Week 3-1

MapReduce

Big Data

Prof. Hwanjo Yu
POSTECH

Size of data

R, SAS, SPSS, Excel, …

SQLite, MySQL, …

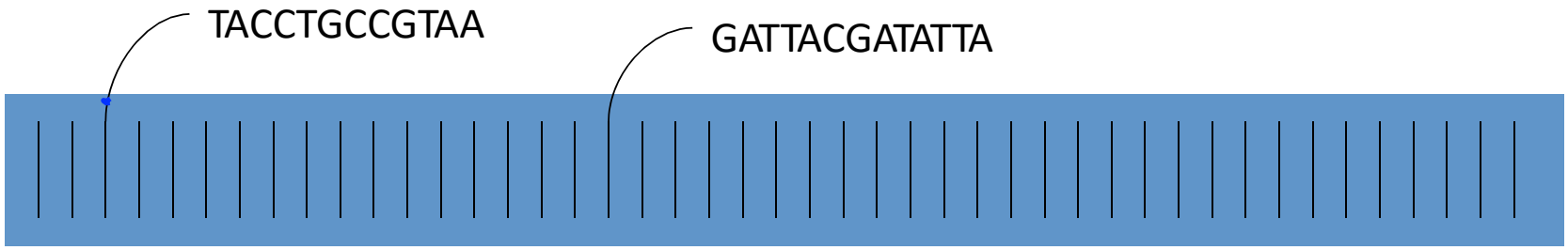
Hadoop, Pig, Hive, NoSQL, SPARK, …

What does scalable mean?

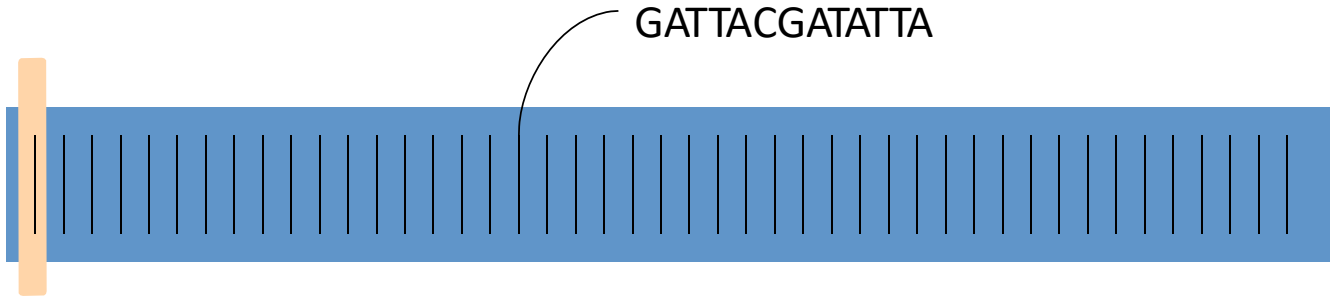
- Operationally:
  - In the past: **“Works even if data doesn’t fit in main memory”**
  - Now: **“Can make use of 1000s of cheap computers”**
- Algorithmically:
  - In the past: **If you have N data items, you must do no more than N^m operations -- “polynomial time algorithms”**
  - Now: **If you have N data items, you must do no more than N^m/k operations, for some large k**
    - Polynomial-time algorithms must be parallelized
  - Soon: **If you have N data items, you should do no more than $N * log(N)$ operations**
    - As data sizes go up, you may only get one pass at the data
    - The data is streaming -- you better make that one pass count
    - Ex: Large Synoptic Survey Telescope (30TB / night)

Example: Find matching DNA sequences

- Given a set of sequences
- Find all sequences equal to “GATTACGATATTA”



Example: Find matching DNA sequences

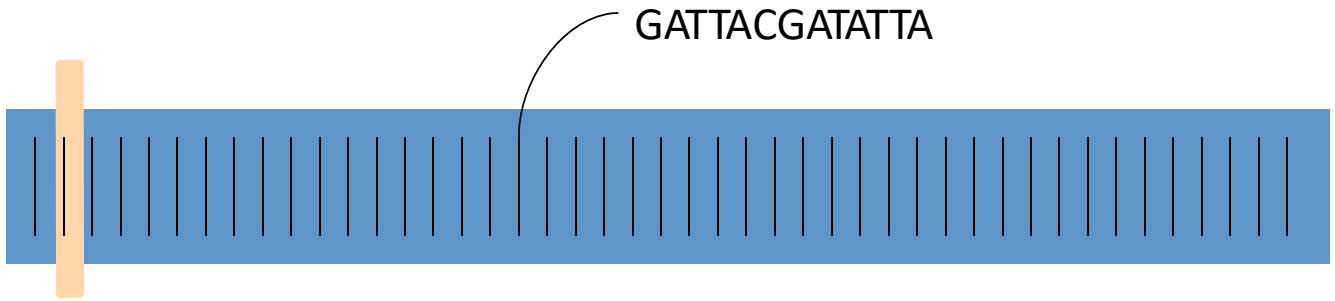


TACCTGCCGTAA = GATTACGATATTA?

No.

time = 0

Example: Find matching DNA sequences

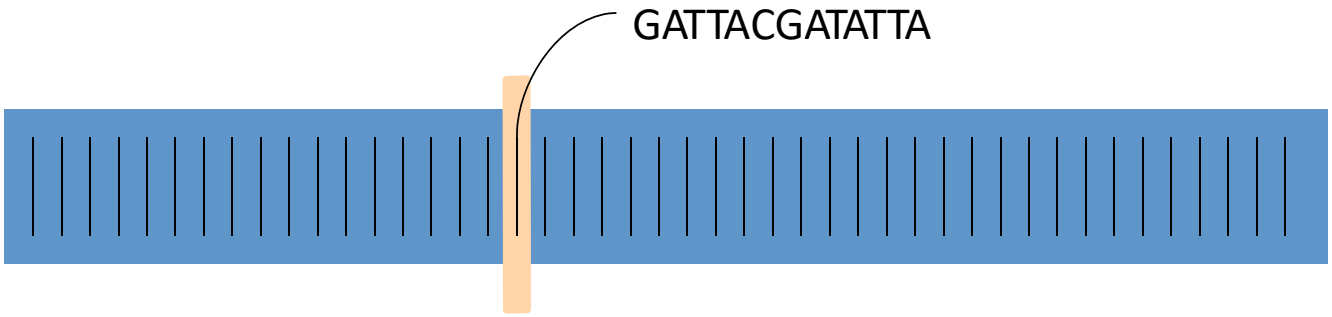


CCCCCAATGAC = GATTACGATATTA?

No.

time = 1

Example: Find matching DNA sequences



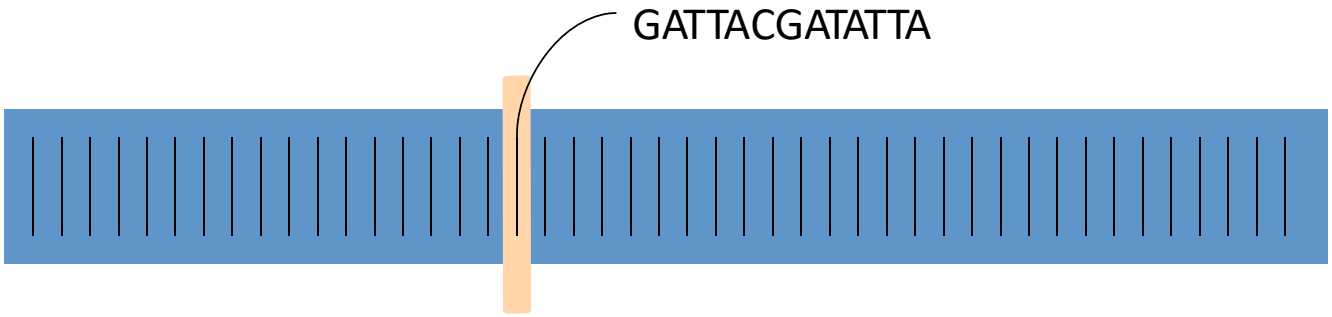
GATTACGATATTA contains GATTACGATATTA?

Yes!

Send it to the output

time = 17

Example: Find matching DNA sequences

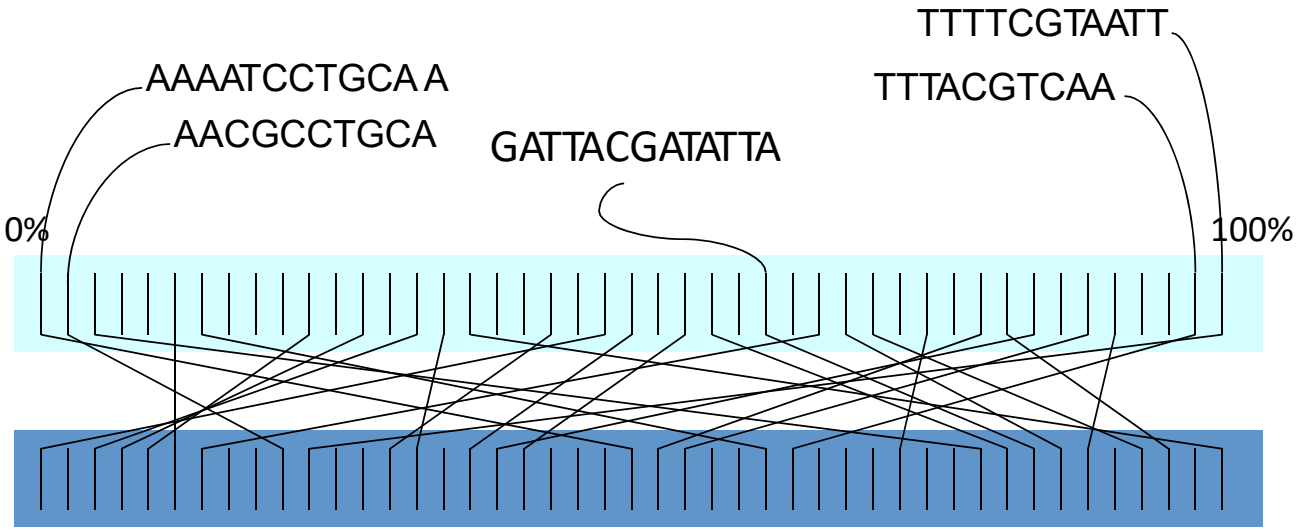


40 records, 40 comparisons

N records, N comparisons

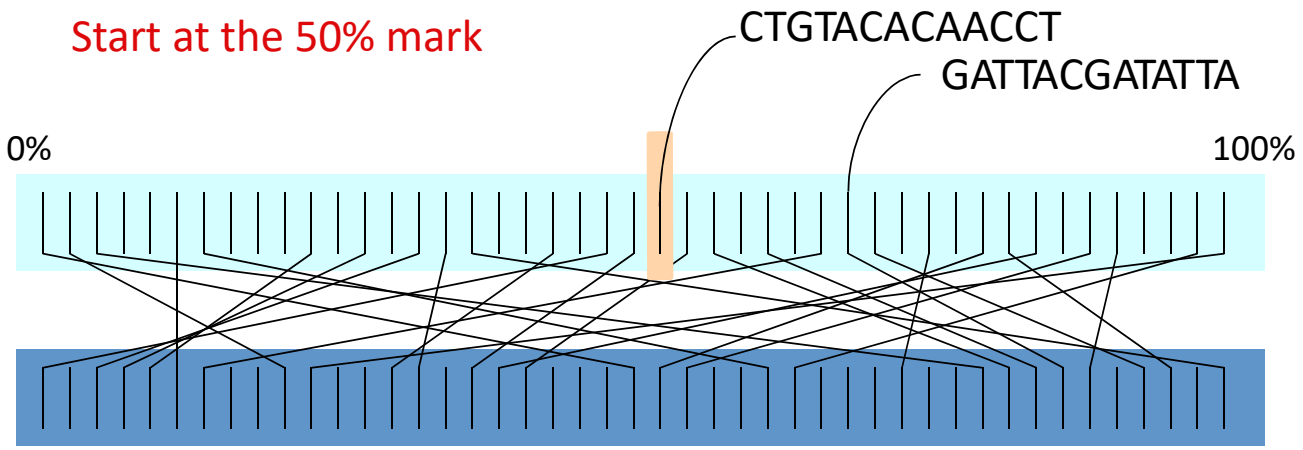
The algorithmic complexity is order N: O(N)

Example: Find matching DNA sequences



What if we sort the sequences?

Example: Find matching DNA sequences



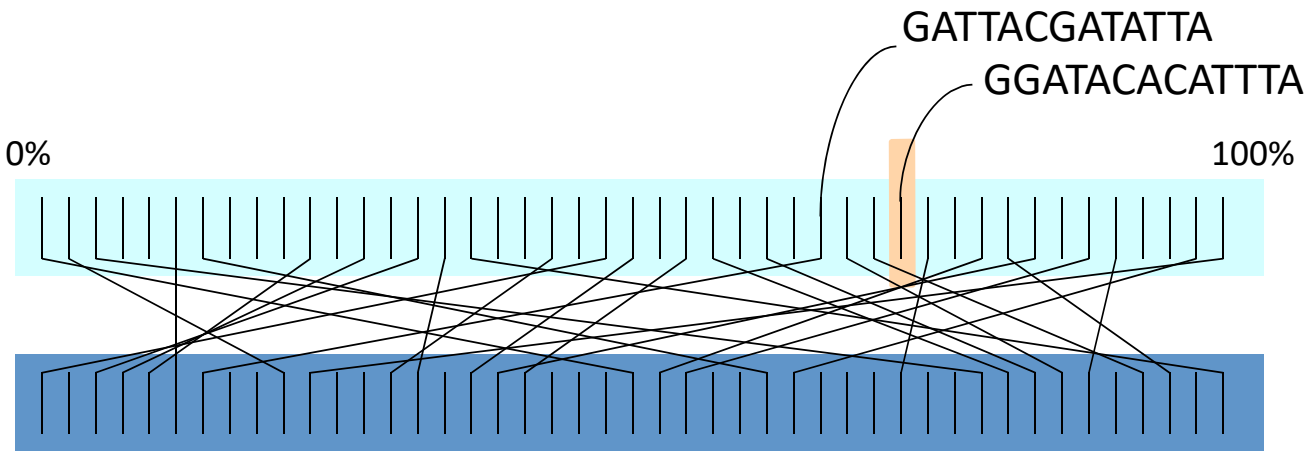
CTGTACACAACCT < GATTACGATATTA

time = 0

No match.

Skip to 75% mark

Example: Find matching DNA sequences



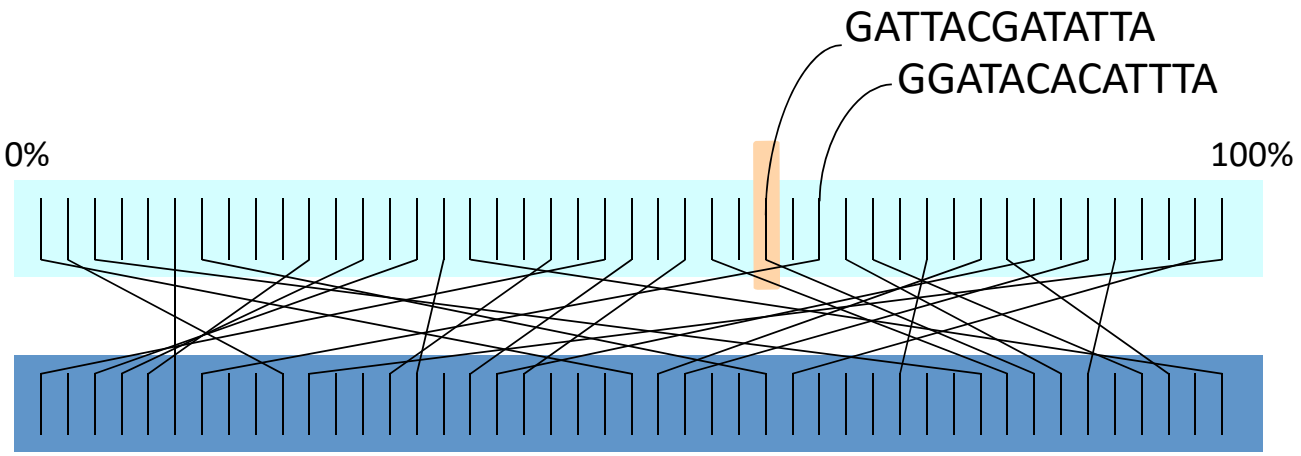
GGATACACATTTA > GATTACGATATTA

time = 1

No match.

Go back to 62.5% mark

Example: Find matching DNA sequences

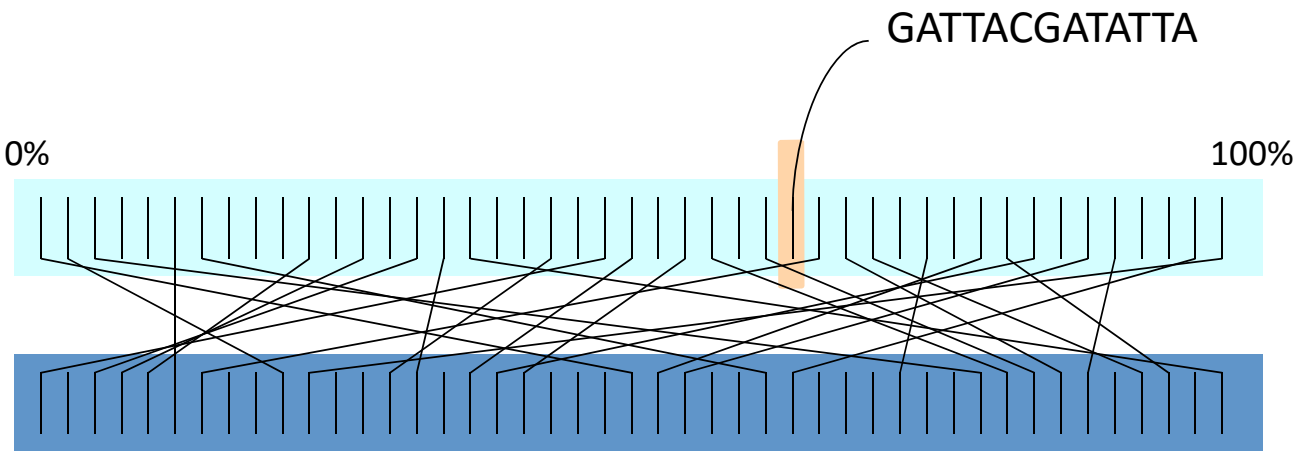


GATATTTTAAGC < GATTACGATATTA

No match.

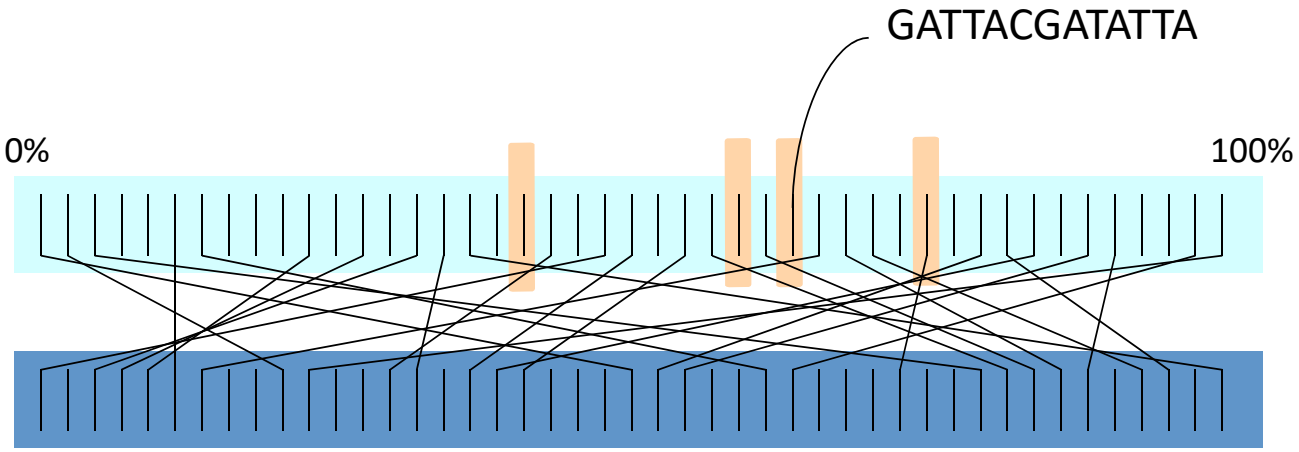
Skip back to 68.75% mark

Example: Find matching DNA sequences



GATTACGATATTA = GATTACGATATTA Match!

Example: Find matching DNA sequences



How many comparisons did we do?

40 records, only 4 comparisons
N records, $\log(N)$ comparisons
This algorithm is $O(\log(N))$

Far better scalability

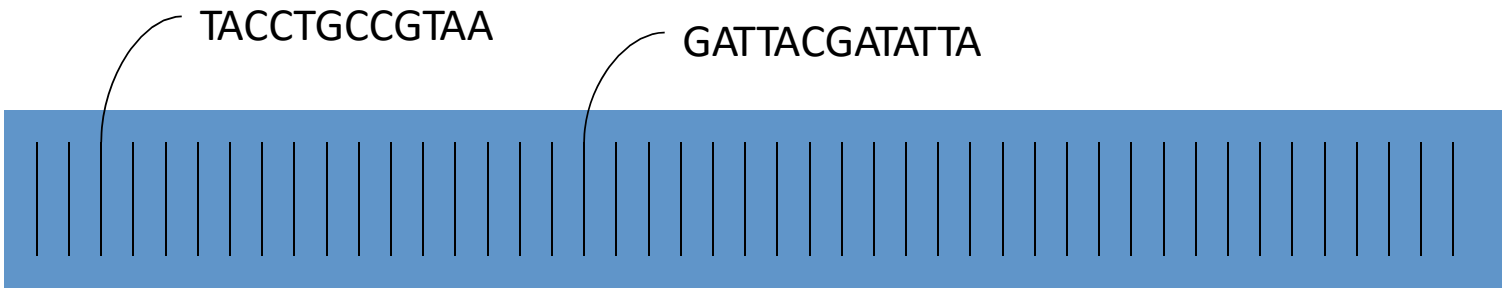
Relational database

- Databases are good at “Needle in Haystack” problems:
  - **Extracting small results from big datasets**
  - **Transparently provide “old style” scalability**
  - **Your query will always finish, regardless of dataset size.**
  - **Indexes are <u>easily built</u> and <u>automatically used</u> when appropriate**

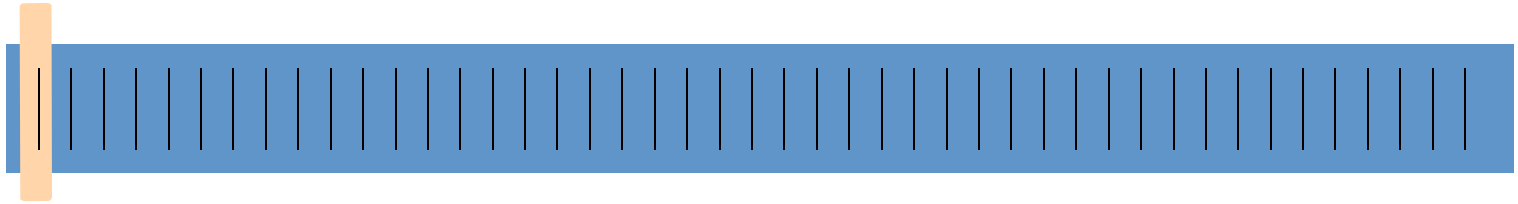
```
CREATE INDEX seq_idx ON sequence(seq);

SELECT seq
FROM sequence
WHERE seq = ‘GATTACGATATTA’;
```

New task: Read trimming

- Given a set of DNA sequences
- Trim the final *n* bps of each sequence
- Generate a new dataset



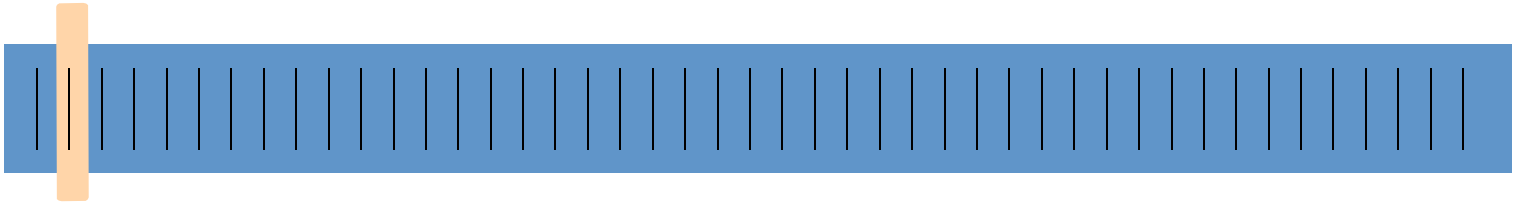
New task: Read trimming



TACCTGCCGTAA becomes TACCT

time = 0

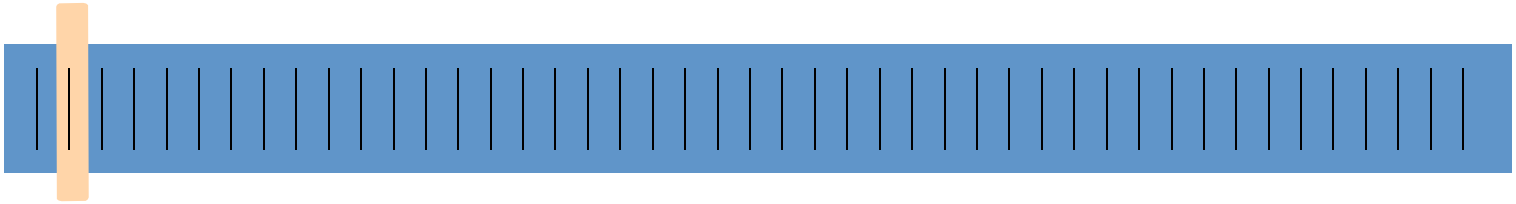
New task: Read trimming



CCCCCAATGAC becomes CCCCC

time = 1

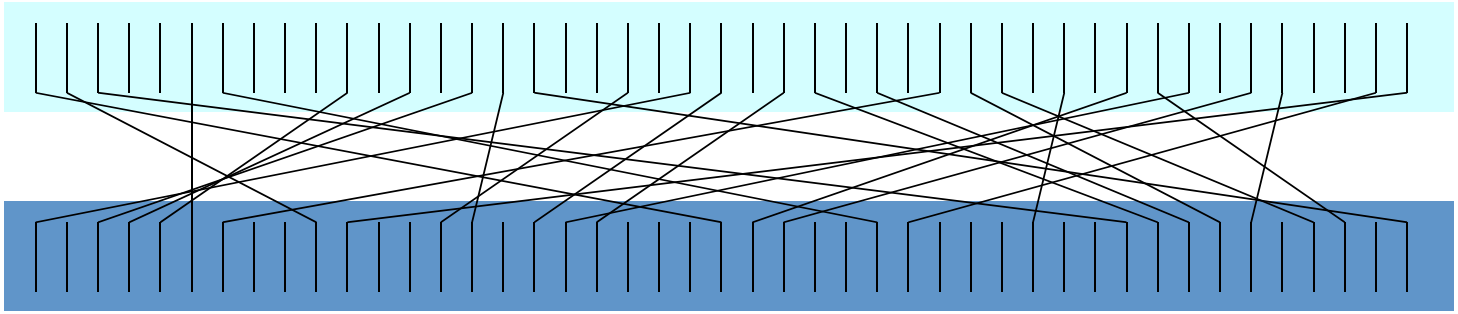
New task: Read trimming



GATTACGATATTA becomes GATTA

time = 17

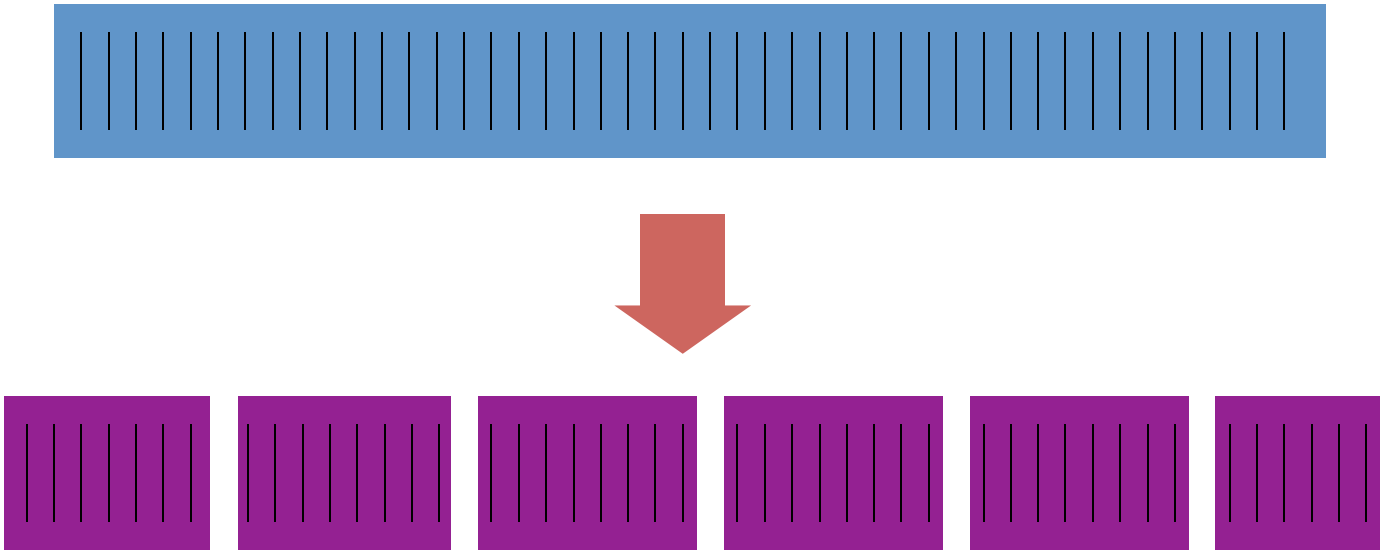
New task: Read trimming

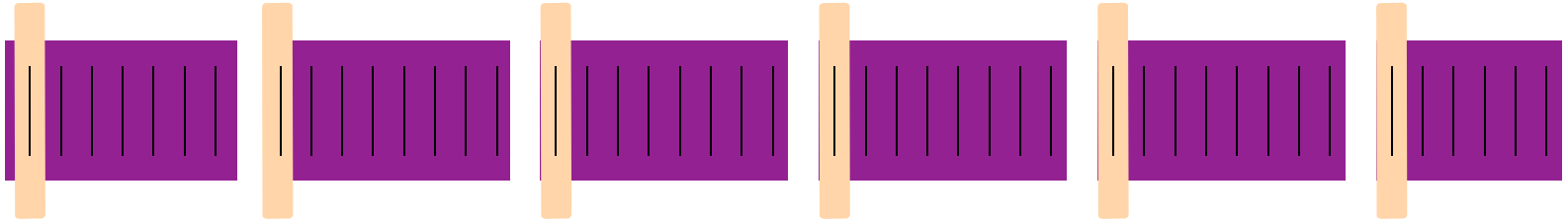


Can we use an index?

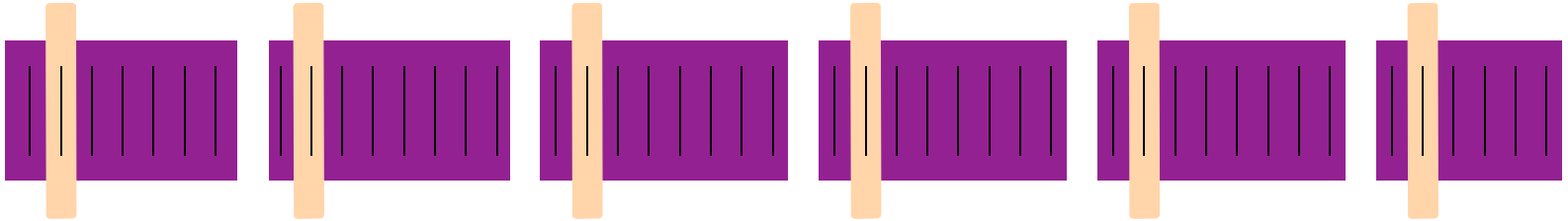
No. We have to touch every record no matter what. The task is fundamentally $O(N)$

Can we do any better?

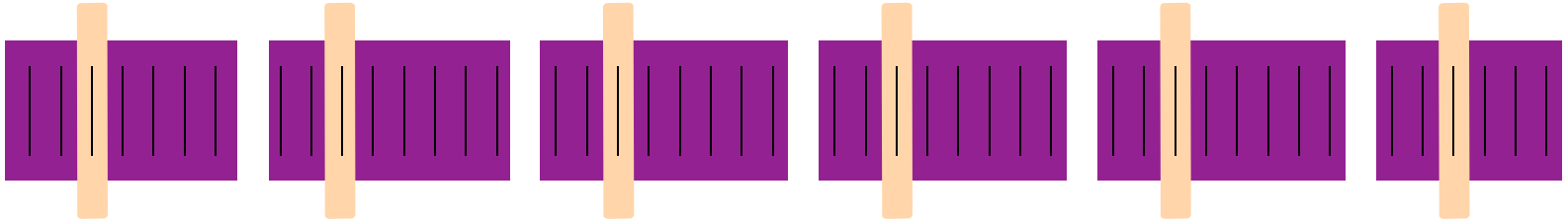




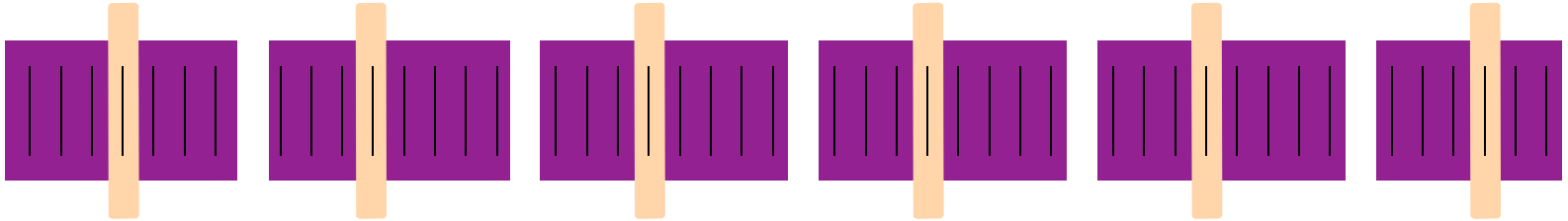
time = 0



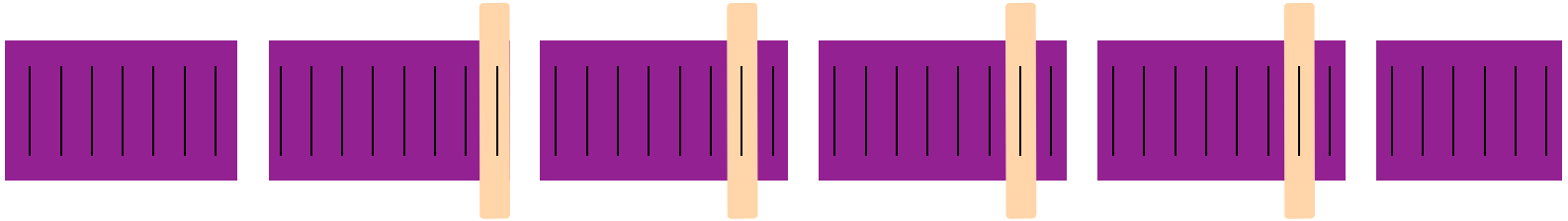
time = 1



time = 2



time = 3



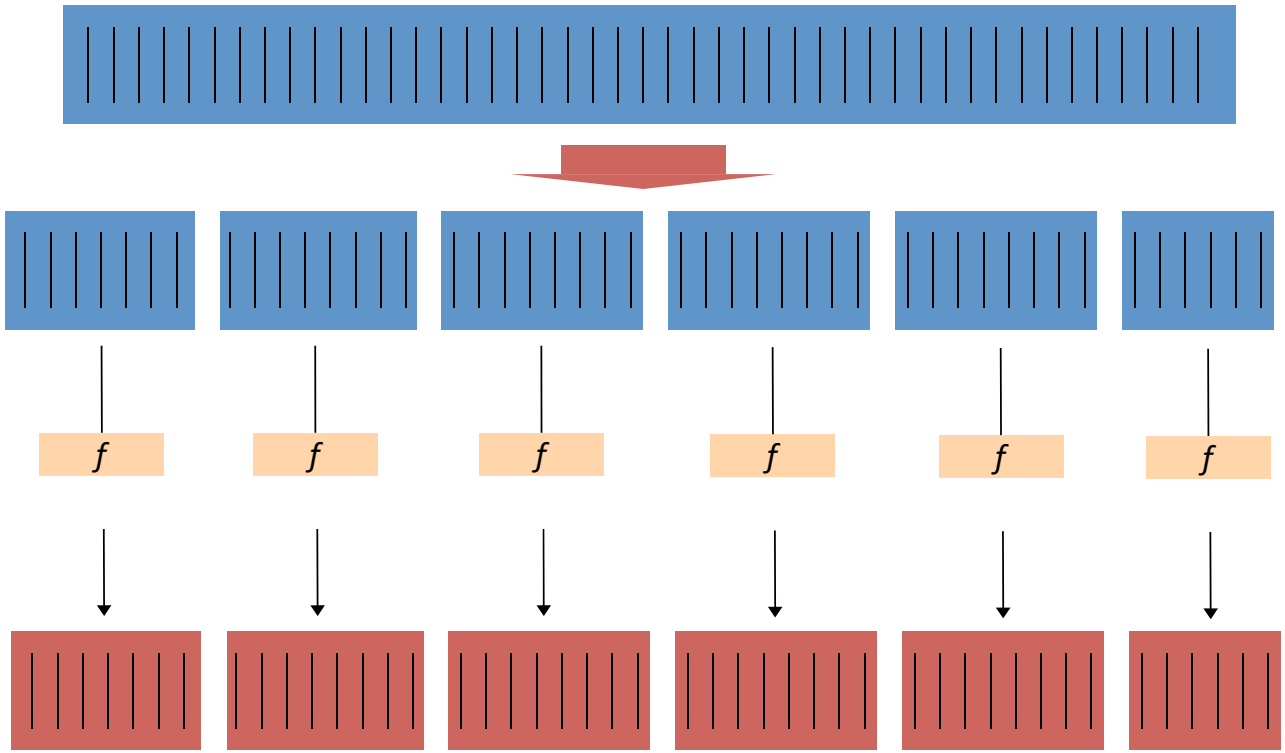
How much time did this take?
7 cycles

time = 7

40 records, 6 workers

$O(N/k)$

Schematic of a parallel “Read Trimming” task



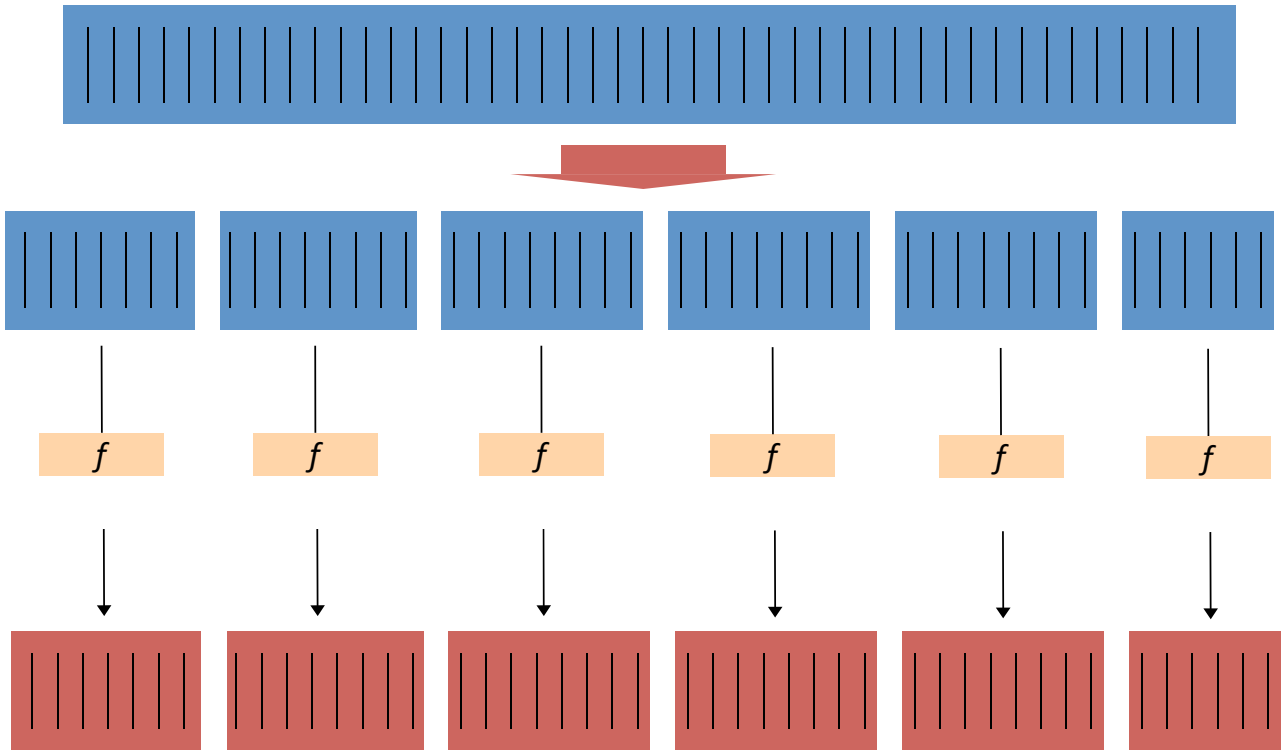
You are given short “reads”: genomic sequences about 35-75 characters each

Distribute the reads among k computers

f is a function to trim a read; apply it to every item

Now we have a big distributed set of trimmed reads

New task: Convert 405k TIFF images to PNG



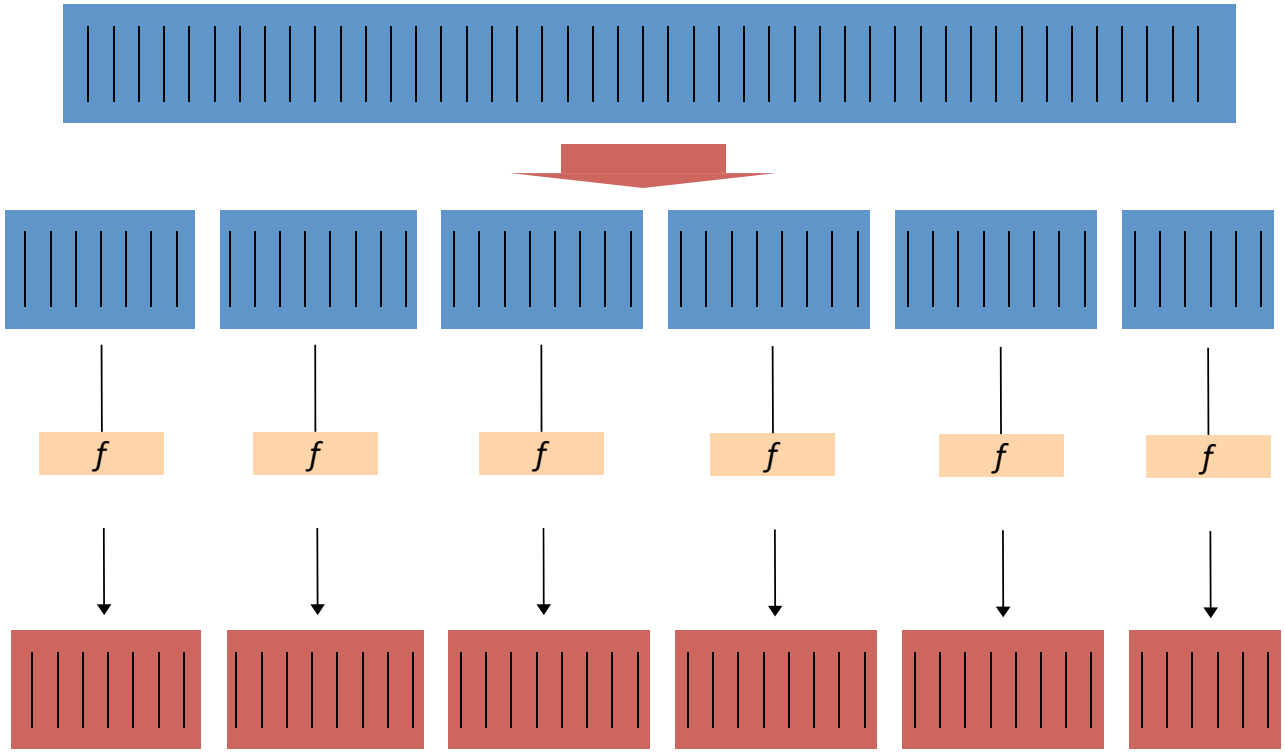
You are given TIFF images

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

New task: Run thousands of simulations



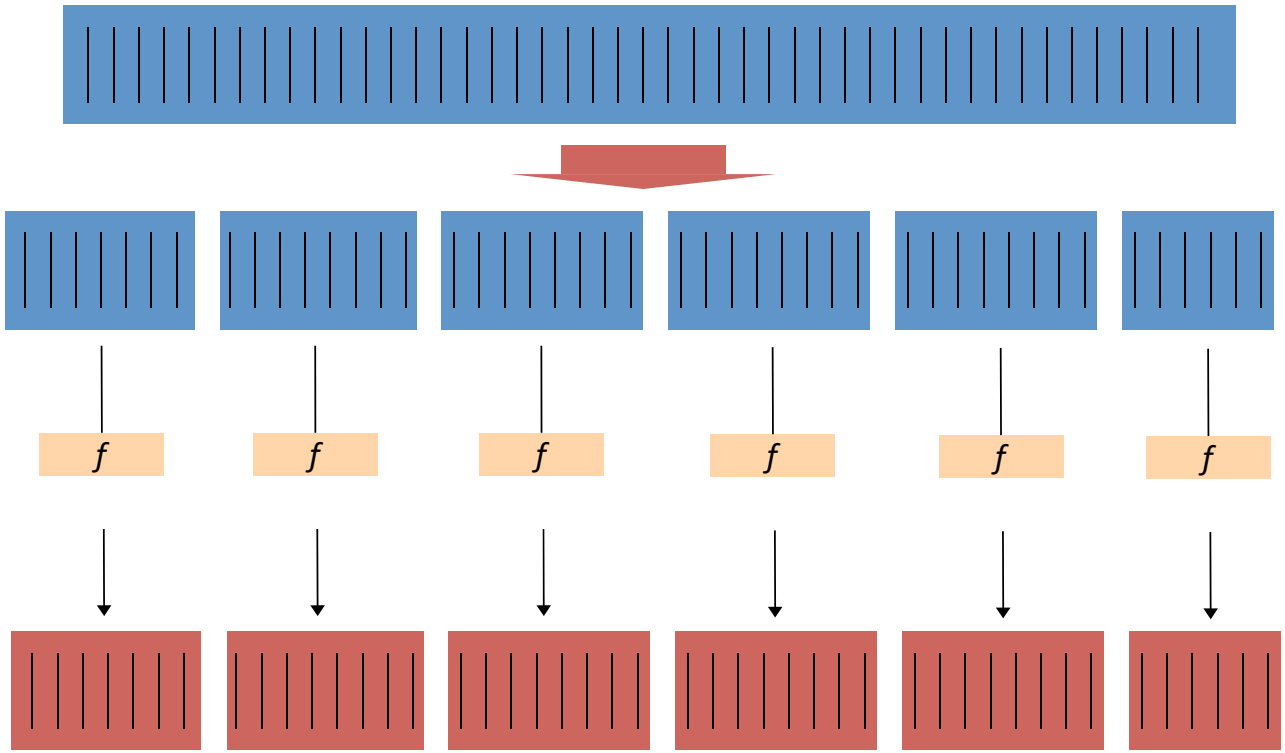
You have sets of parameters for thousands of small simulations

Divide the parameter sets among k computers

f is runs the simulation and produces some output; apply it to every item

Now we have a big distributed set of simulation results

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 a single document

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

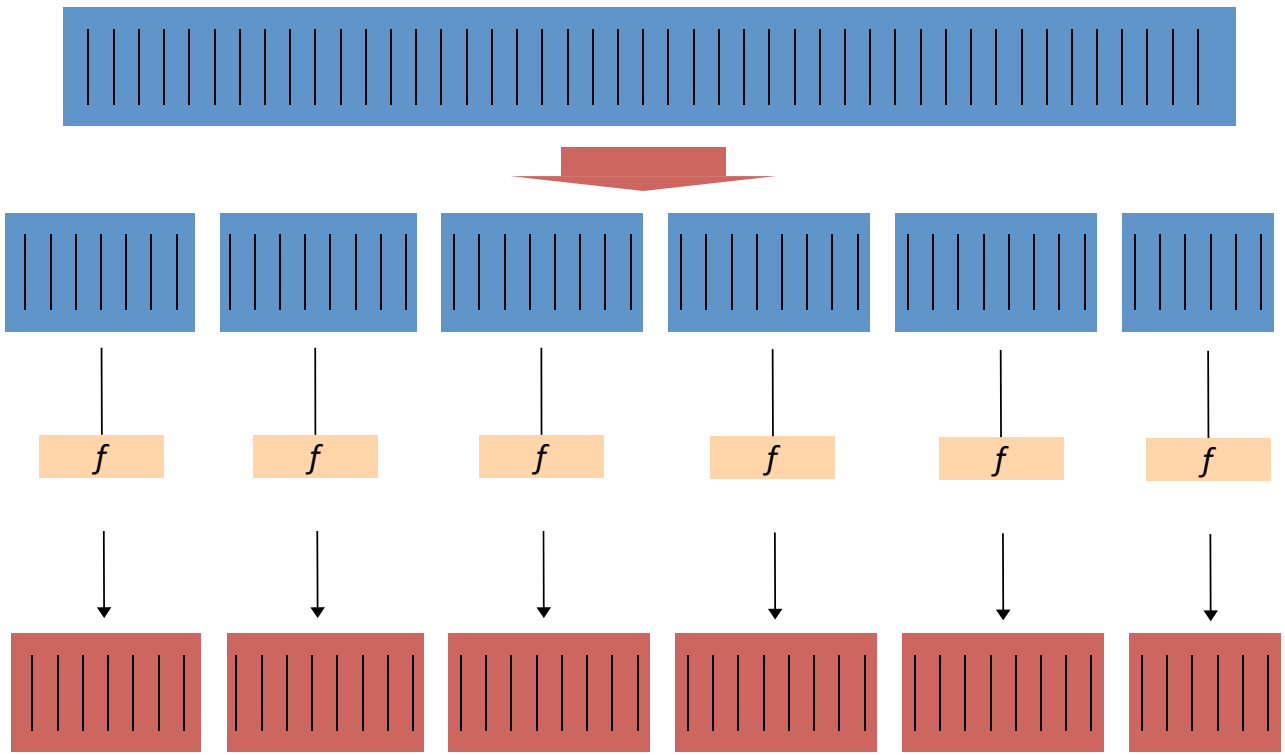
- **Consider a slightly more general program to compute the word frequency of every word in a single document**

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 th e laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they shoul d 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 creatio n 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 en ds, 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 happine ss. Prudence indeed will dictate that governments long established should not be changed for light and transient caus es: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, th an to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and us urpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them t o 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 an d usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of whi ch have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitte d to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.

(people, 2)
(government, 6)
(assume, 1)
(history, 2)
…

Compute the word frequency of 5M documents



You have millions of documents

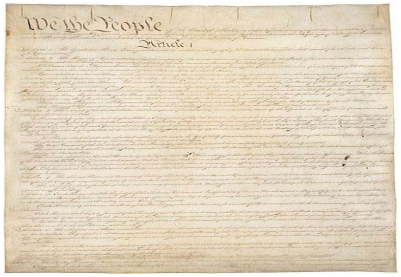
Distribute the documents among k computers

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

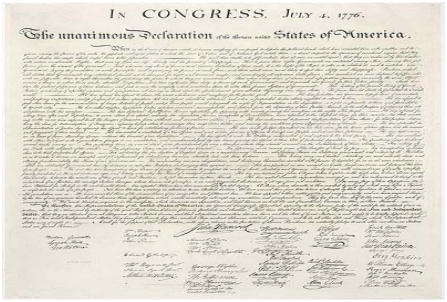
Now we have a big distributed list of sets of word freqs

- There’s a pattern here….
  - A function that *maps* a read to a trimmed read
  - A function that *maps* a TIFF image to a PNG image
  - *A* function that *maps* a set of parameters to a simulation result
  - A function that *maps* a document to its most common word
  - A function that *maps* a document to a histogram of word frequencies

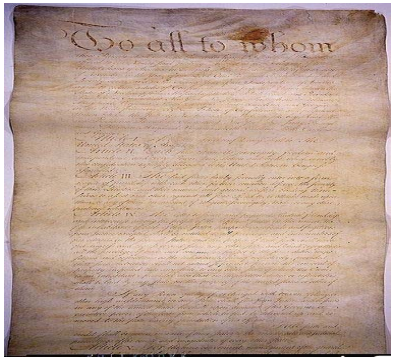
- What if we want to compute the word frequency across *all* documents?



US Constitution



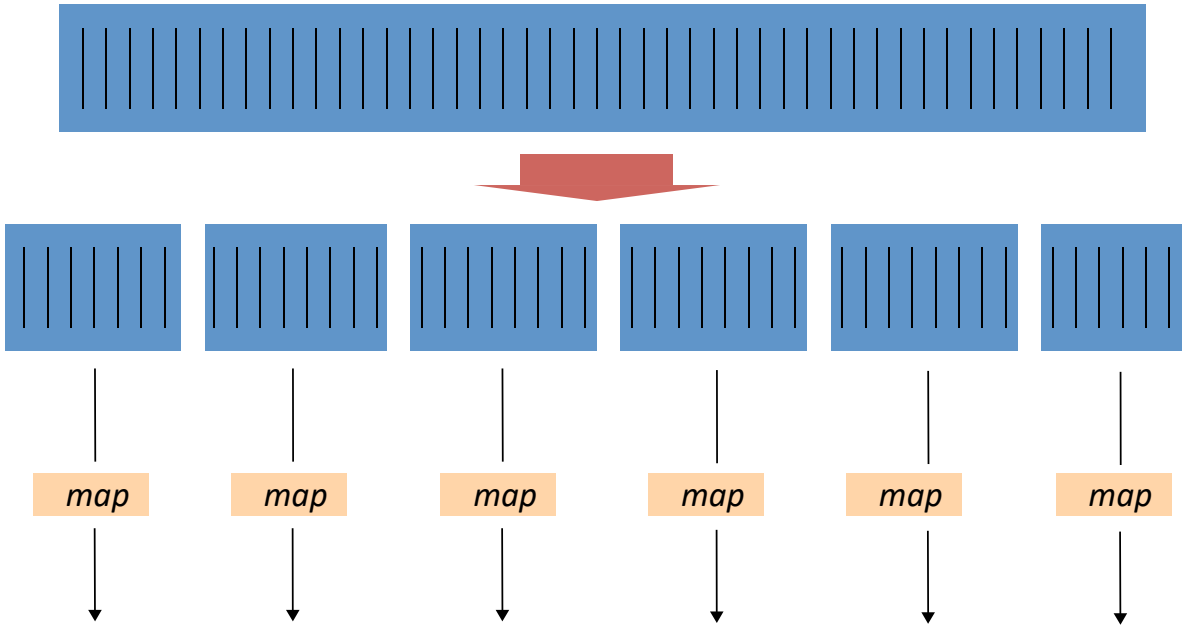
Declaration of Independence



Articles of Confederation

(people, 78)
(government, 123)
(assume, 23)
(history, 38)
…

Compute the word frequency across 5M documents



You have millions of documents

Distribute the documents among k computers

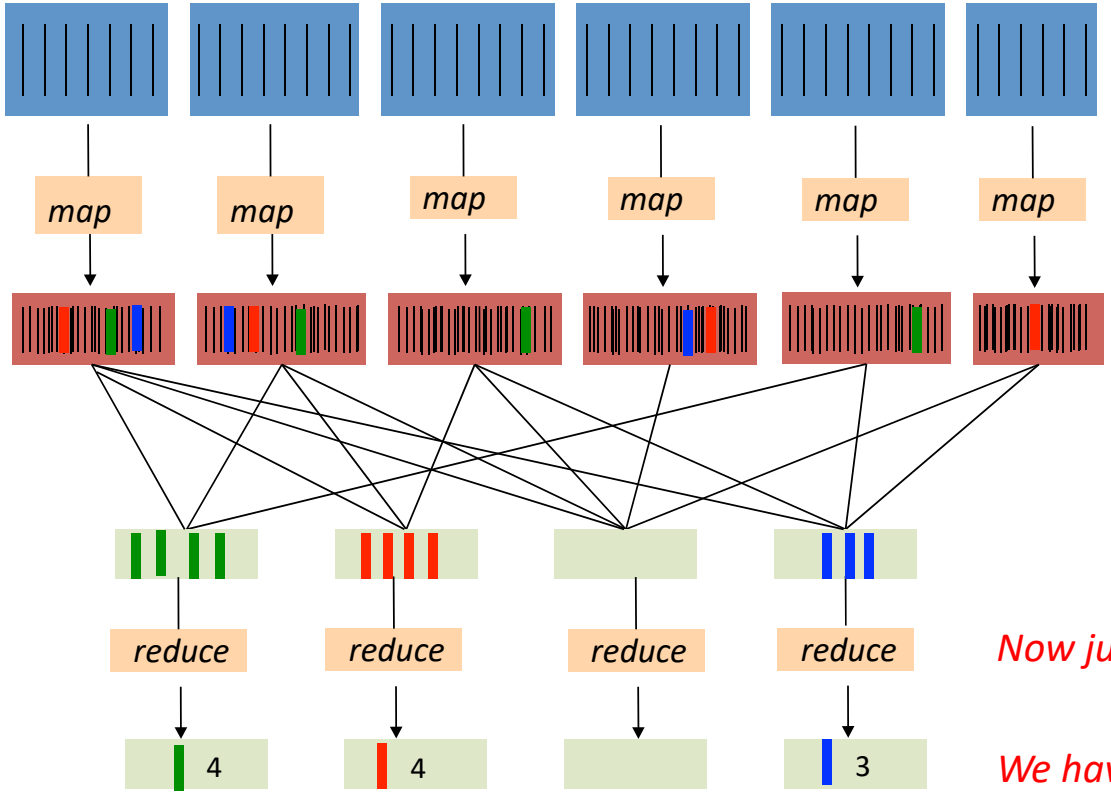
For each document, return a set of (word, freq) pairs

Now what?

But we don't want a bunch of little histograms – we want one big histogram.

How can we make sure that a single computer has access to every occurrence of a given word regardless of which document it appeared in?

Compute the word frequency across 5M documents



Distribute the documents among k computers

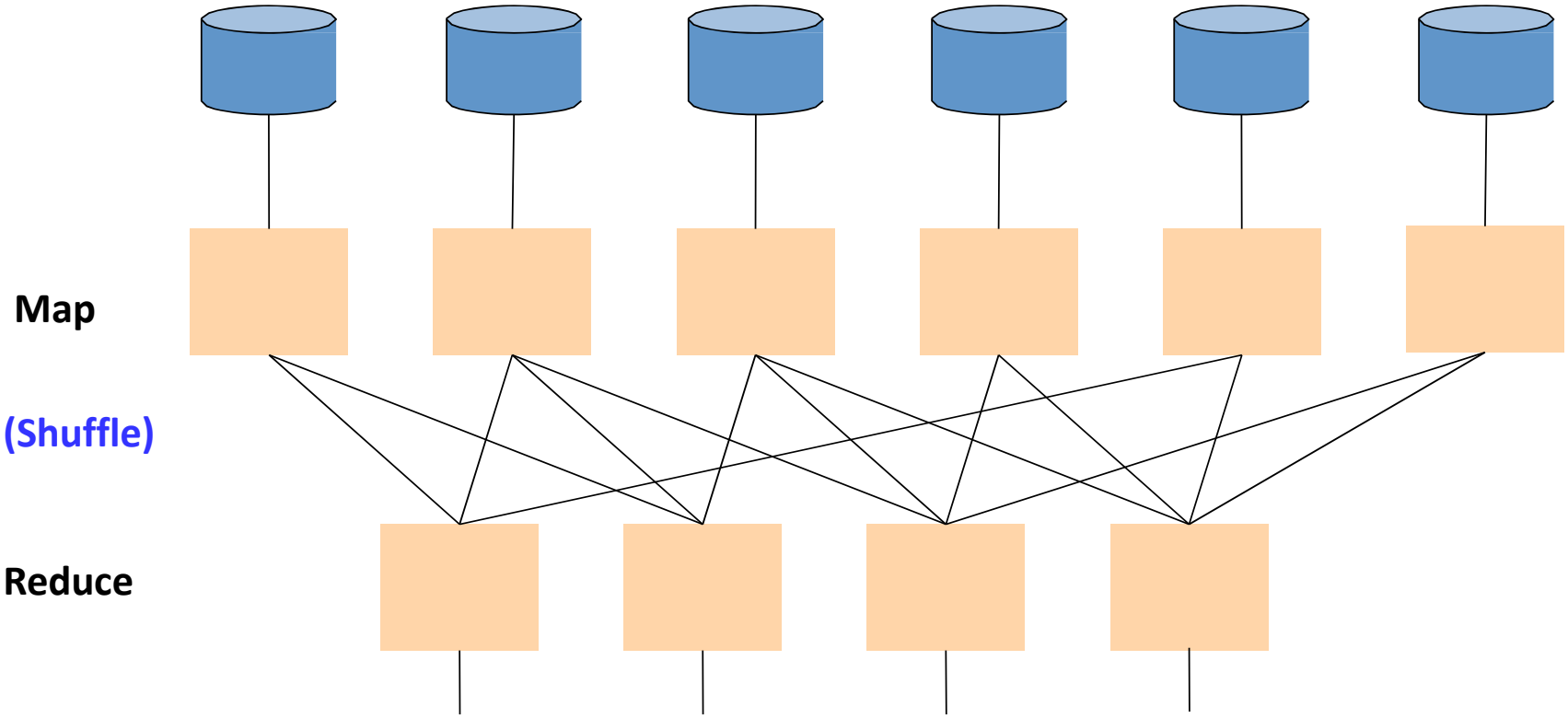
For each document, return a set of (word, freq) pairs

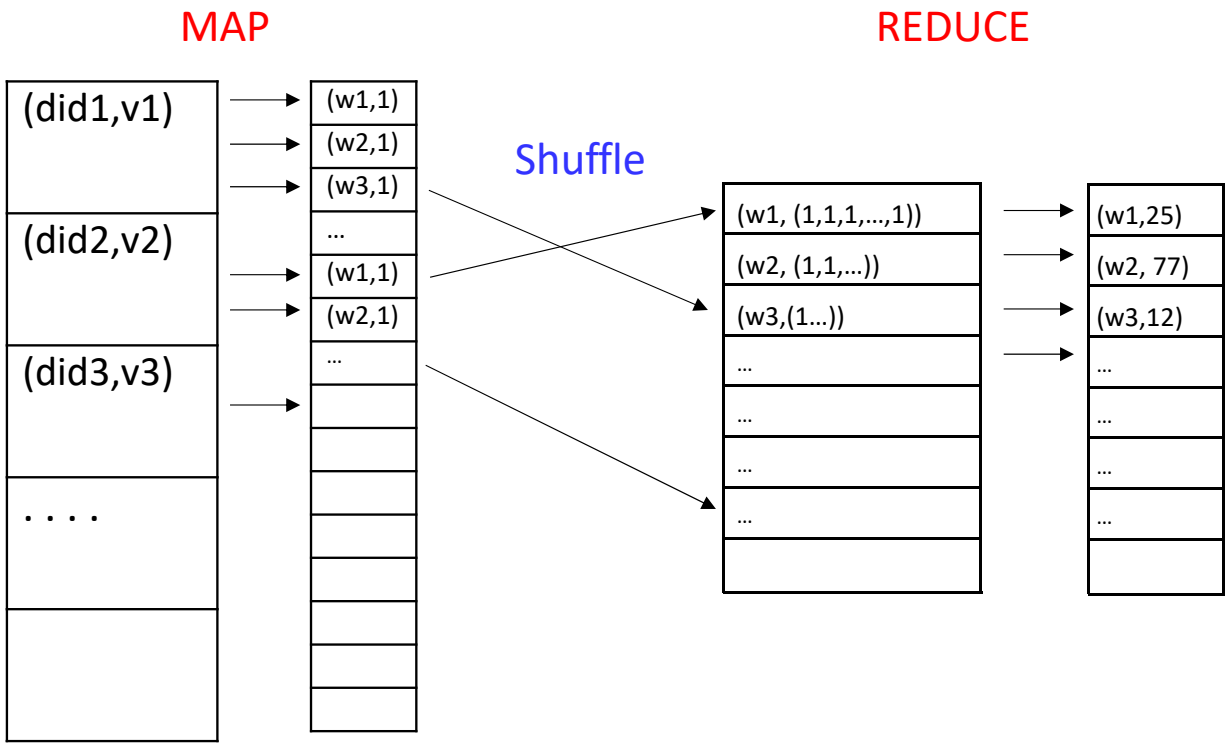
Now we have a big distributed list of sets of word freqs.

Now just count the occurrences of each word

We have our distributed histogram

Distributed algorithm




MAP
REDUCE
(did1,v1)
(did2,v2)
(did3,v3)
. . . .
(w1,1)
(w2,1)
(w3,1)
…
(w1,1)
(w2,1)
…
Shuffle
(w1, (1,1,1,…,1))
(w2, (1,1,…))
(w3,(1…))
…
…
…
…
(w1,25)
(w2, 77)
(w3,12)
…
…
…
…

Map Reduce

- Google: paper published 2004
- Free variant: Hadoop
- Map-reduce = high-level programming model for large-scale distributed data processing

Map Reduce

- Data Model
  - Files!
  - A file = a bag of (key, value) pairs
- A Map-Reduce program
  - Input: a bag of (input_key, value) pairs
  - Output: a bag of (output_key, value) pairs

Step 1 : the Map phase

- User provides the Map function
  - Input: (input_key, value)
  - Output: a bag of (intermediate_key, value)
- System applies the Map function in parallel to all (input_key, value) pairs in the input file.

Step 2: the Reduce phase

- User provides the Reduce function
  - Input: (intermediate_key, bag of values)
  - Output: a bag of output (value)
- System will group all pairs with the same intermediate_key, and passes the bag of values to the Reduce function.

Map Reduce programming model

- Input & Output: each a set of (key, value) pairs
- Programmer defines two functions

- map(in_key, in_value) => list(intermediate_key, intermediate_value)
  - Processes input (key, value) pair
  - Produces a set of intermediate (key, value) pairs

- reduce(intermediate_key, list(intermediate_value)) => list(out_value)
  - Combines all intermediate values for a particular key
  - Produces a set of merged output values (usually just one)

Example: What does this do ?

```
map(String in_key, String in_value):
    // in_key: document name
    // in_value: document contents
    For each word w in in_value:
        Emit(w,1);

reduce(String intermediate_key, Iterator intermediate_values):
    // intermediate_key: word
    // intermediate_values: ???
    Int result = 0;
    For each v in intermediate_values:
        result += v;
    Emit(intermediate_key, result);
```