



Introduction to Data Science

(Lecture 22)

Dr. Mohammad Pourhomayoun

Assistant Professor

Computer Science Department

California State University, Los Angeles







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 uses **computer science (machine learning & big data)** and **statistics** to analyze and interpret biological data¹.



1: Wikipedia



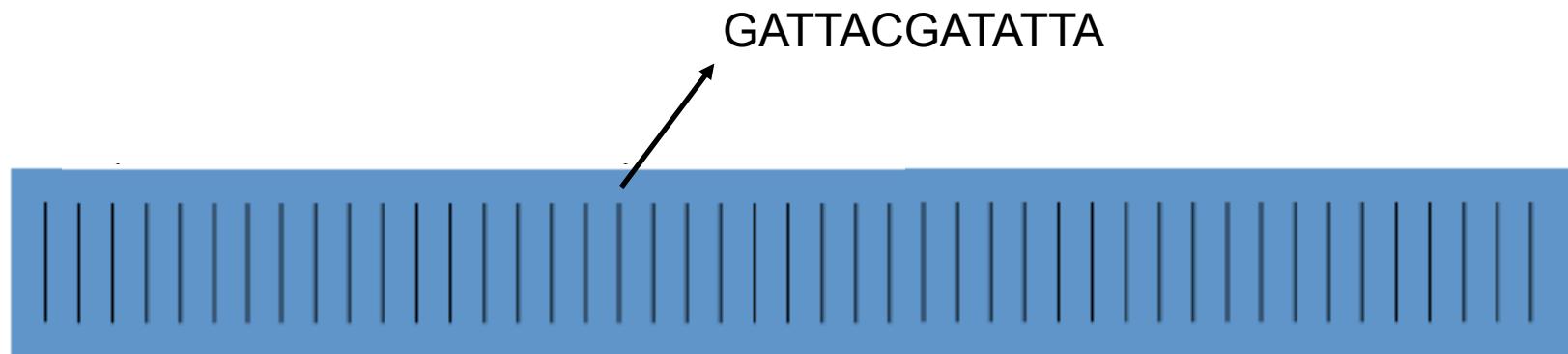
One Example to Start ...

- **Genomics (more info):**
 - 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

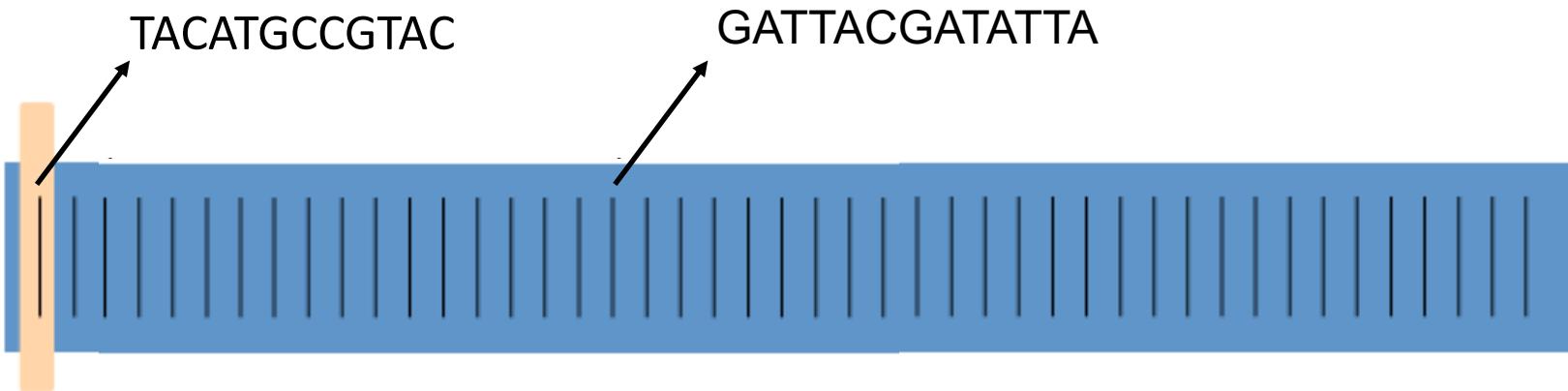
- Given a set of sequences, Find all sequences equal to:
 - “GATTACGATATTAA”



* Example from Bill Howe, University of Washington



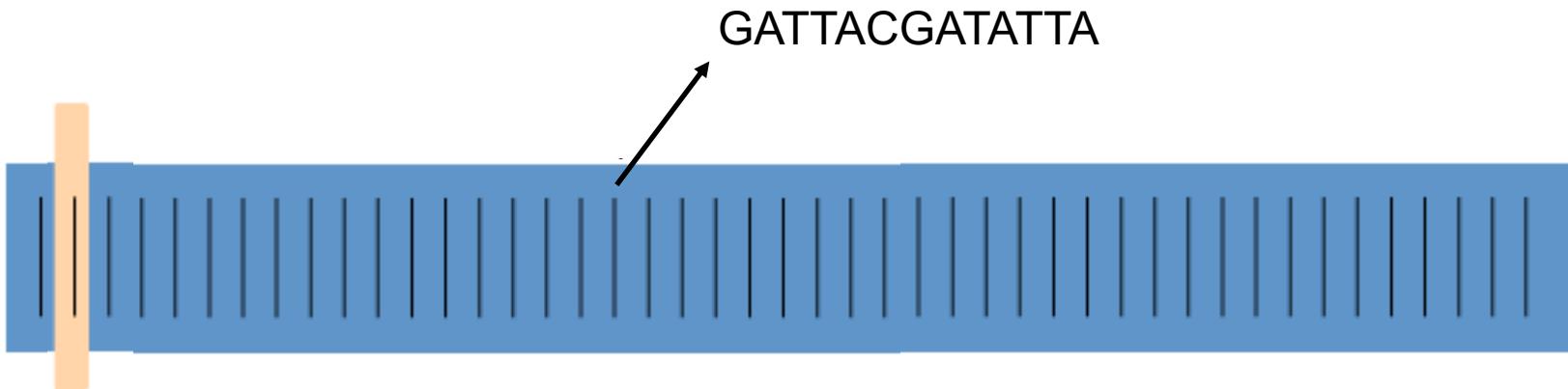
Example: Find Matching DNA



Time 0:

Is (TACATGCCGTAC = GATTACGATATTA)? → No

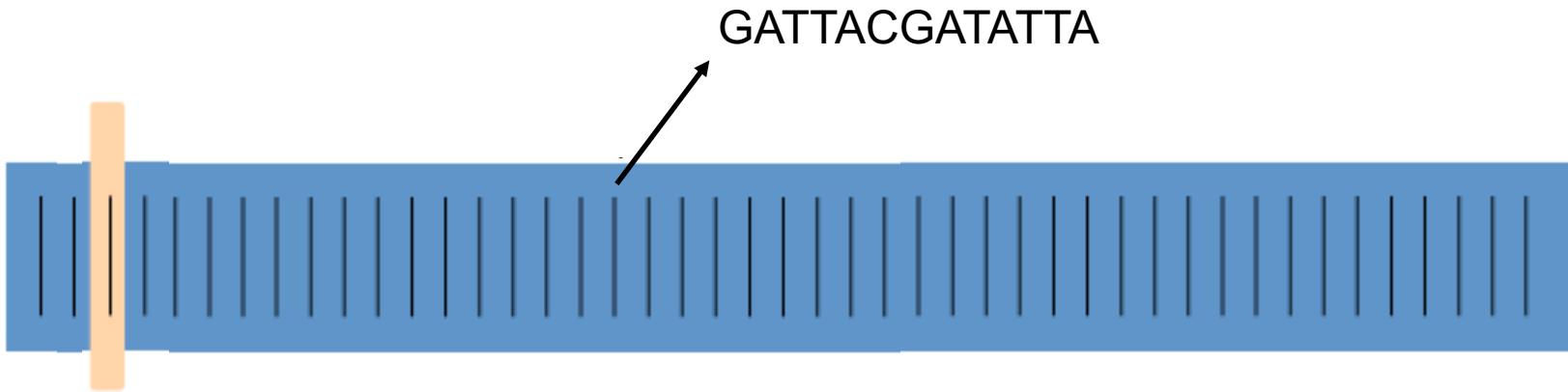
Example: Find Matching DNA



Time 1:

Is (TAATGCCGTACGC = GATTACGATATTA)? → No

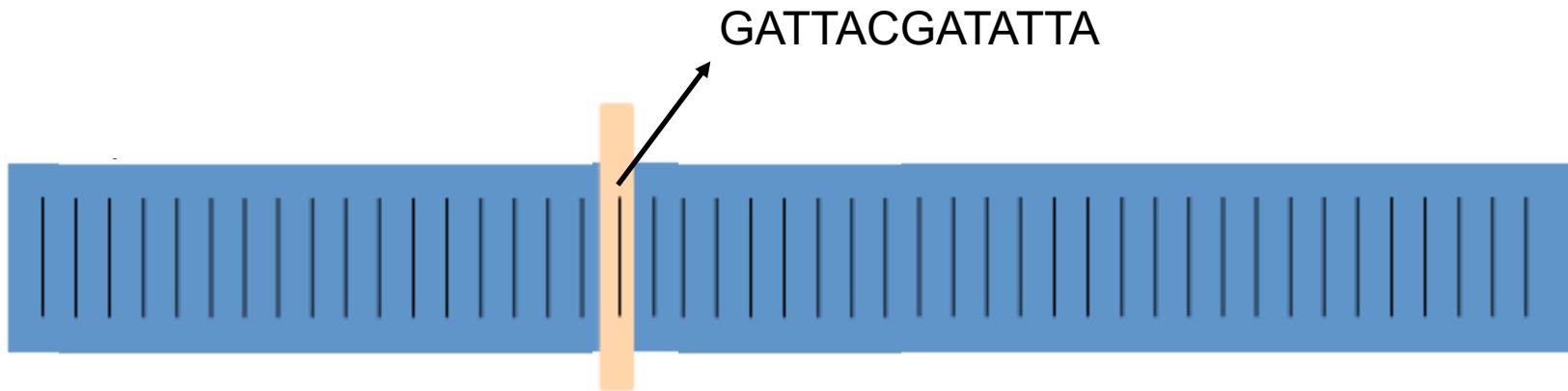
Example: Find Matching DNA



Time 2:

Is (TACGTAGCATGC = GATTACGATATTAA)? → No

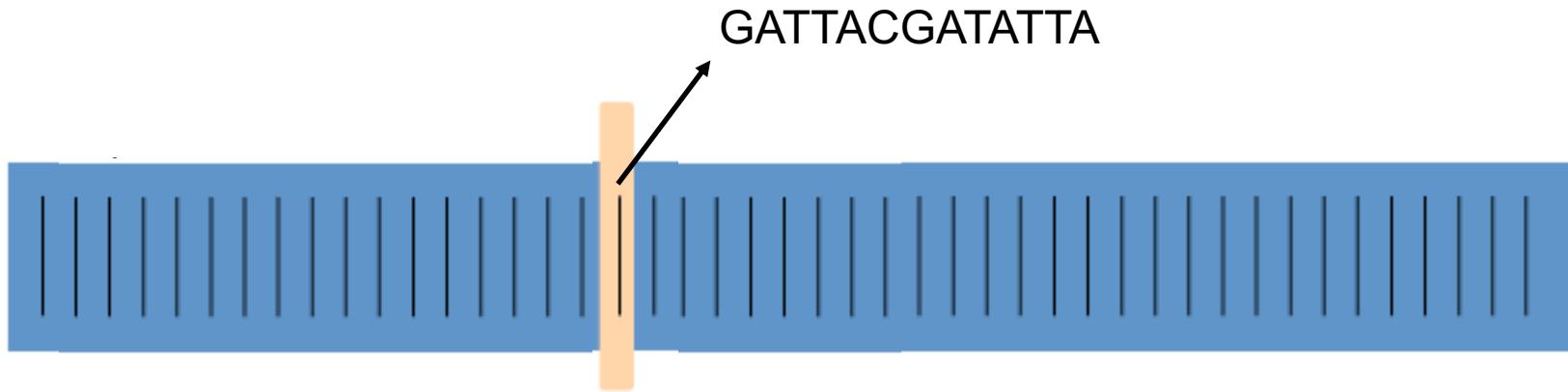
Example: Find Matching DNA



Time 17:

Is (ATTACGATATTA = GATTACGATATTA)? → Yes

Example: Find Matching DNA

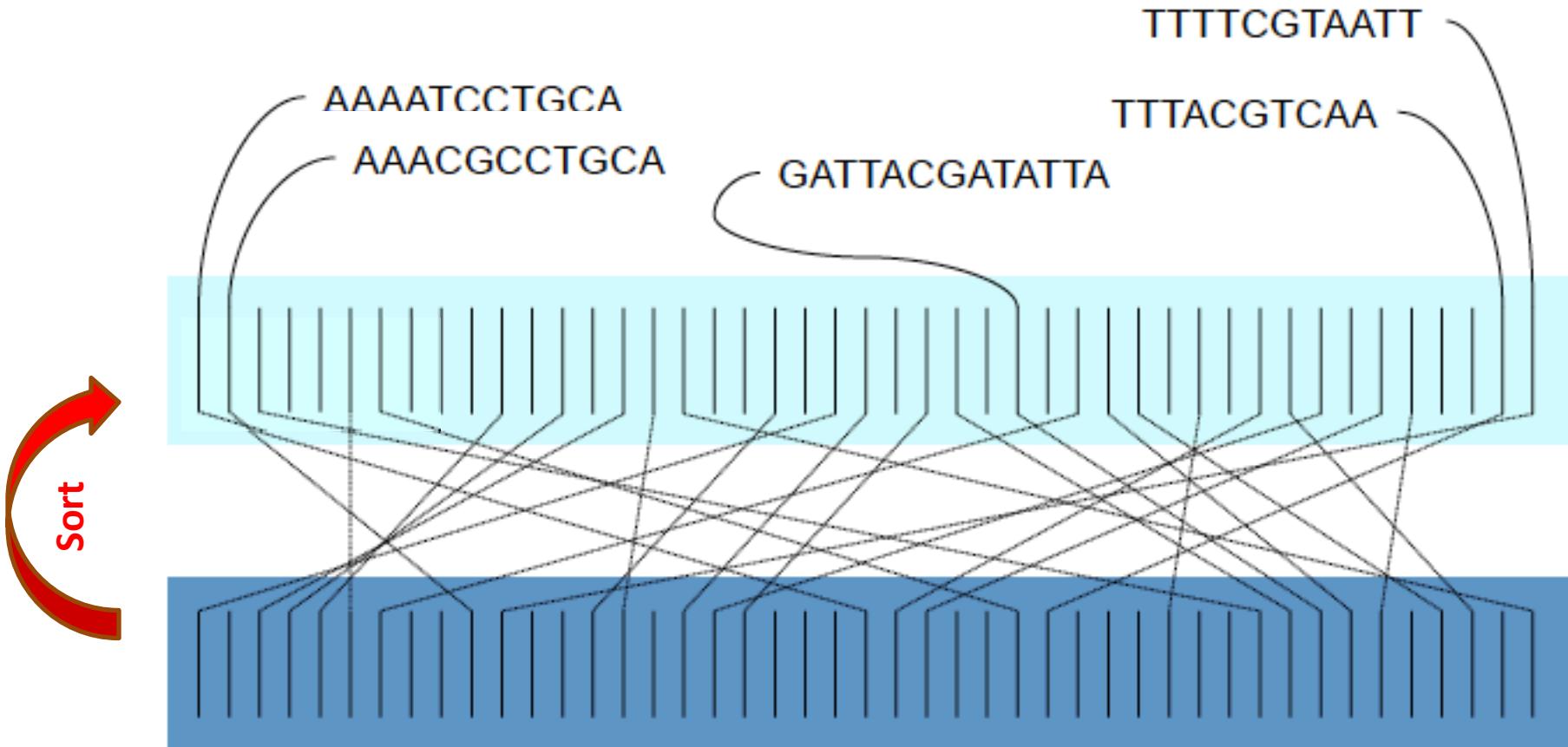


N records, N comparisons

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:

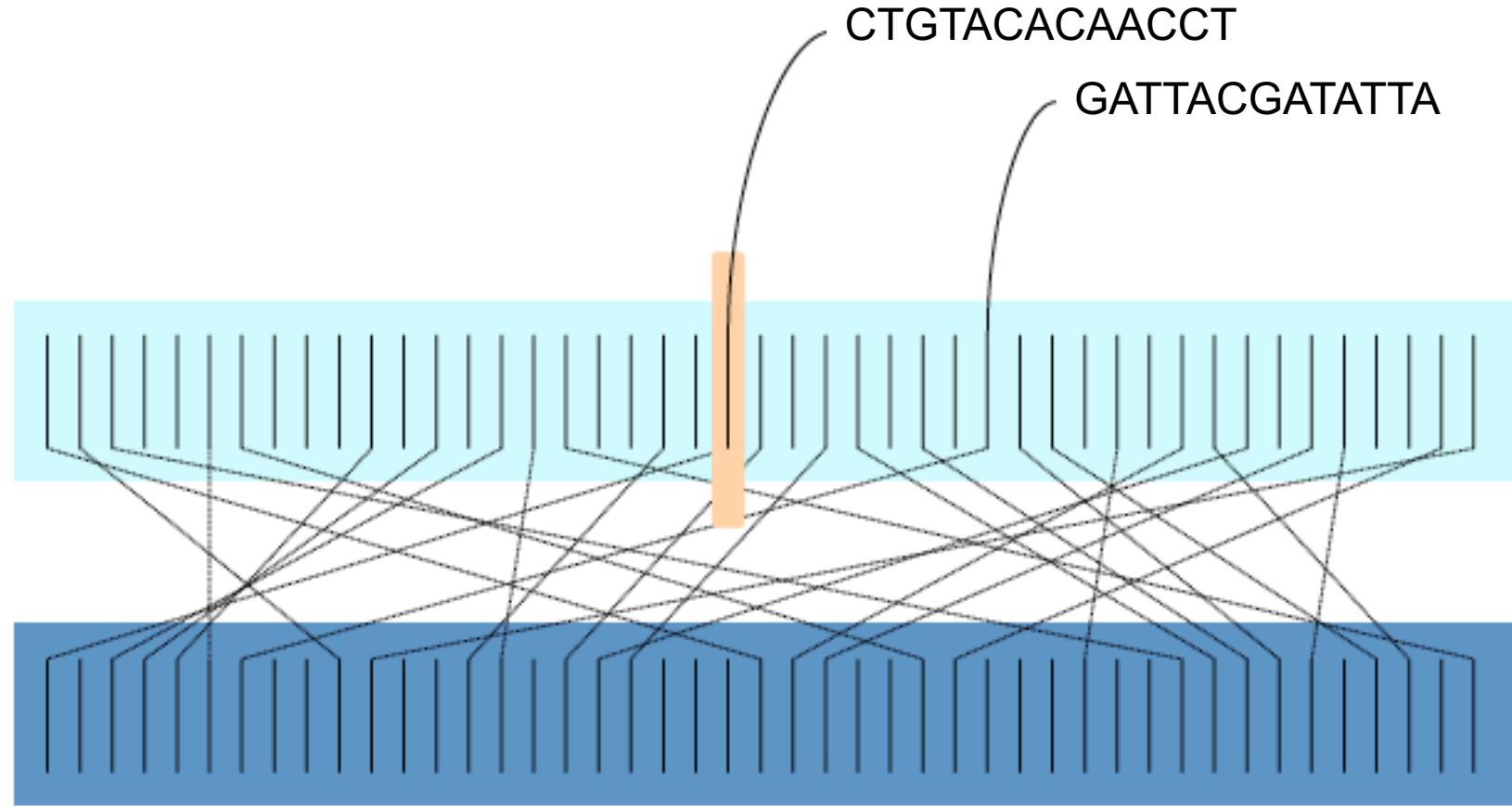


* Example from Bill Howe, University of Washington

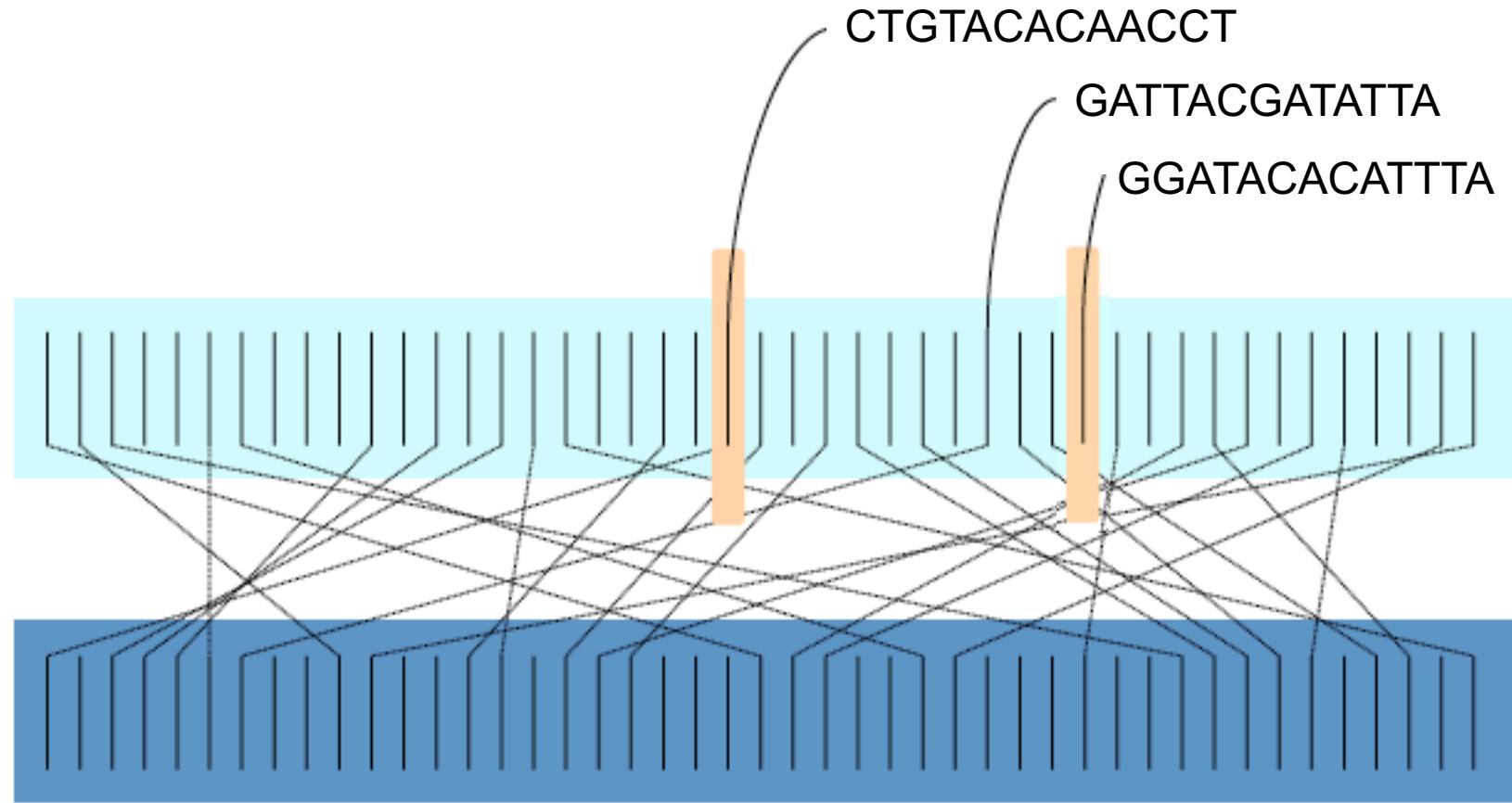


- Start at the middle:

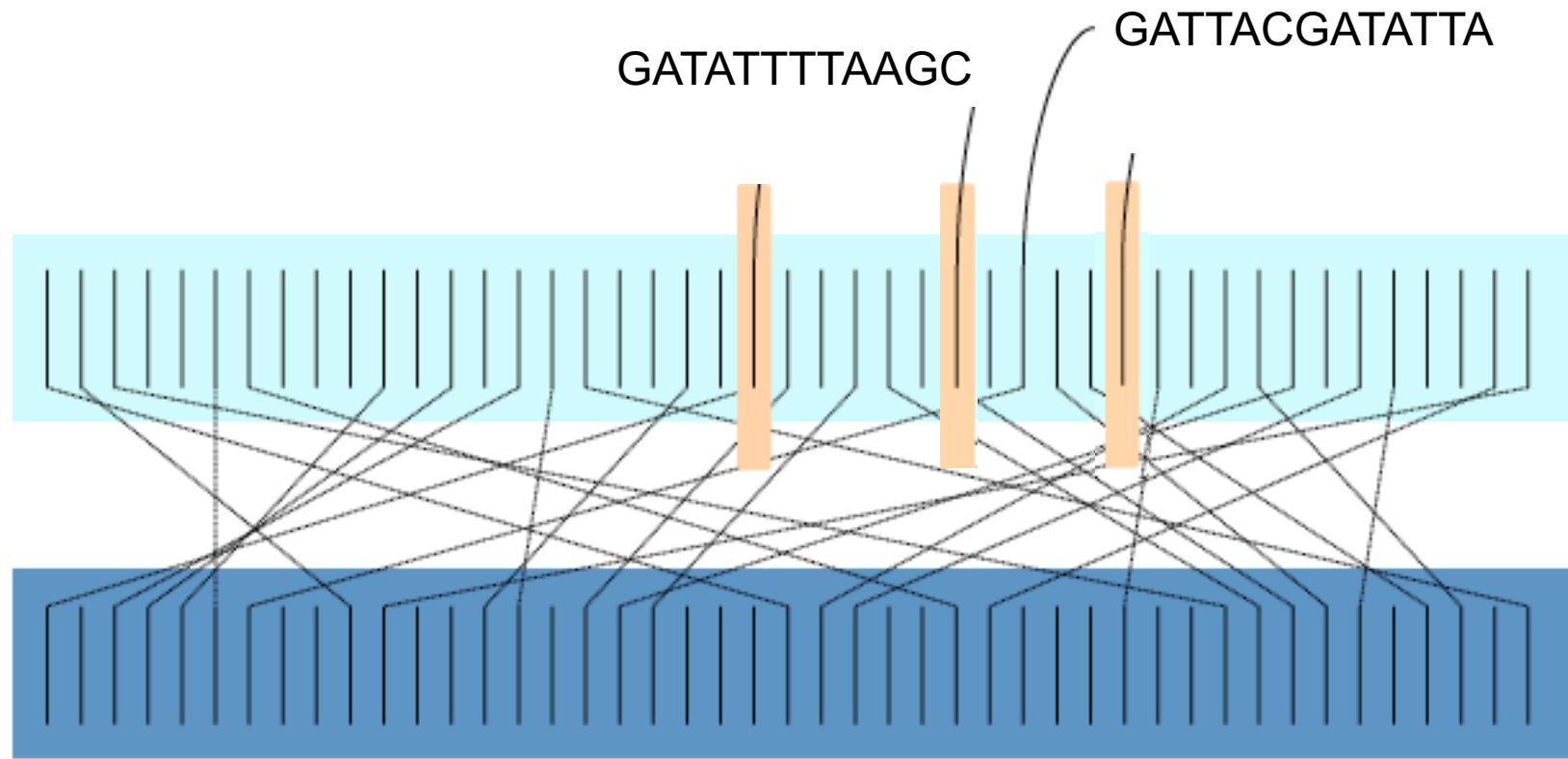
CTGTACACAAACCT < **GATTACGATATTA** → Right Side



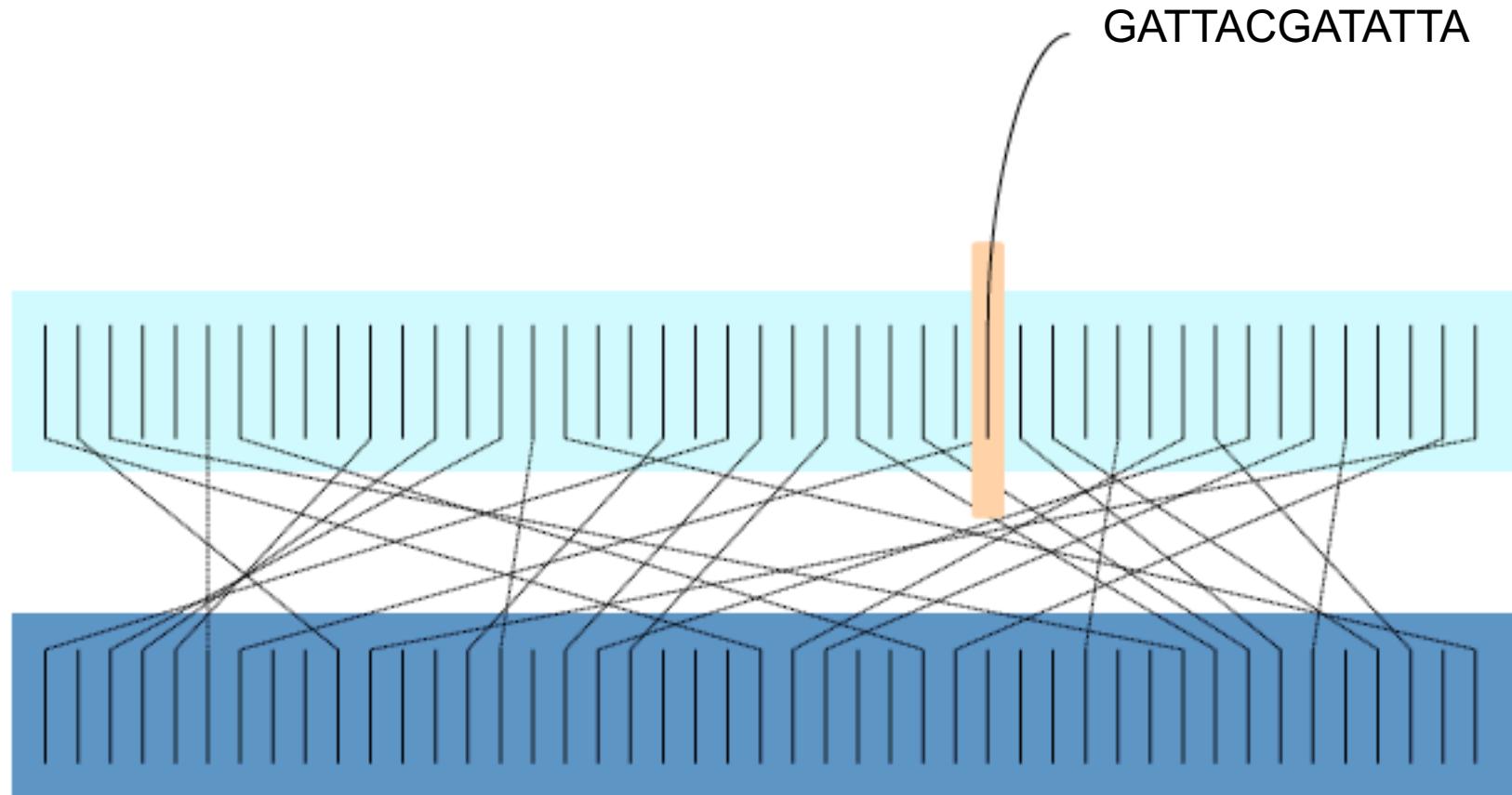
- GGATACACATTAA > **GATTACGATATTA** → Left Side



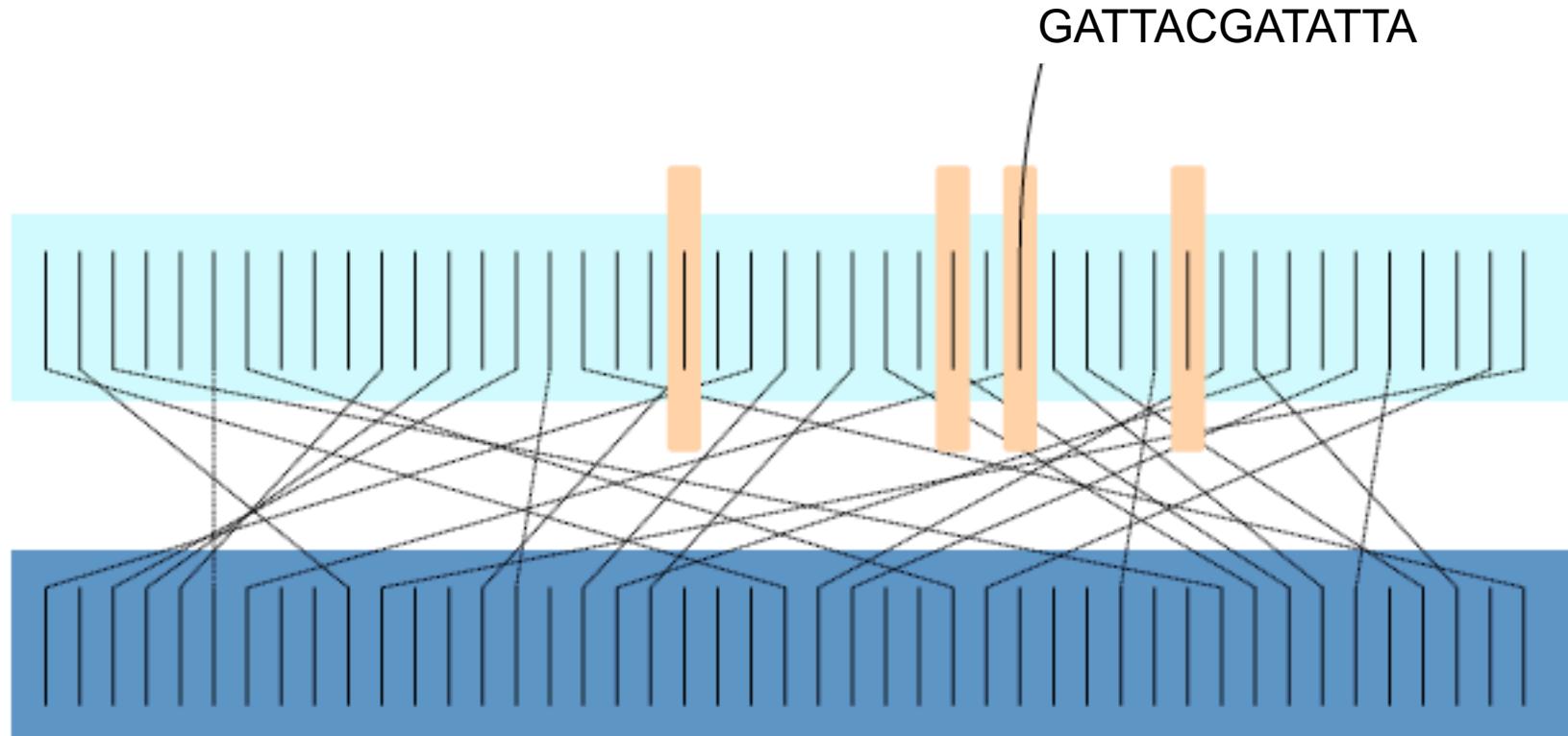
- GATATTAAAGC < **GATTACGATATTA** → Right Side



- GATTACGATATTAA = **GATTACGATATTAA** → Match!!!



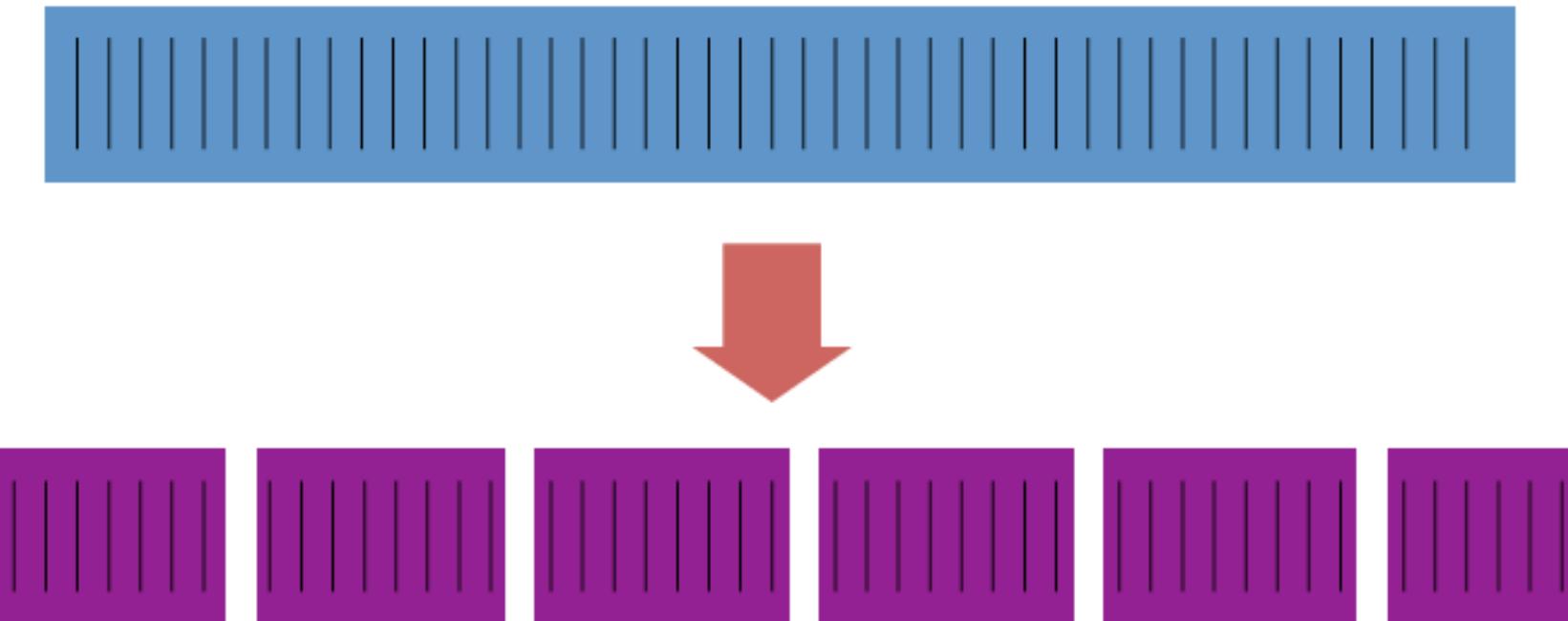
- 40 records, only 4 comparisons
- N records, $\log(N)$ comparisons
- This algorithm is $O(\log(N)) \rightarrow$ Much Better Scalability!



Can we do any better?

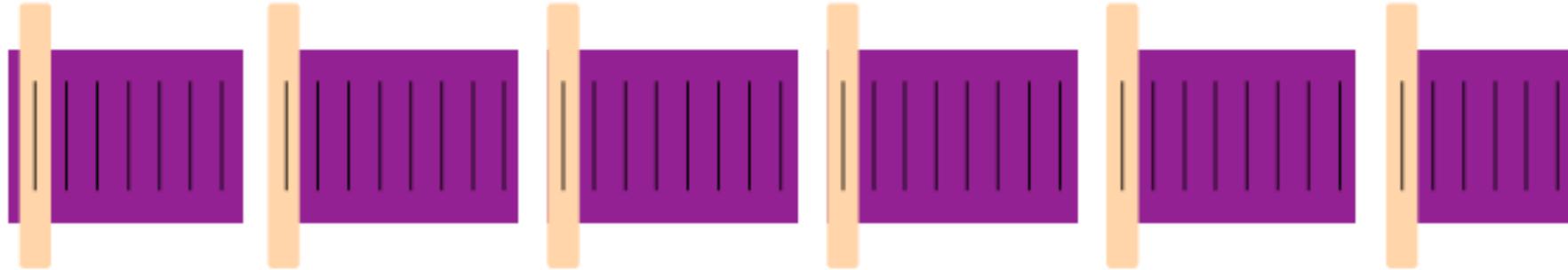
- **Can we do any better?**
- Yes! Because the processing of each comparison is independent from others, so we can split a big task to several smaller tasks and process them in parallel!





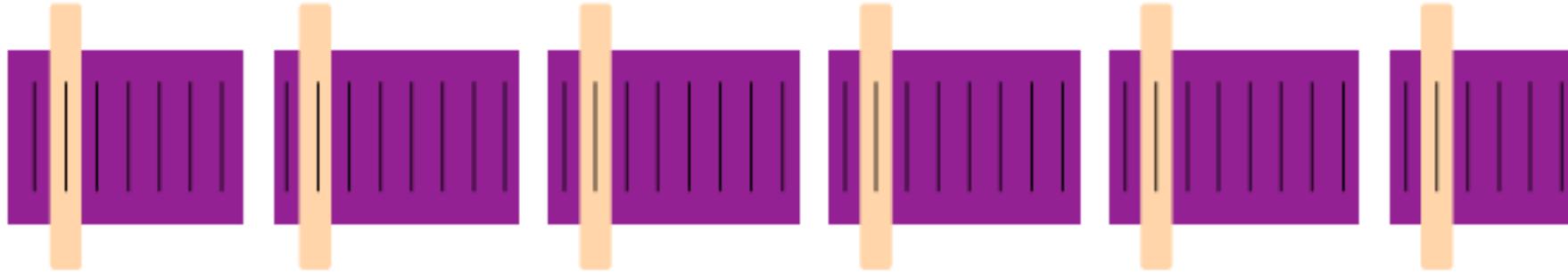
* Example from Bill Howe, University of Washington





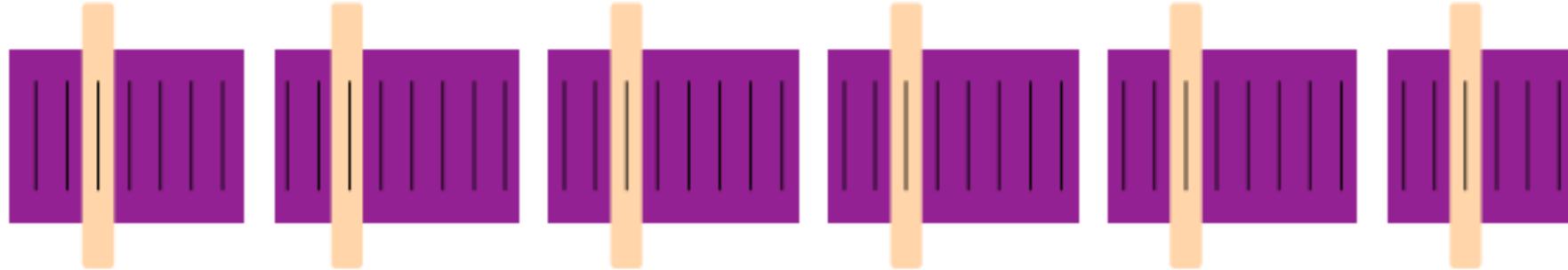
Time 0





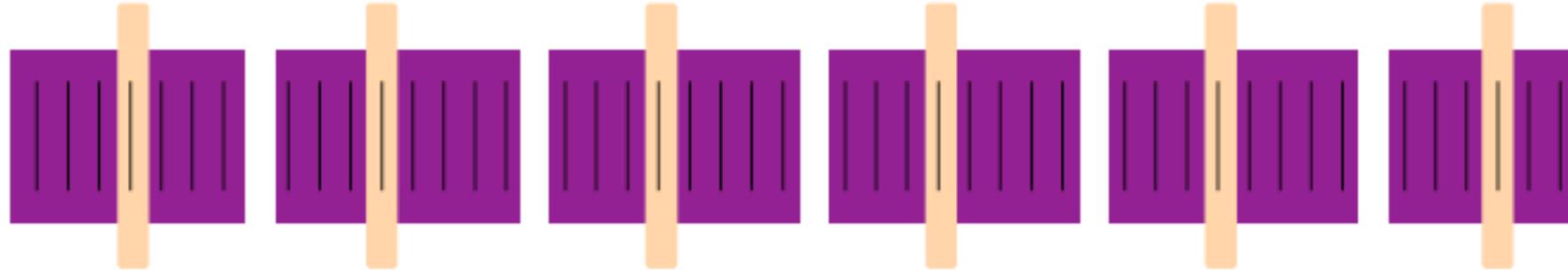
Time 1





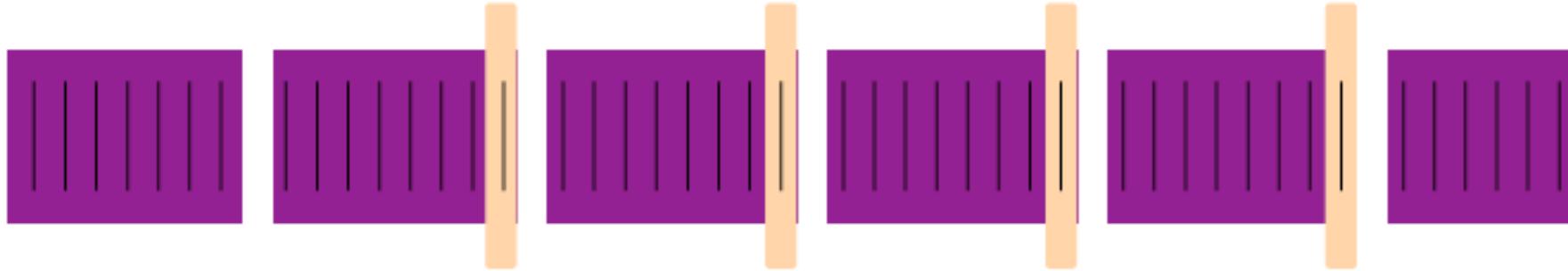
Time 2





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?

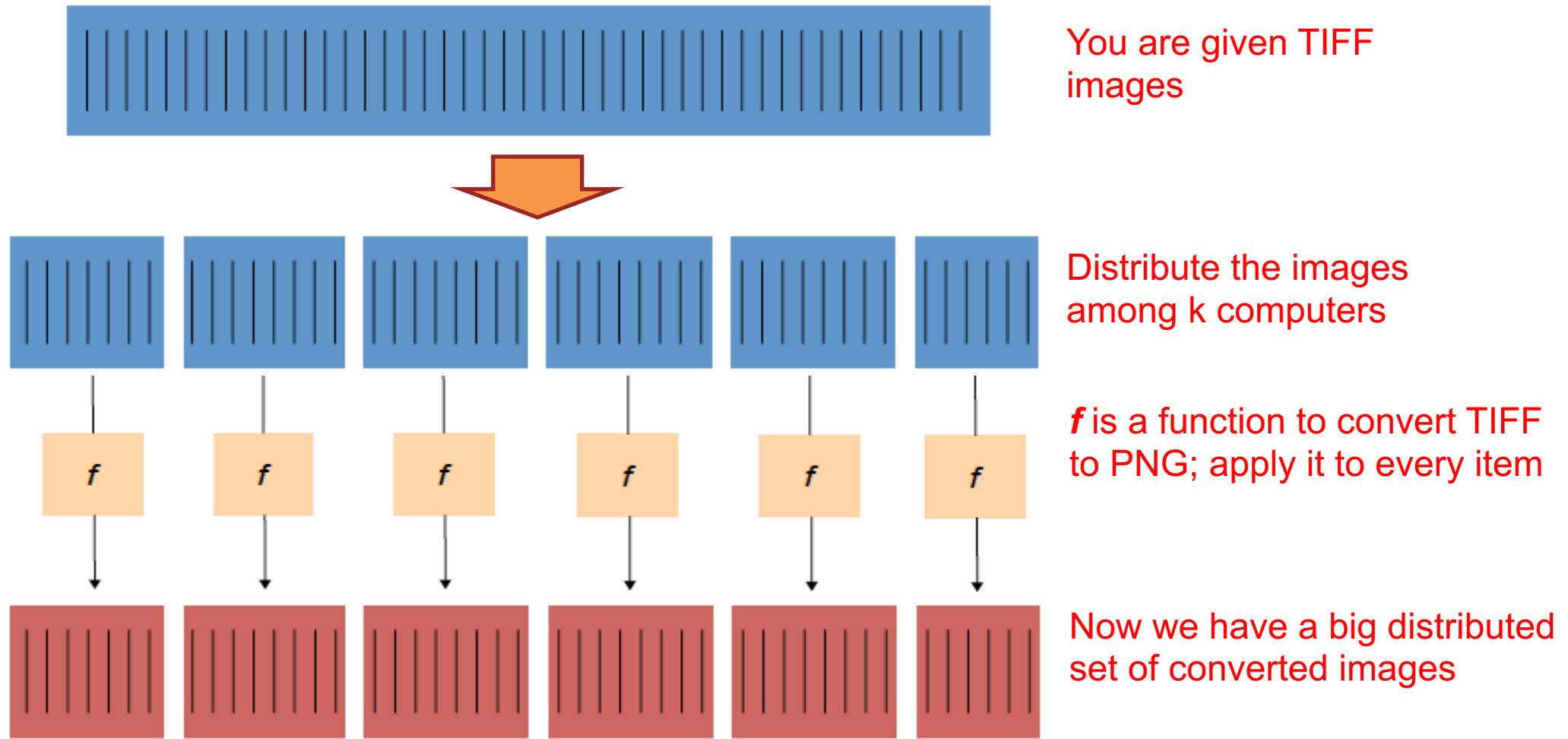
Map-Reduce

- **Map-Reduce** is a **programming model** for processing and generating **big data sets** with a **parallel and distributed** algorithm.
- **map function** processes input **key/value pairs** to generate a set of intermediate **key/value pairs**.
- **reduce function** merges all intermediate values associated with the same intermediate key.

[Ref]: Dean, Jeffrey & Ghemawat, Sanjay. (2004). MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM.

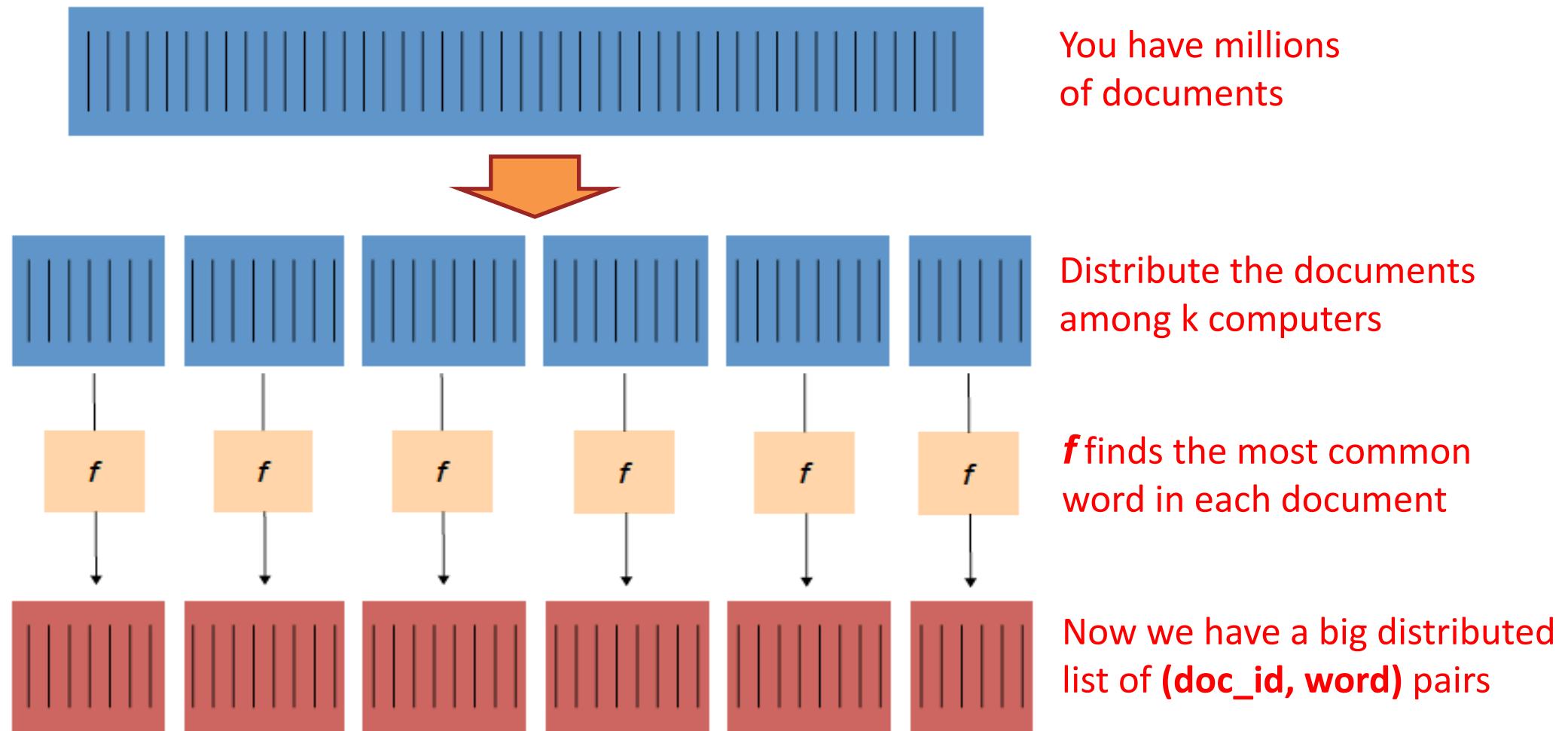


Example1: Converting 405k TIFF Images to PNG¹



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

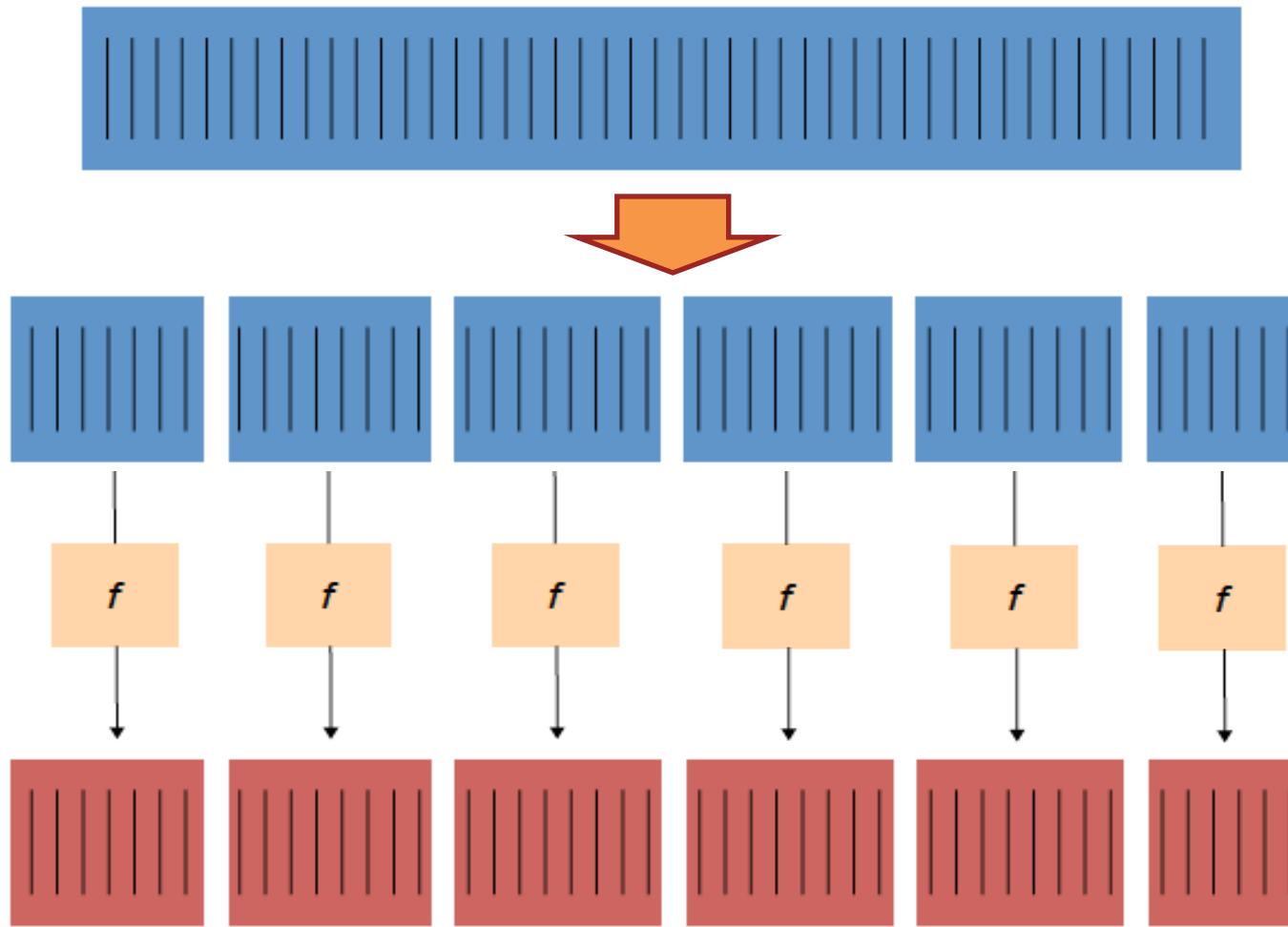
Example2: Find the most common word in each document



* Example from Bill Howe, University of Washington



Example3: Compute overall word frequency across 5M docs



You have millions
of documents

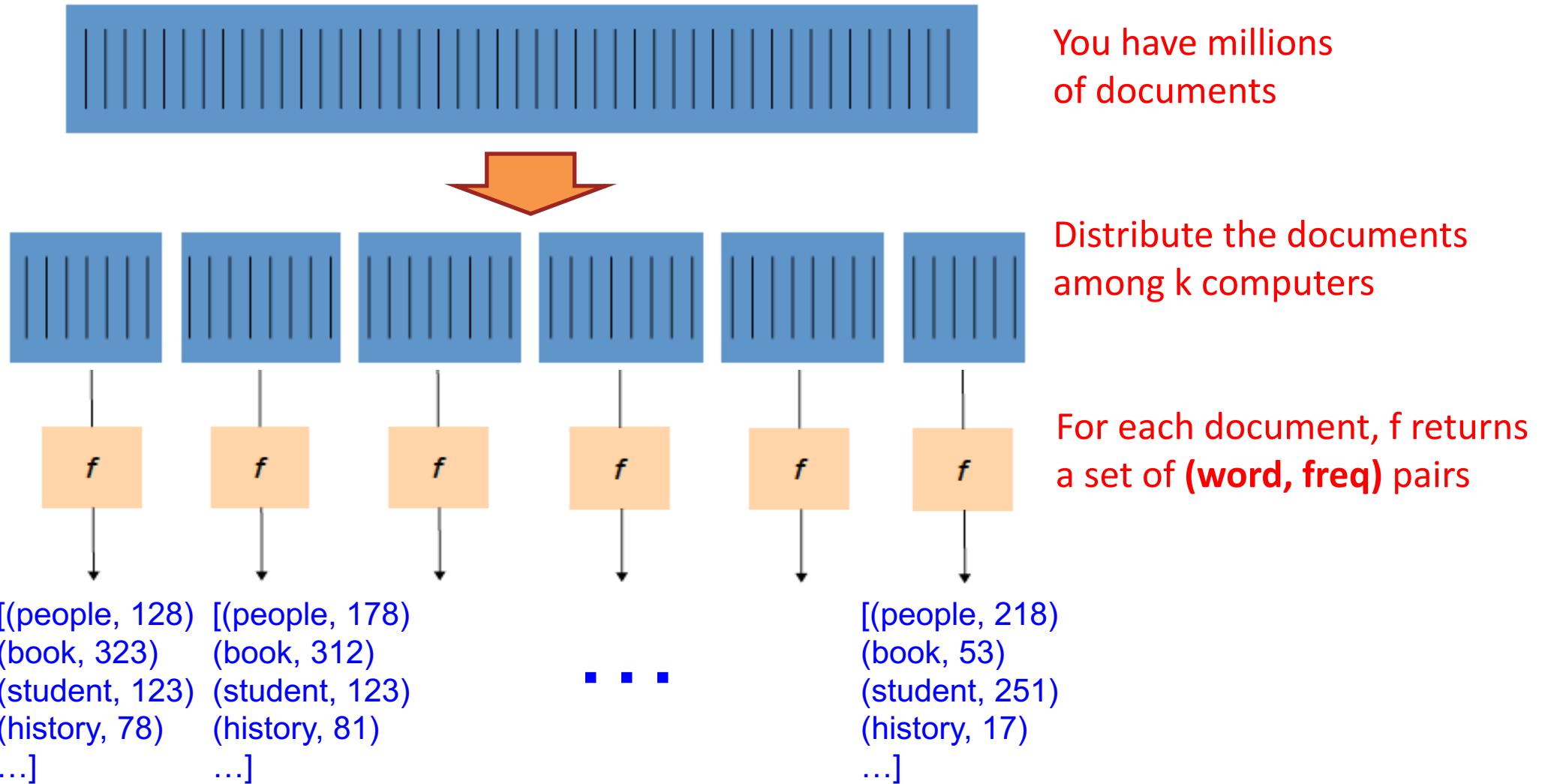
Distribute the documents
among k computers

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 Example3 : Compute overall word frequency across 5M docs

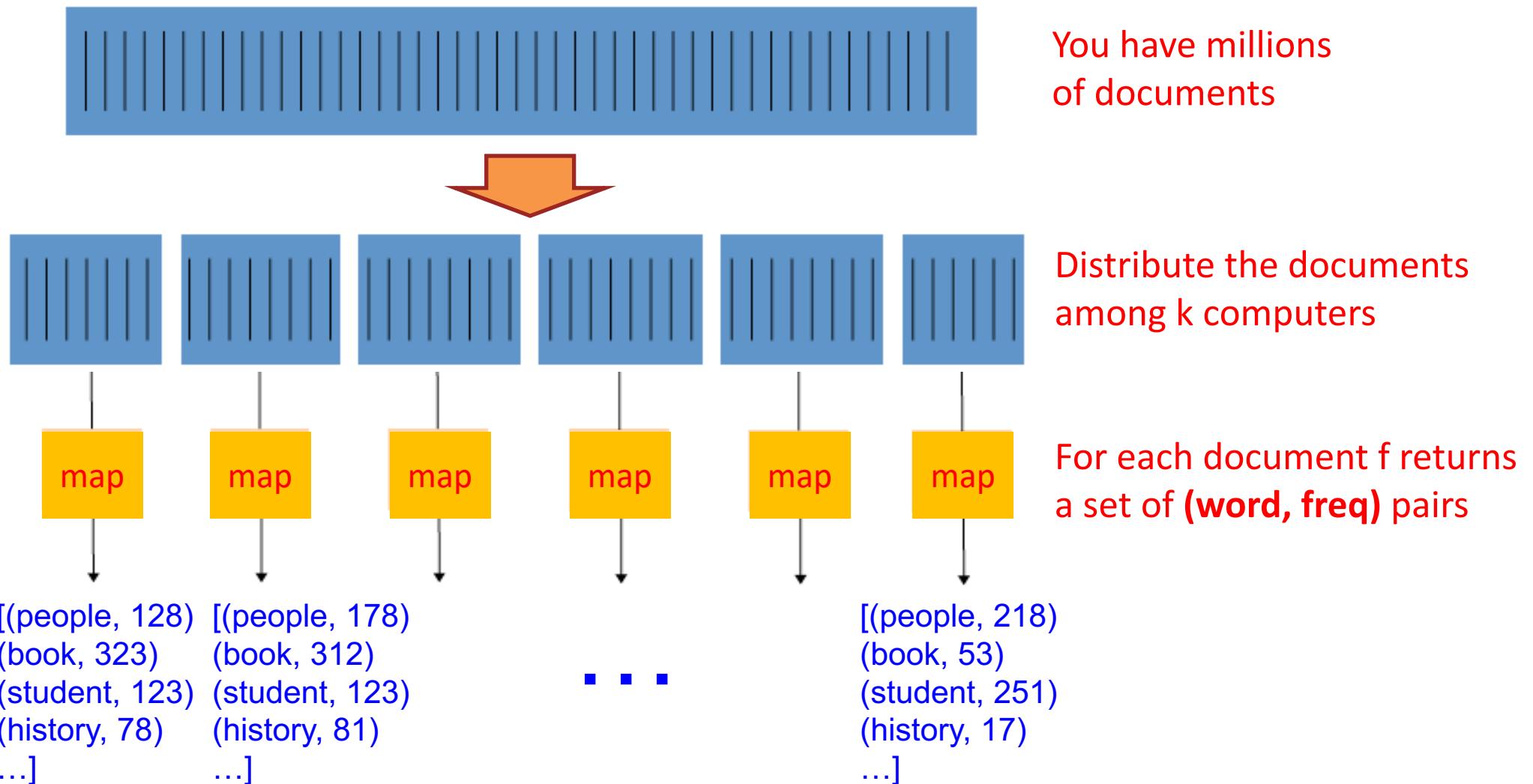


MAP

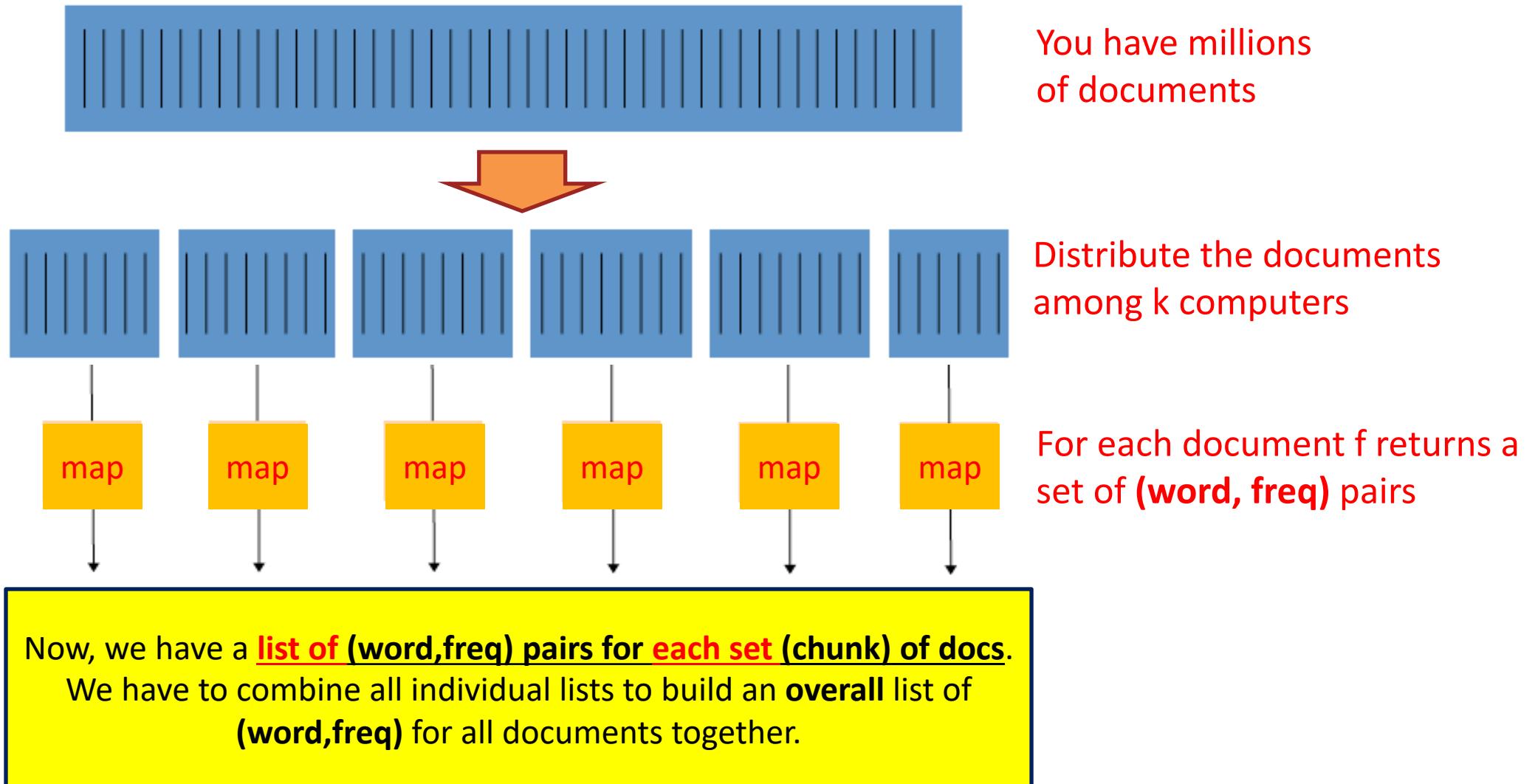
- **map function** processes input **key/value pairs** to generate a set of **intermediate key/value pairs**.
- In the 1st example, function “f” **maps** a **TIFF image to a PNG image**.
- In the 2nd example, function “f” **maps** a **document to its most common word**.
- In the 3rd example, function “f” **maps** a **set of documents to its word frequencies**.

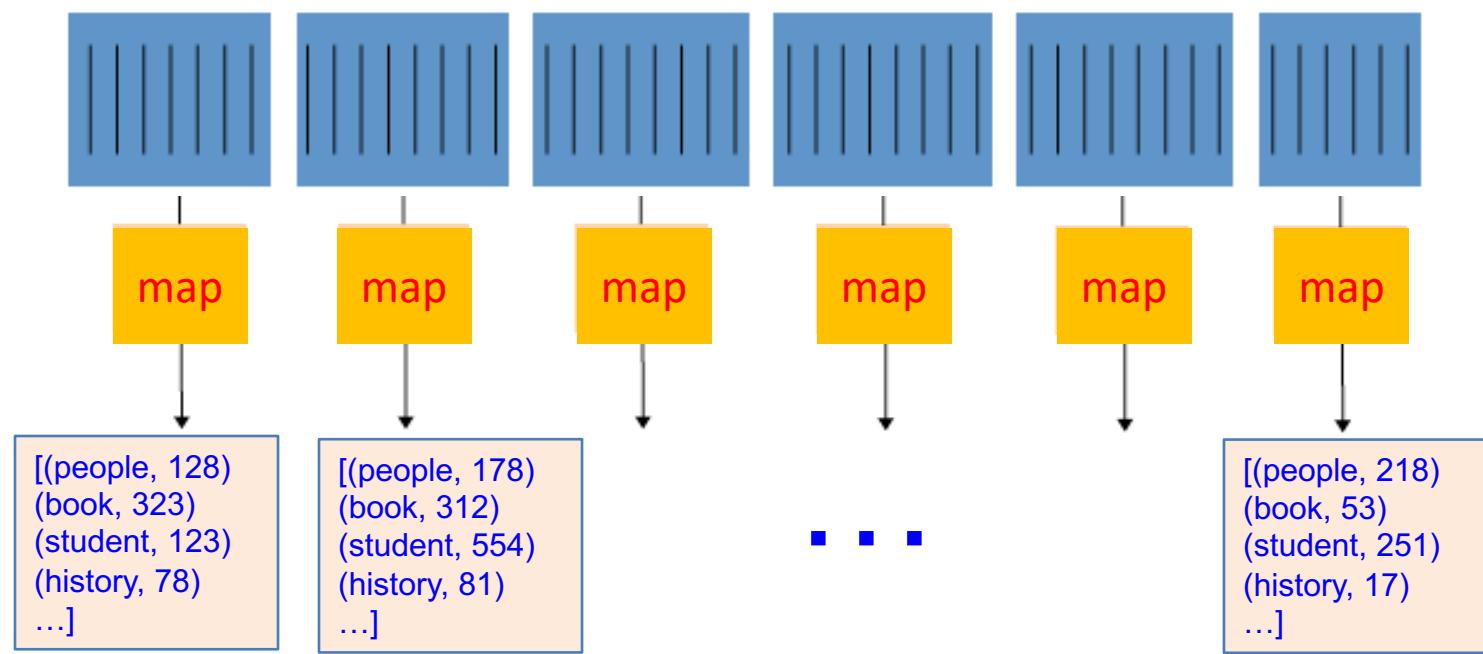


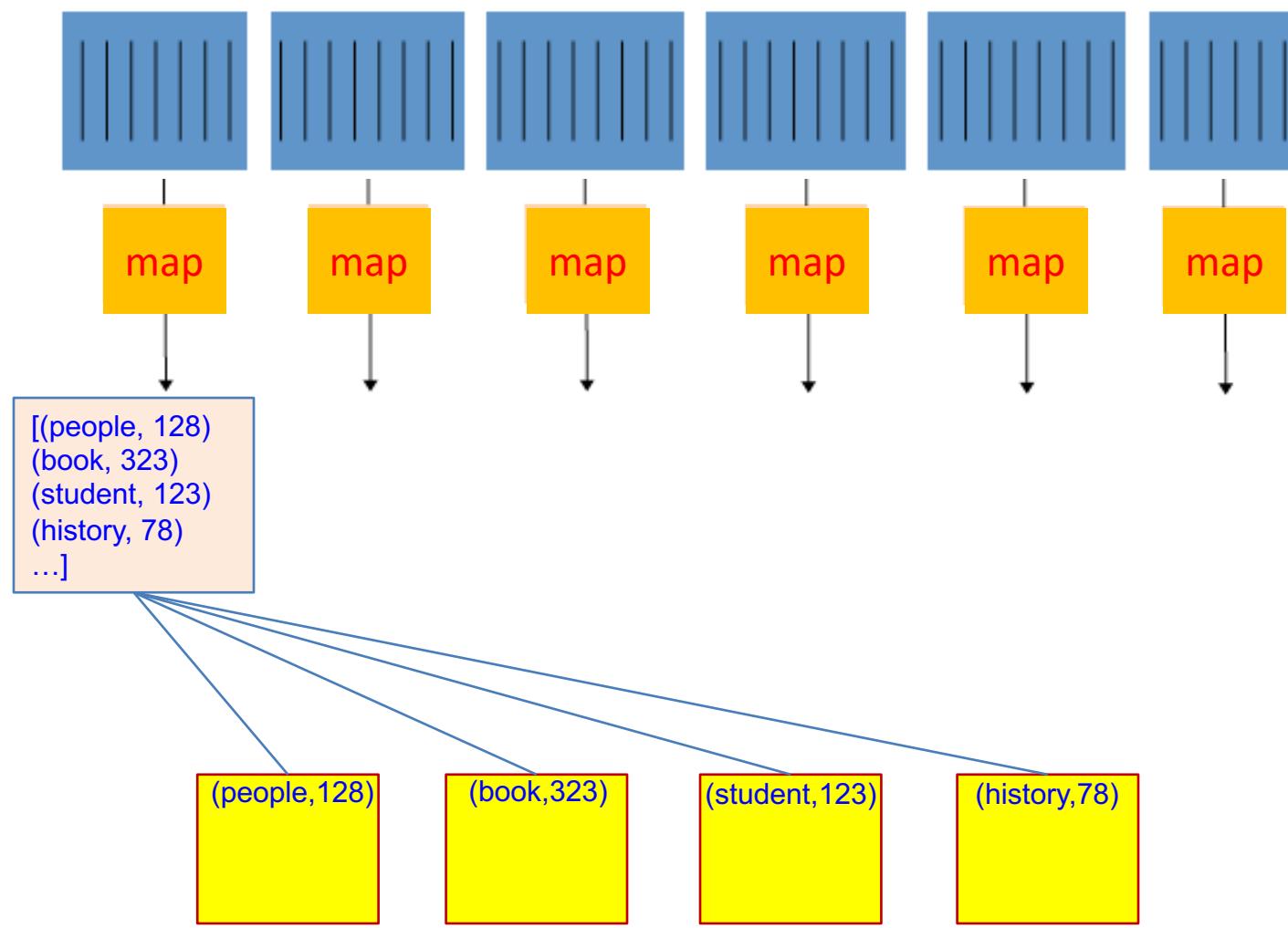
Continue Example3 : Compute overall word frequency across 5M docs



Continue Example3 : Compute overall word frequency across 5M docs

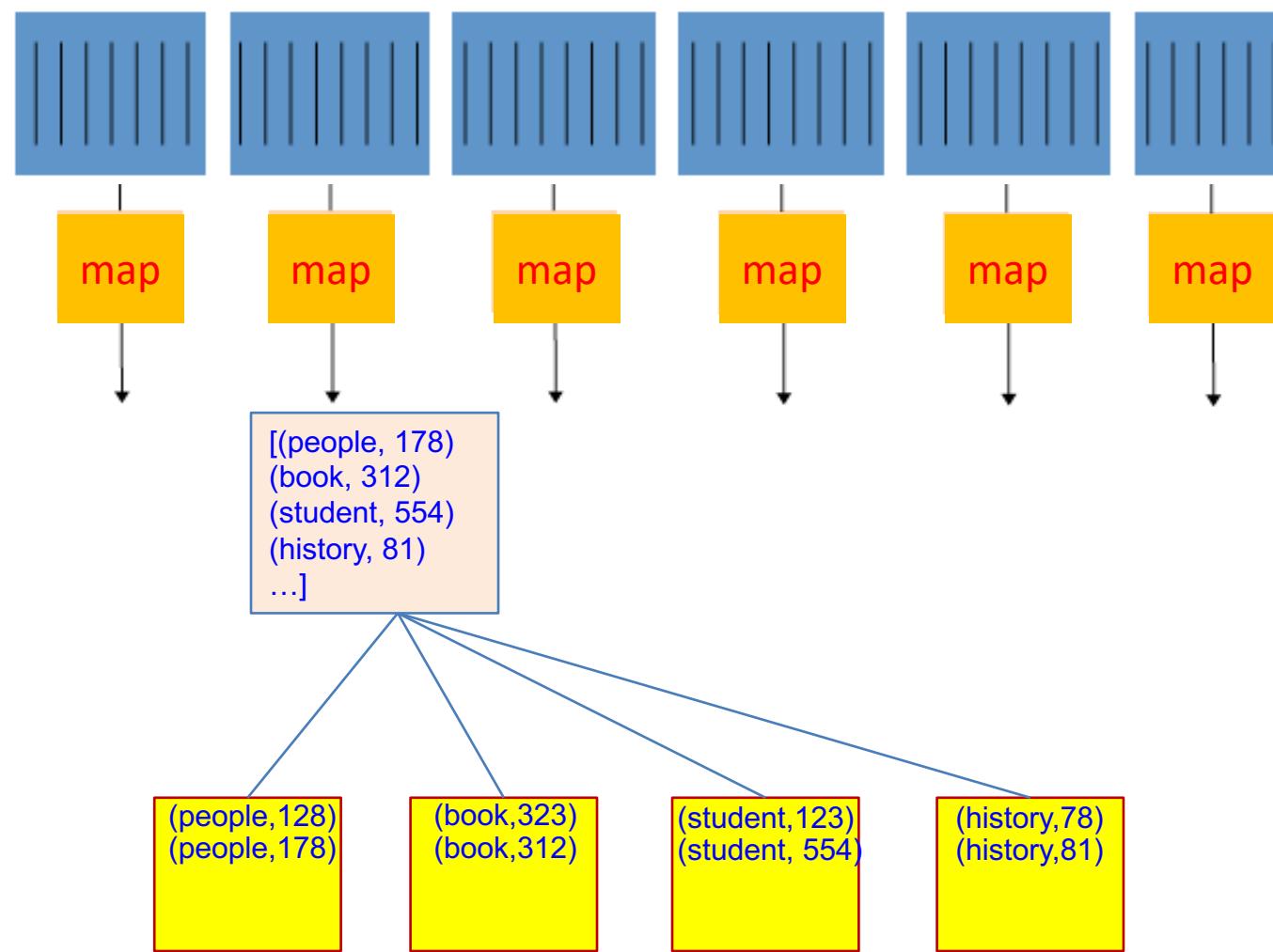


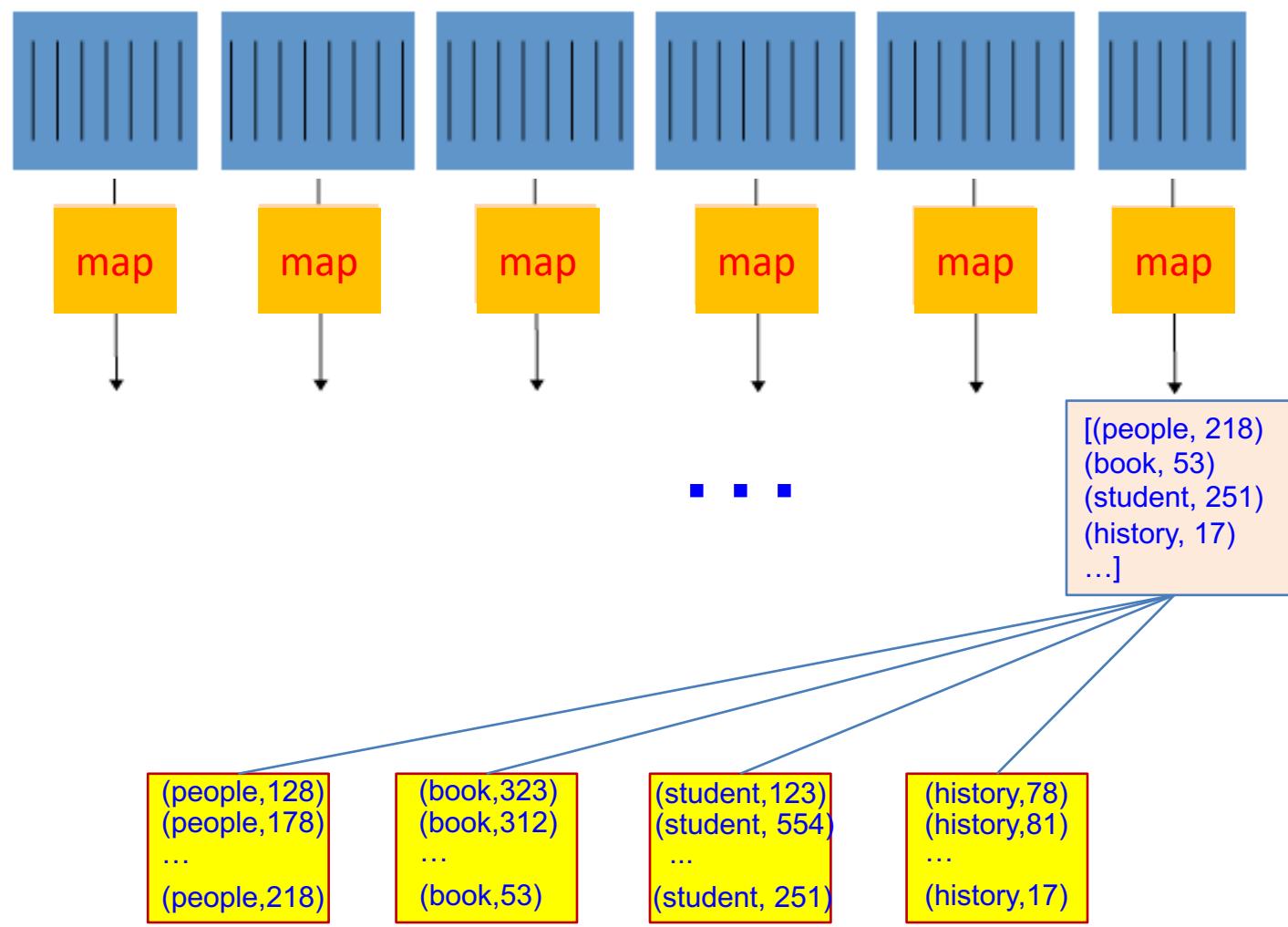


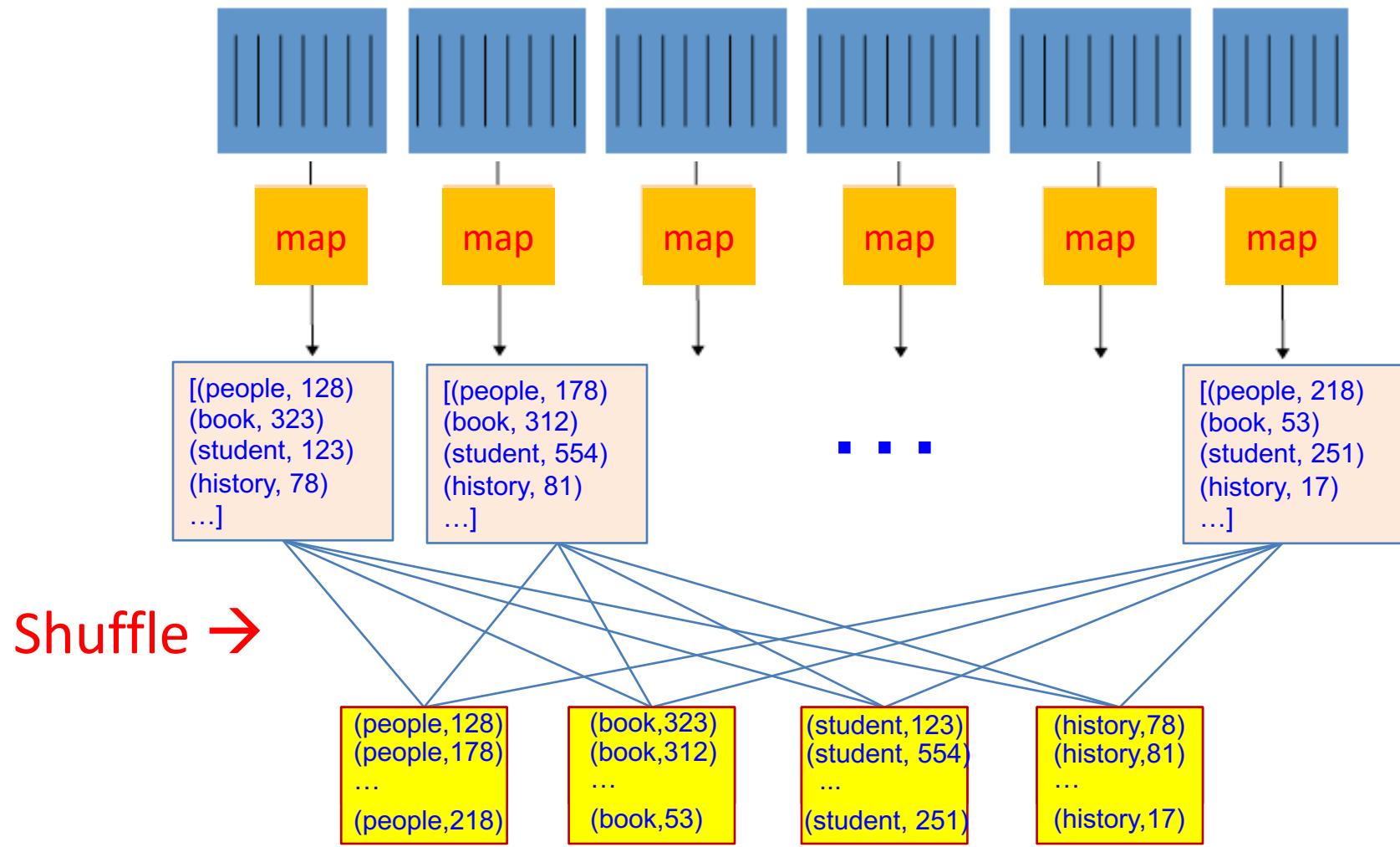


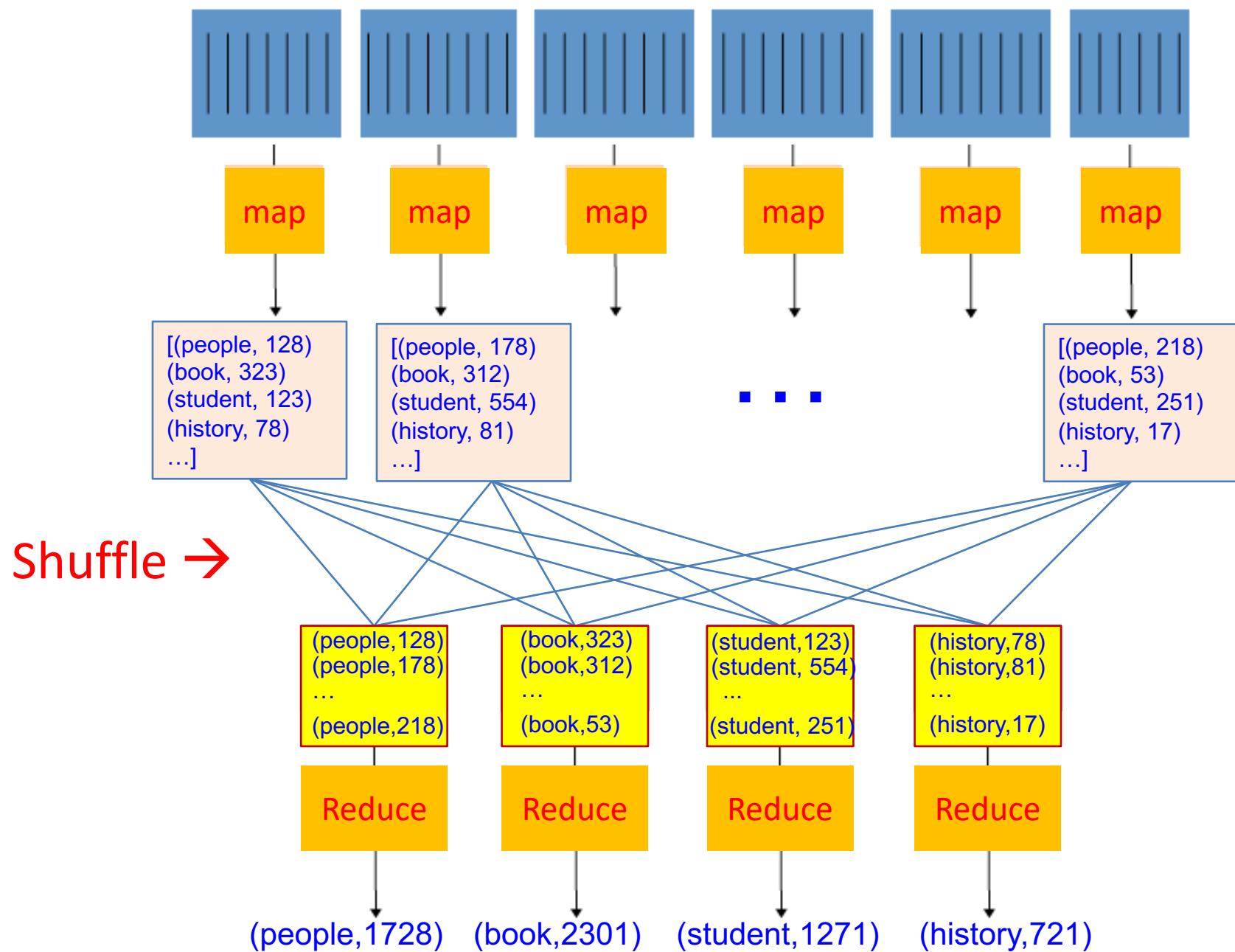
* Example from Bill Howe, University of Washington

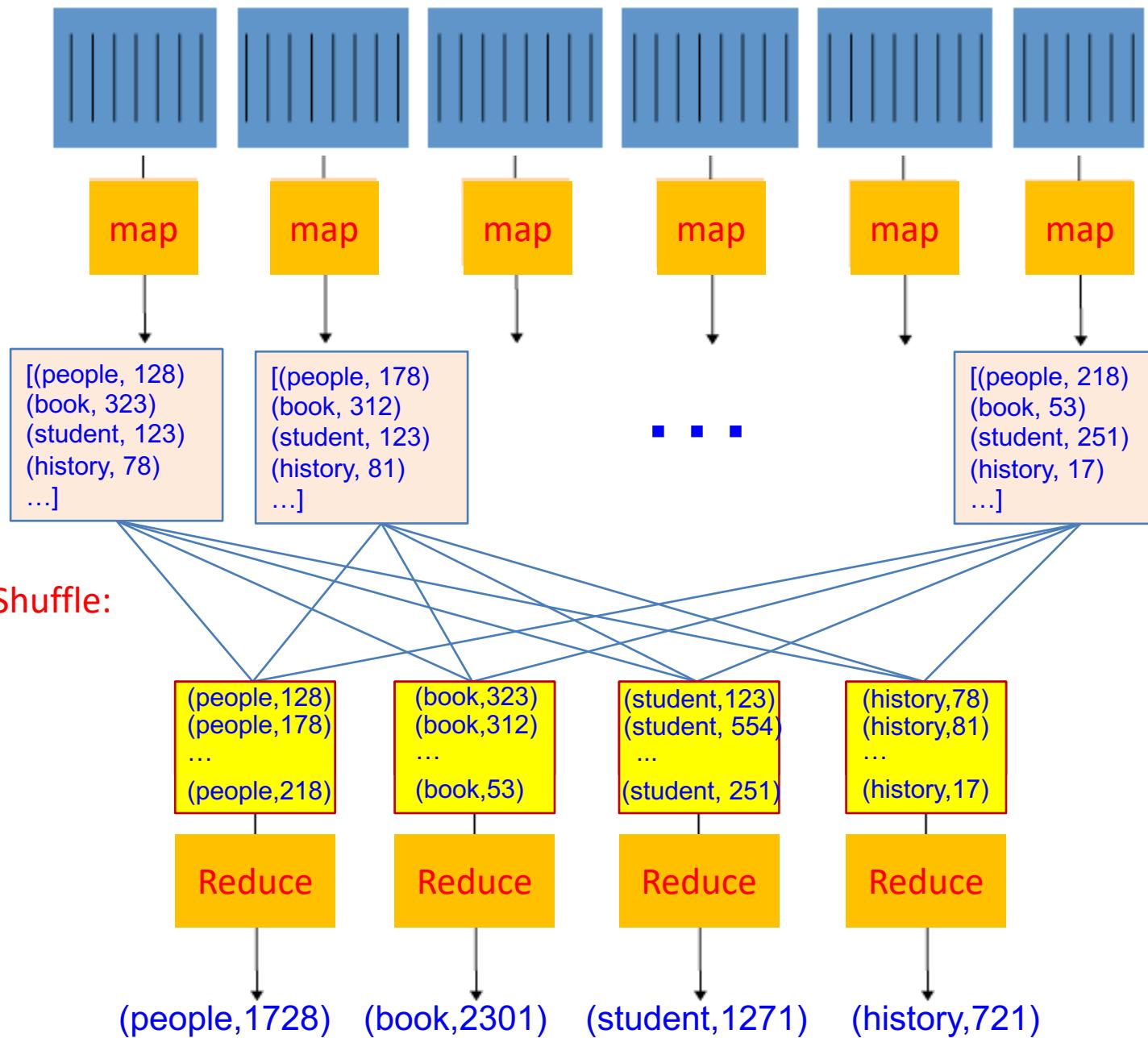








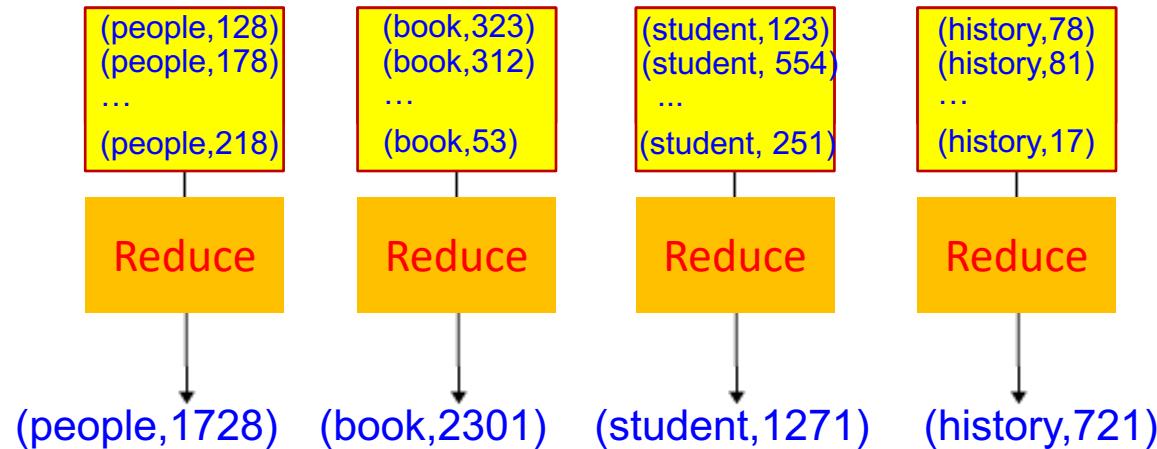




Map: Count All words in
Each chunk of data

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

REDUCE



- **reduce function** merges all intermediate values associated with the same intermediate key.
- 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)



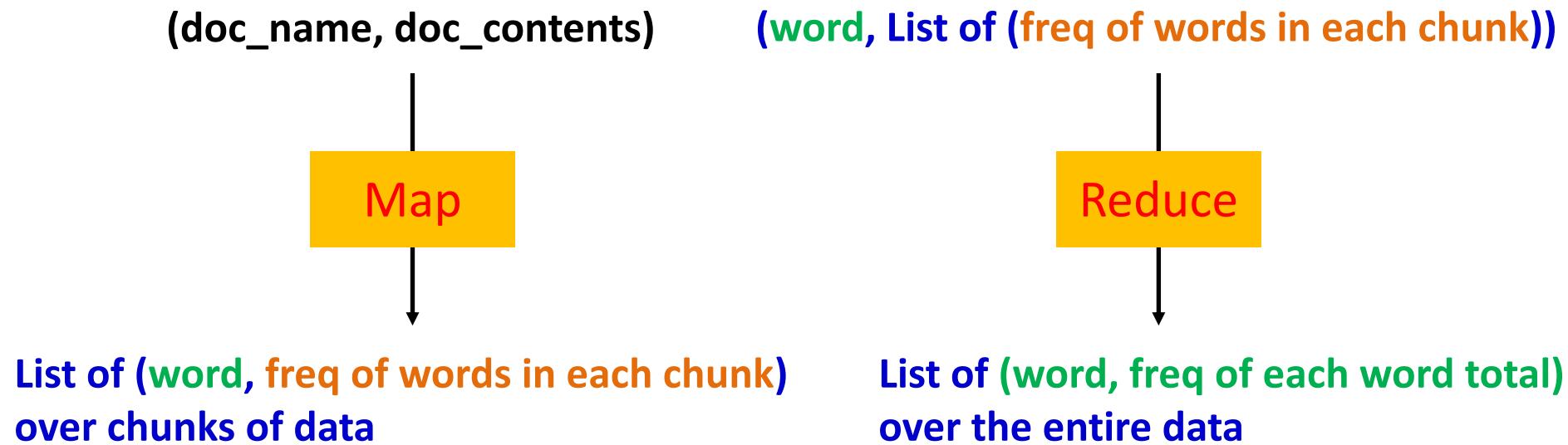
Inputs and Outputs

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



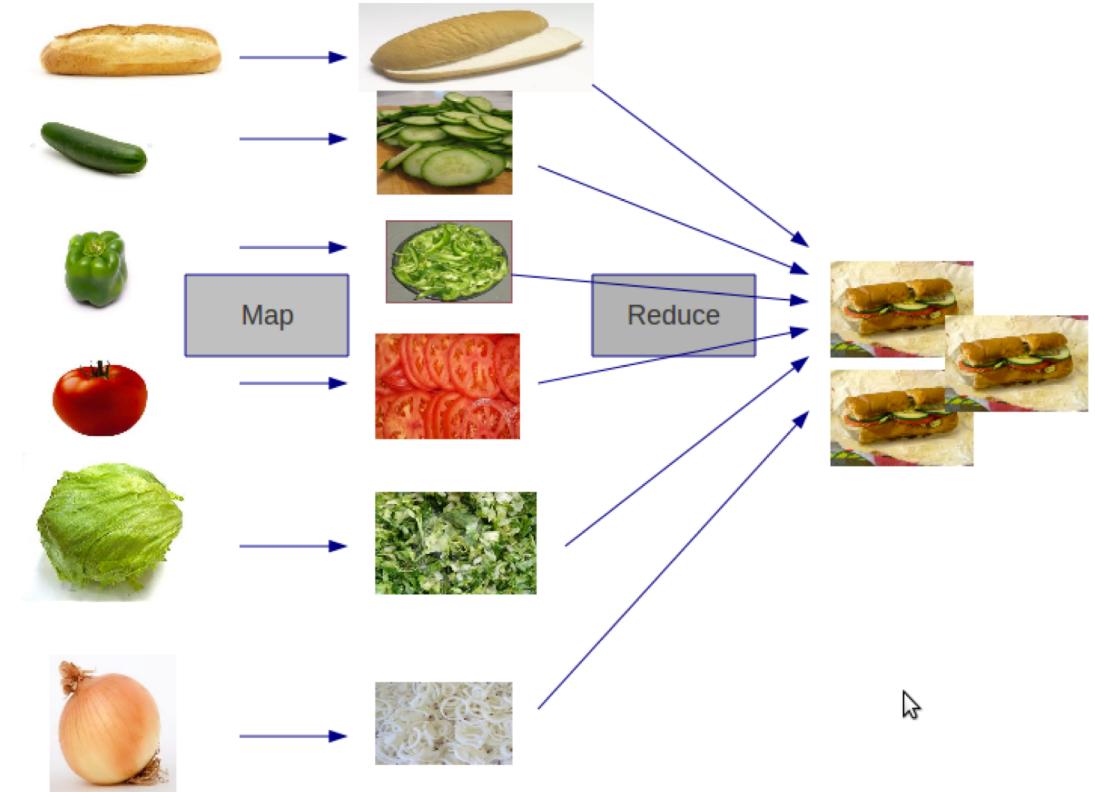
Inputs and Outputs (Example)

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



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?