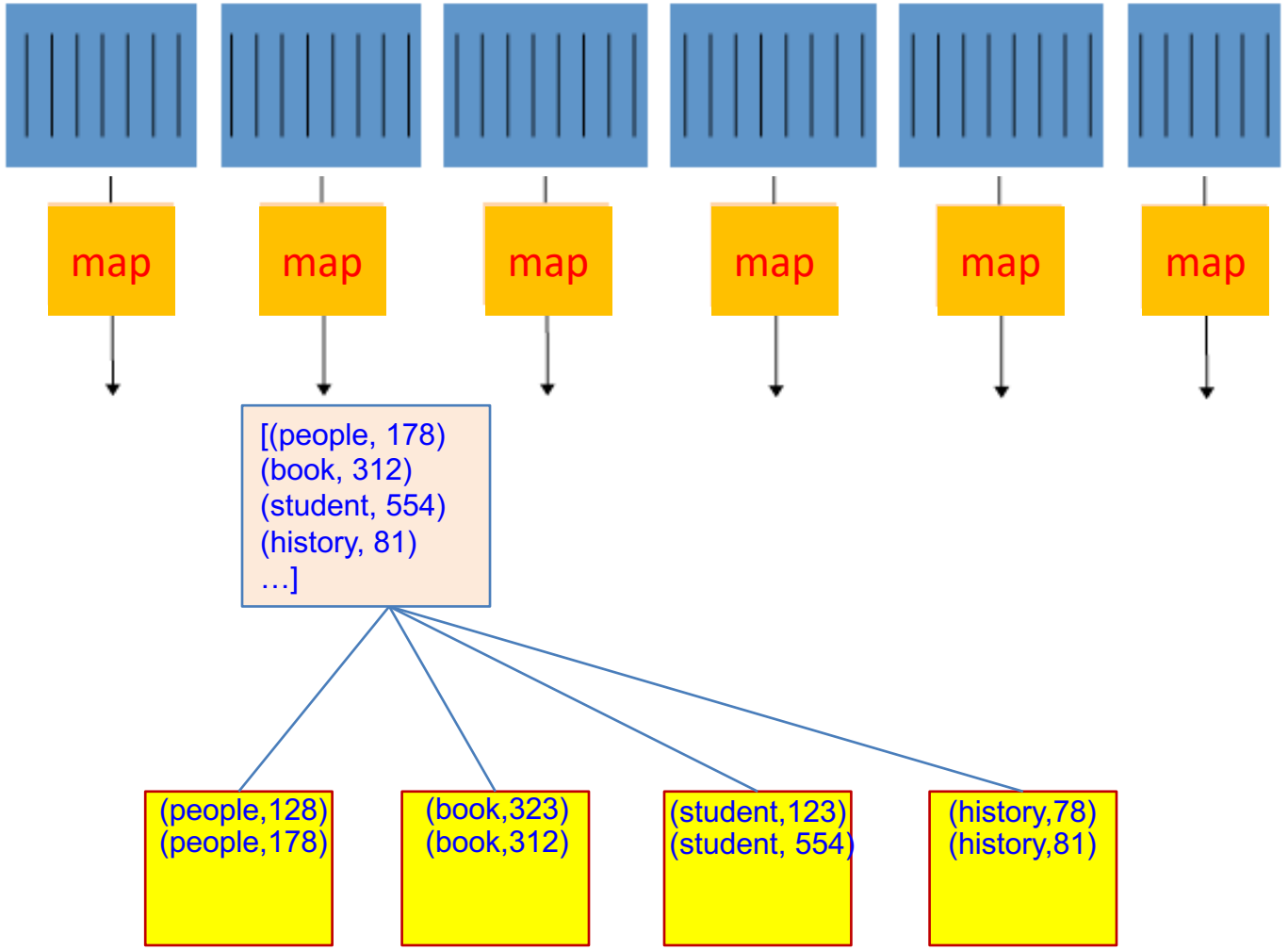
Distribute the documents among k computers

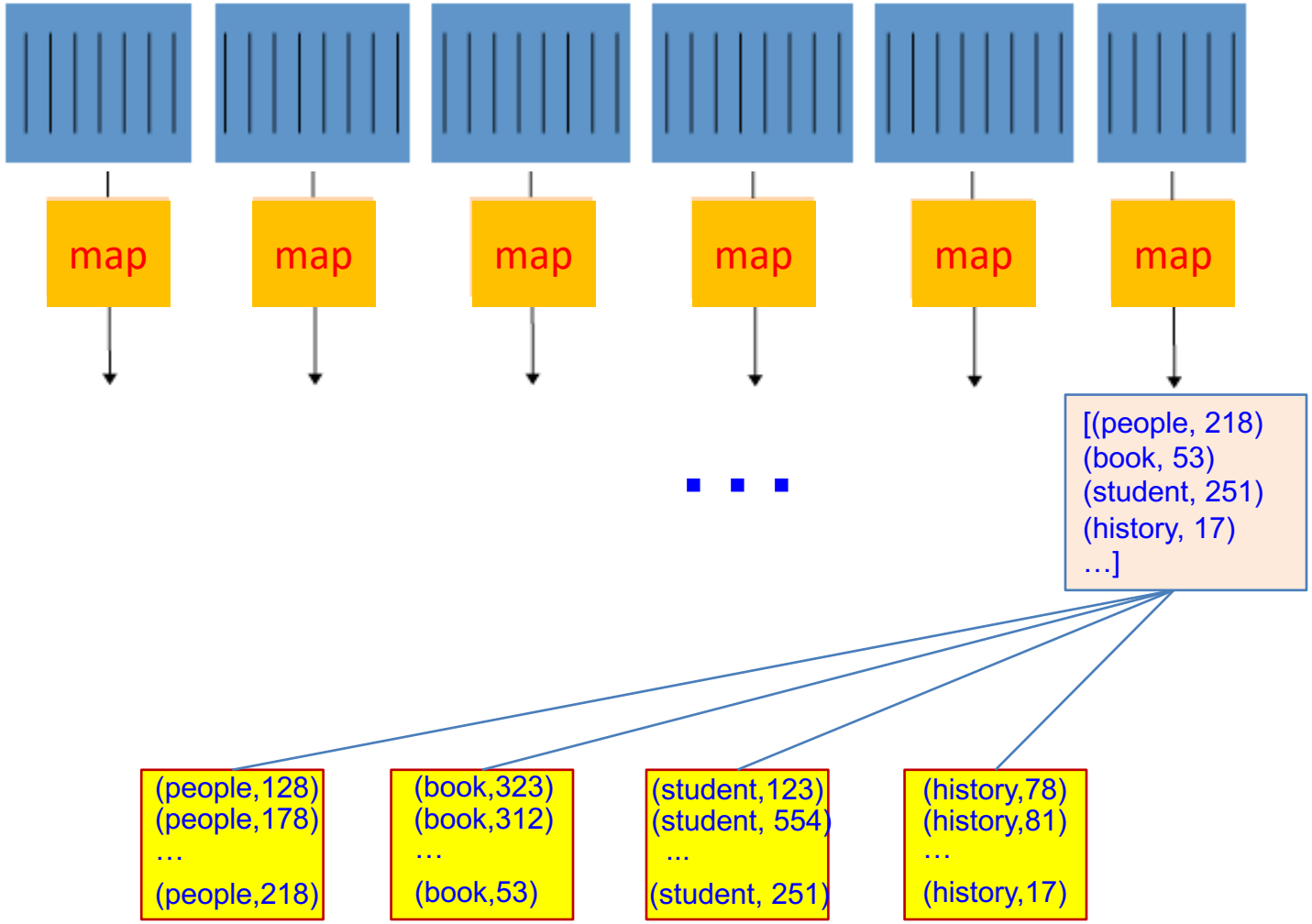
For each document f returns a set of **(word, freq)** pairs

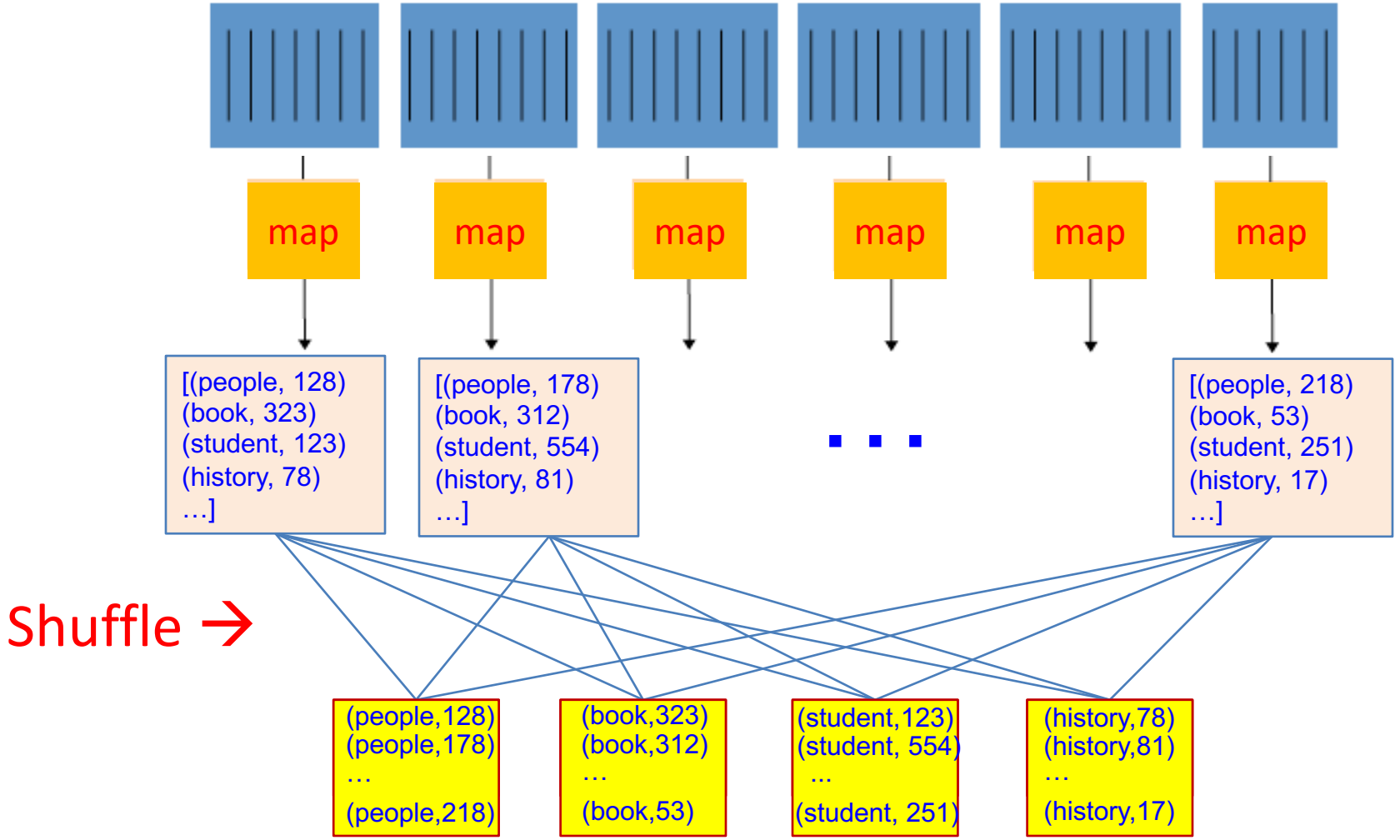
Now, we have a **list of (word,freq) pairs for each set (chunk) of docs**. We have to combine all individual lists to build an **overall** list of **(word,freq)** for all documents together.

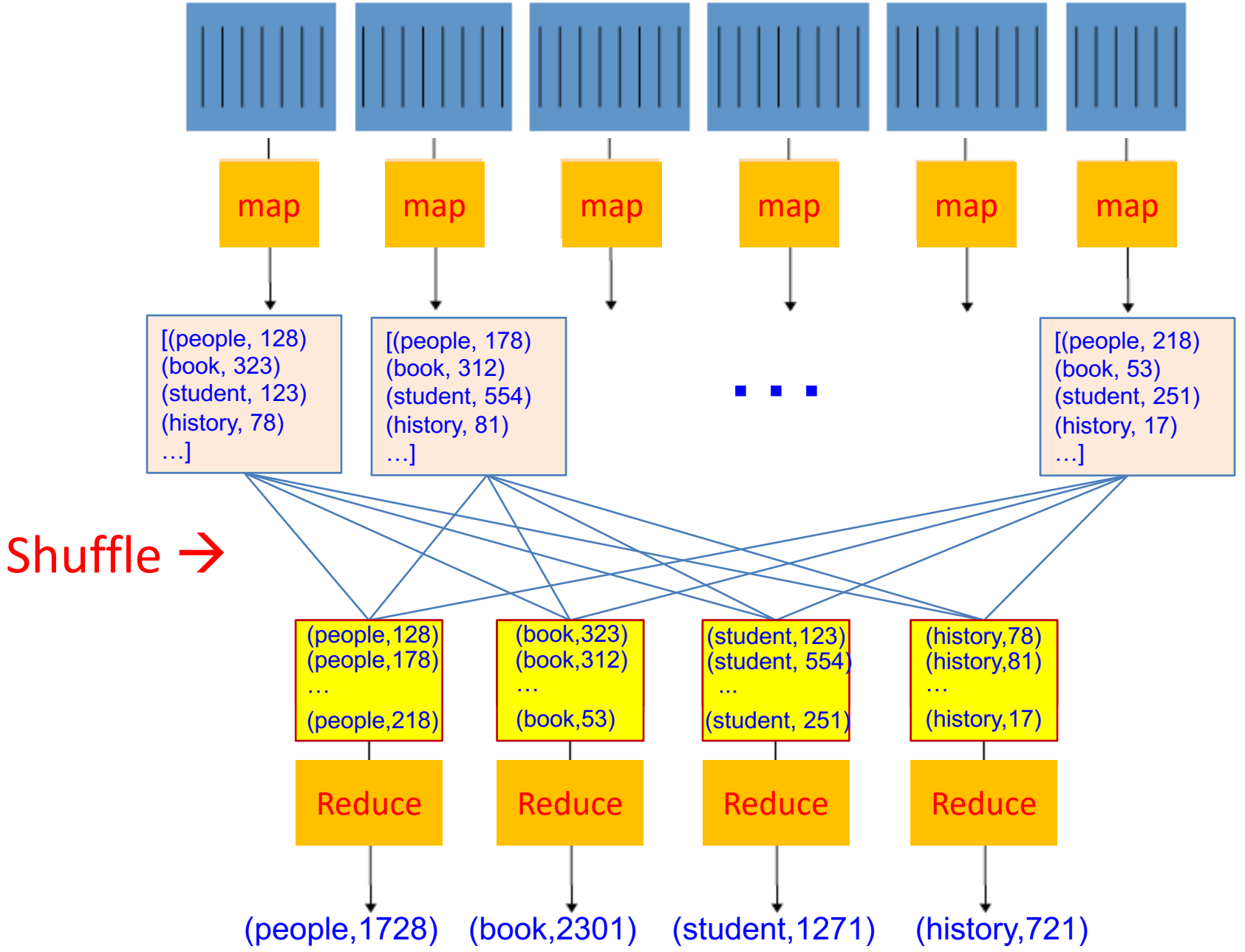

map
map
map
map
map
map
[(people, 128)
(book, 323)
(student, 123)
(history, 78)
…]
[(people, 178)
(book, 312)
(student, 554)
(history, 81)
…]
[(people, 218)
(book, 53)
(student, 251)
(history, 17)
…]

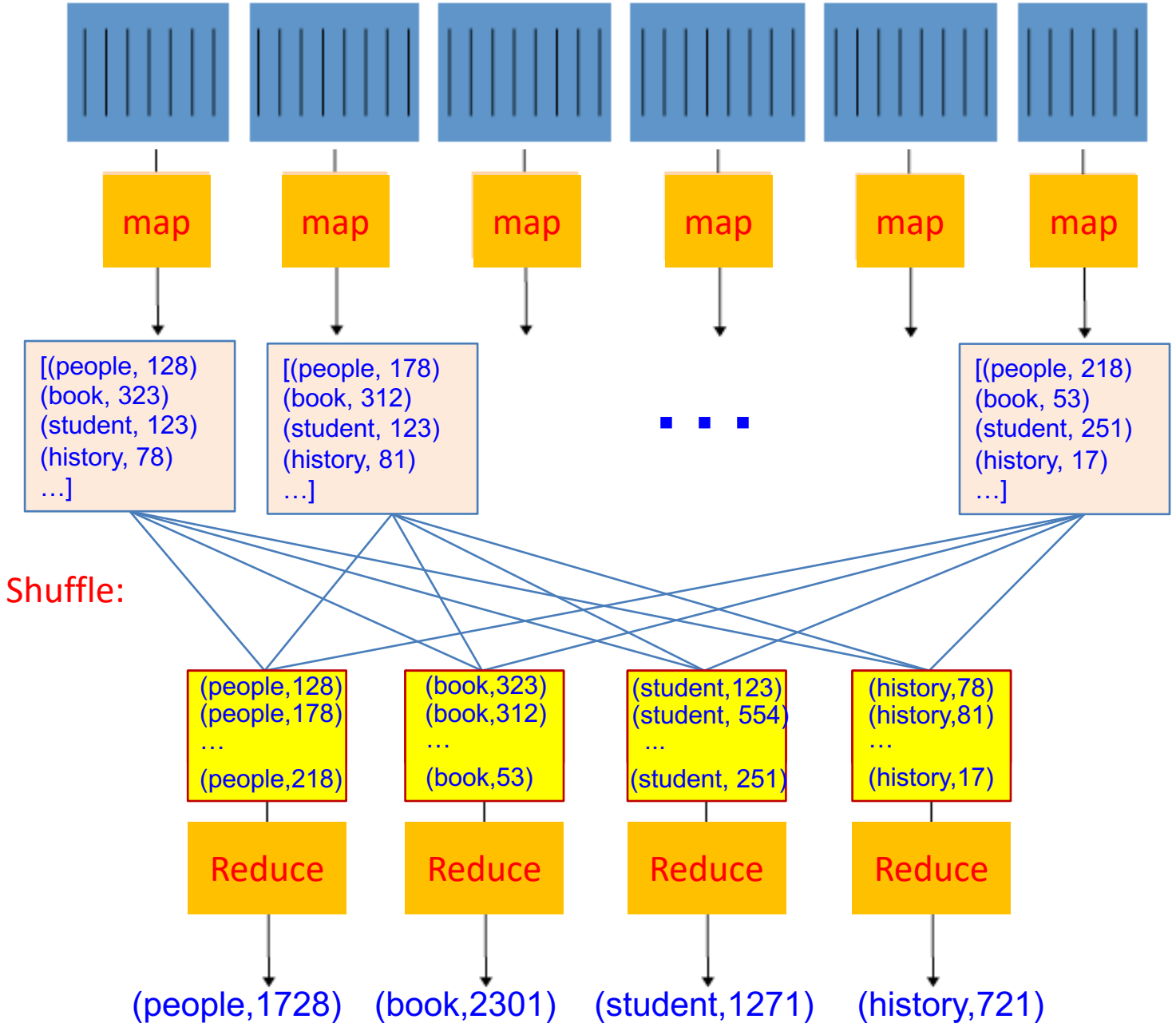

map
map
map
map
map
map
[(people, 128)
(book, 323)
(student, 123)
(history, 78)
…]
(people,128)
(book,323)
(student,123)
(history,78)


map
map
map
map
map
map
[(people, 178)
(book, 312)
(student, 554)
(history, 81)
…]
(people,128)
(people,178)
(book,323)
(book,312)
(student,123)
(student, 554)
(history,78)
(history,81)


map
map
map
map
map
map
[(people, 218)
(book, 53)
(student, 251)
(history, 17)
…]
(people,128)
(people,178)
…
(people,218)
(book,323)
(book,312)
…
(book,53)
(student,123)
(student, 554)
…
(student, 251)
(history,78)
(history,81)
…
(history,17)


map
map
map
map
map
map
[(people, 128)
(book, 323)
(student, 123)
(history, 78)
…]
[(people, 178)
(book, 312)
(student, 554)
(history, 81)
…]
[(people, 218)
(book, 53)
(student, 251)
(history, 17)
…]
Shuffle →
(people,128)
(people,178)
…
(people,218)
(book,323)
(book,312)
…
(book,53)
(student,123)
(student, 554)
…
(student, 251)
(history,78)
(history,81)
…
(history,17)

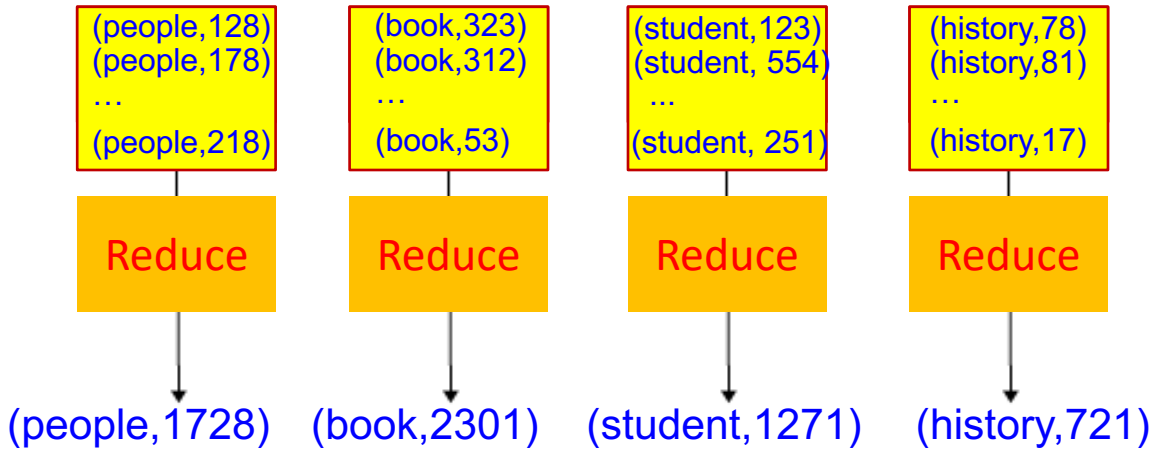

map
map
map
map
map
map
[(people, 128)
(book, 323)
(student, 123)
(history, 78)
…]
[(people, 178)
(book, 312)
(student, 554)
(history, 81)
…]
[(people, 218)
(book, 53)
(student, 251)
(history, 17)
…]
Shuffle →
(people,128)
(people,178)
…
(people,218)
(book,323)
(book,312)
…
(book,53)
(student,123)
(student, 554)
…
(student, 251)
(history,78)
(history,81)
…
(history,17)
Reduce
Reduce
Reduce
Reduce
(people,1728)
(book,2301)
(student,1271)
(history,721)



Map: Count **All words in Each chunk of data**

Reduce: Count the occurrences of **Each word in the Entire data**

REDUCE



- Note: For the sake of simplicity, In this example we only assigned a single word to each machine in Reduce stage. In practice, each machine in Reduce stage will take care of a set of words!

MAP-REDUCE

- In this example:
  - **Map:** Counts **All** words in **Each** chunk of data
  - **Reduce:** Counts the occurrences of **Each** word in the **Entire** dataset

MapReduce Programming Model

- In MapReduce model the Input and Output data is in the form of Key-Value Pairs:
- **Input** : a set of (in_key , in_value) pairs
- **Output**: a set of (out_key , out_value) pairs

- Programmer specifies two functions:

map (in_key, in_value) -> list of (out_key, intermediate_value)

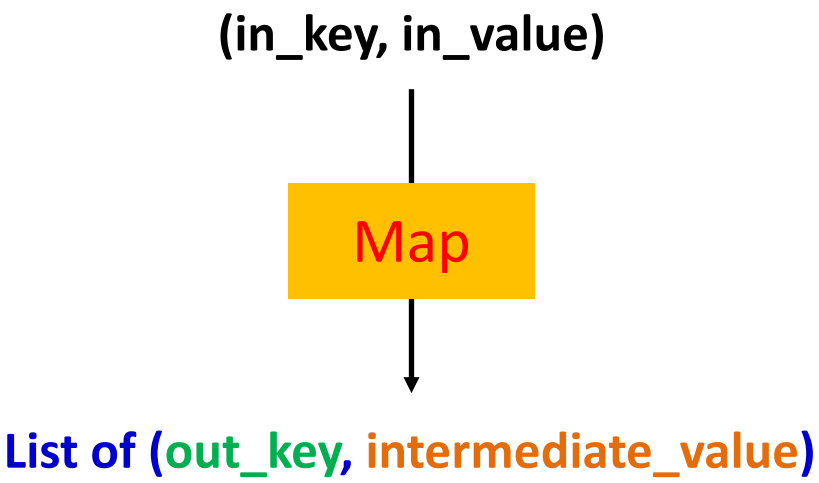
  - Processes input (key,value) pairs
  - Produces set of intermediate pairs

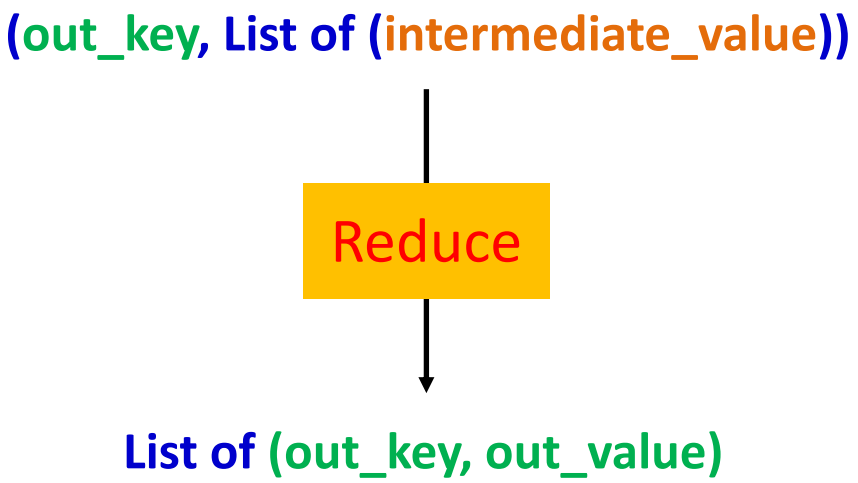
reduce (out_key, list of (intermediate_value)) -> list of (out_key, out_value)

  - Combines all intermediate values for a particular key
  - Produces a set of merged output values (usually just one)

Inputs and Outputs

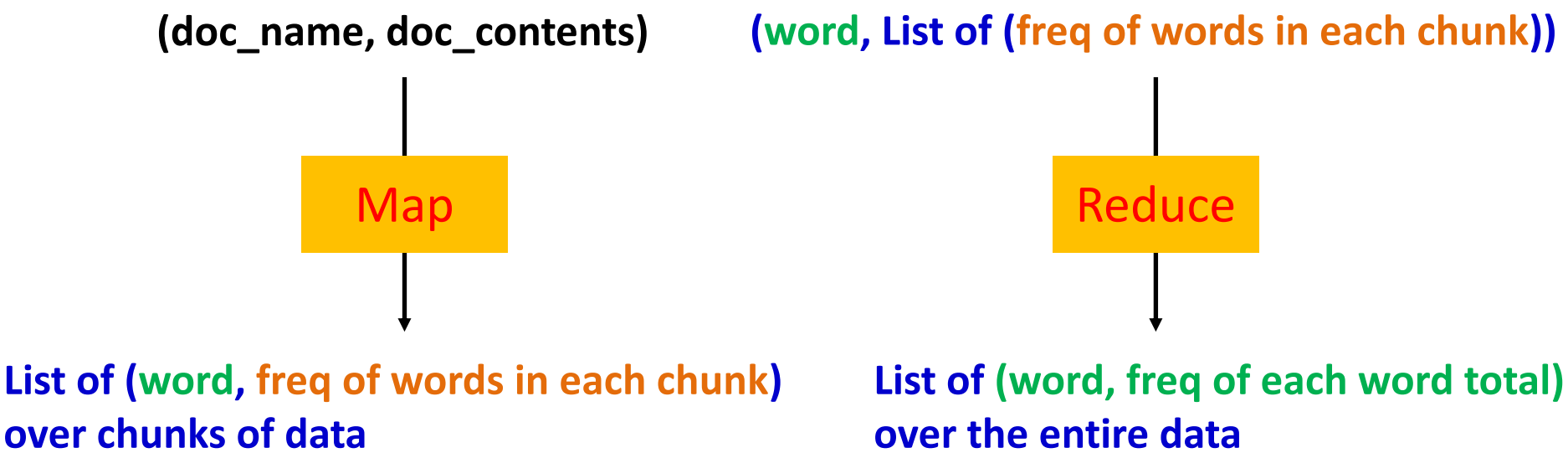
- Input : a set of (in_key,in_value) pairs
- Output: a set of (out_key,out_value) pairs





Inputs and Outputs (Example)

- Input : a set of (in_key,in_value) pairs
- Output: a set of (out_key,out_value) pairs



Example: Compute the overall word frequency across 5M documents

map(String input_key, String input_value):

```
Intermediate_values = {}
for each word w in input_value:
        Intermediate_values[w] += 1
```

reduce(String output_key, Iterator intermediate_values):

```
output_values = 0;
for each v in intermediate_values:
        output_values += v
        Emit(output_key,output_values )
```

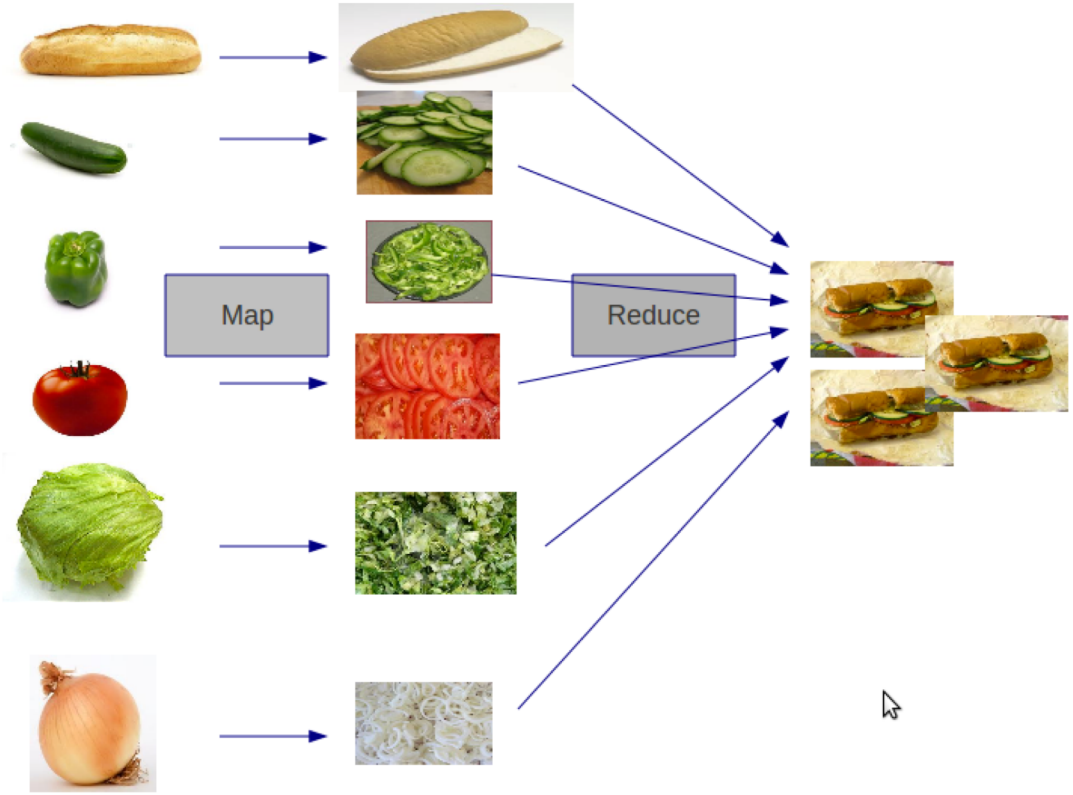
- **Very important to distinguish between (in_key, in_value) and (out_key, out_value)!!!**

```
map(String input_key, String input_value):
        # input_key: document name/id
        # input_value: document contents
Intermediate_values = {}
for each word w in input_value:
        Intermediate_values[w] += 1
```

```
reduce(String output_key, List of [intermediate_values]):
        # output_key: word
        # output_values: freq of each word
output_values = 0;
for each v in intermediate_values:
        output_values += v
return(output_key,output_values )
```

Some Notes about Map Reduce

- Everything is in the form of **key-value** pairs!
- In map stage, **parallelism** is achieved since different parts of data can be processed by different machines simultaneously.
- In reduce stage, **parallelism** is achieved as reducers operating on different keys simultaneously.
- Mappers manipulate the keys, but reducers do not usually change the keys.
- All mappers need to finish before reducers can begin.
- A map-reduce program may consist of several rounds of different map and reduce functions.



Thank You!

Questions?