

# The problem:

- Big data means ...

# The problem:

- Big data means ...  
lots of hard drives



# The solution:

- Lots of data means we should...

# The solution:

- Lots of data means we should...

bring computation to data!

# Lots of disks:



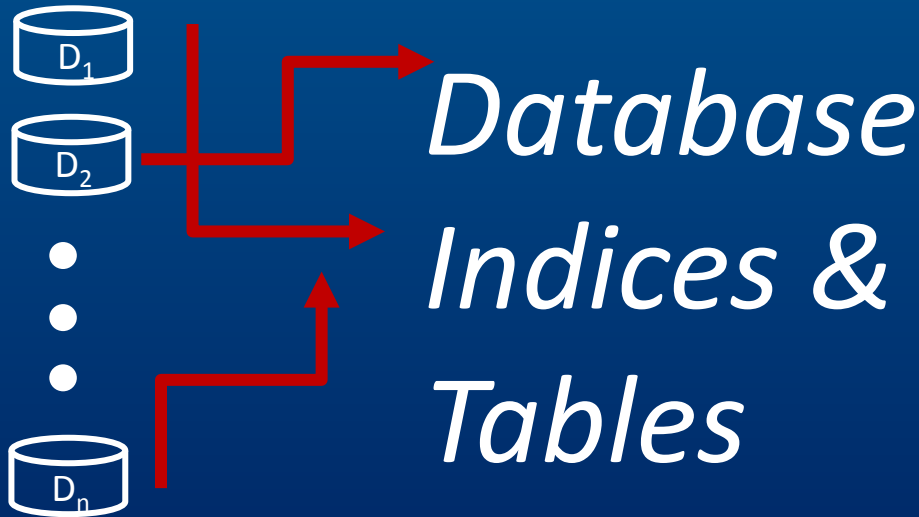
# Possibilities:

- Case 1: data needs updating



# Possibilities:

- Case 1: data needs updating so ...



# Possibilities:

- Case 2: need to sweep through data





# Possibilities:

- Case 2: need to sweep through data so...



# The framework:

- User defines:
  - a. <key, value>

# The framework:

- User defines:
  - a. `<key, value>`
  - b. mapper & reducer functions

# The framework:

- User defines:
  - a. `<key, value>`
  - b. mapper & reducer functions
- Hadoop handles the logistics

# The logistics:

- Hadoop handles the distribution and execution



# Map/Reduce flow

- User defines a map function

```
map()
```

# Map/Reduce flow

- `map()` reads data and outputs `<key,value>`



# Map/Reduce flow

- User defines a reduce function

```
reduce()
```



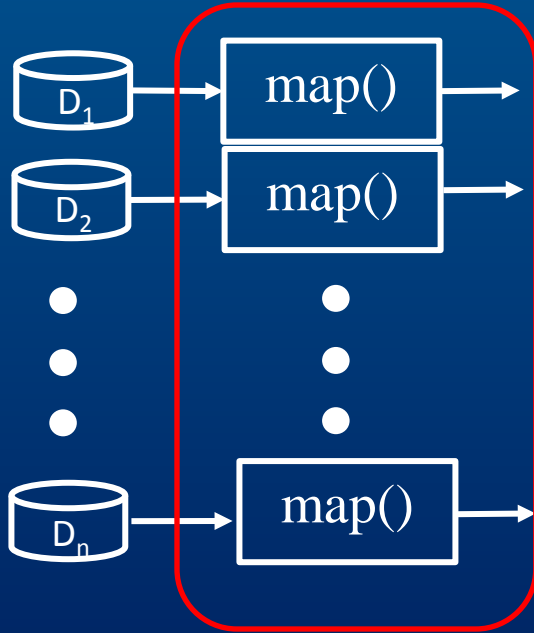
# Map/Reduce flow

- `reduce()` reads `<key,value>` and outputs your result



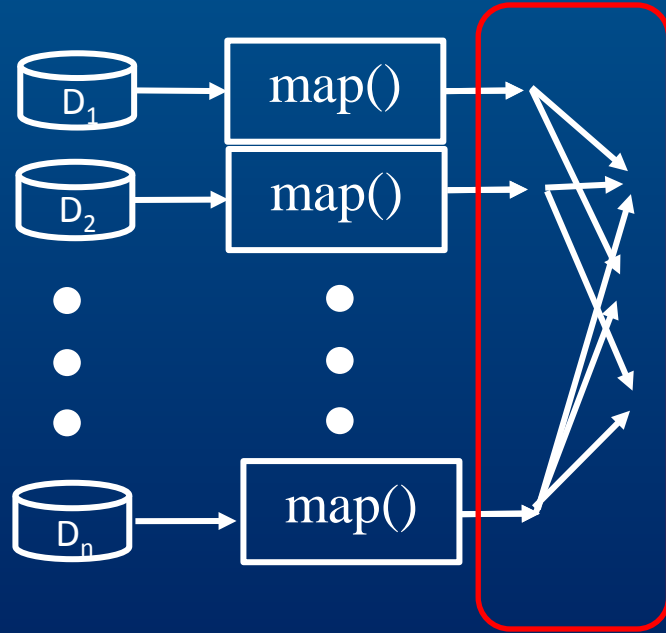
# Map/Reduce flow

- Hadoop distributes `map()` to data



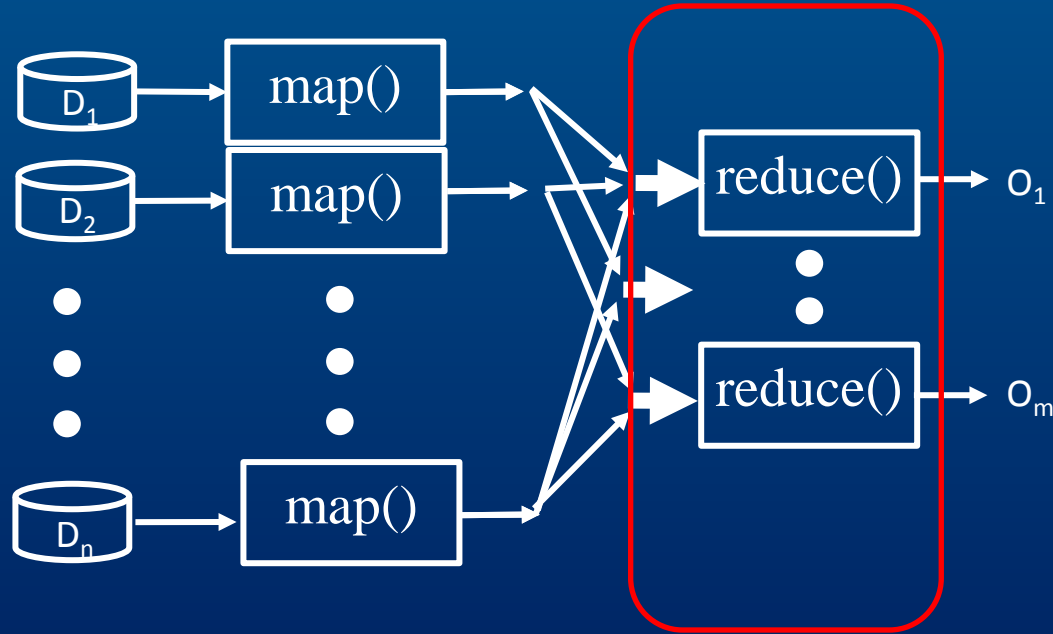
# Map/Reduce flow

- Hadoop groups  $\langle \text{key}, \text{value} \rangle$  data



# Map/Reduce flow

- Hadoop distributes groups to reducers()



# The paradigmatic example:

- Count word frequencies

# Wordcount task:

- How would you count all the words in Star Wars?

*Episode IV*

*A long  
time ago,  
in a galaxy  
far, far,  
away ...*

# Wordcount serial code:

- In a nutshell:

1. *Get word*

# Wordcount serial code:

- In a nutshell:

1. *Get word*
2. *Look up word in table*



# Wordcount serial code:

- In a nutshell:

1. *Get word*
2. *Look up word in table*
3. *Add 1 to count*

# Wordcount serial code:

- **Result Table:**

Word	Count
a	1000
far	2000
Jedi	5000
Luke	9000
...	

# Wordcount task:

- How would you count all the words in all the Star Wars scripts and ...

Episode  
MMMDXXLIV

Yet another  
long saga  
of cute versus  
not so cute...

# Wordcount task:

- ... books, blogs, and fan-fiction?



# Wordcount task:

- ... books, blogs, and fan-fiction?

A photograph of a row of comic books standing upright on a dark surface. The books are of various colors and designs. A prominent green rectangular box is overlaid on the middle of the row, containing white text. The text inside the box reads "use map/reduce of course".

*use map/reduce of course*

# Map/Reduce Strategy

- Keep it simple!

# Wordcount Strategy

- Let  $\langle \text{word}, 1 \rangle$  be the  $\langle \text{key}, \text{value} \rangle$

# Wordcount Strategy

- Let Hadoop do the hard work



# Wordcount Map/Reduce:

## *The Mapper:*

Loop  
Until  
Done



***Get word***  
***Emit <word> < 1>***

# What One Mapper Does

line =

A long time ago in a galaxy far far ...

keys =

A

long

time

ago

in

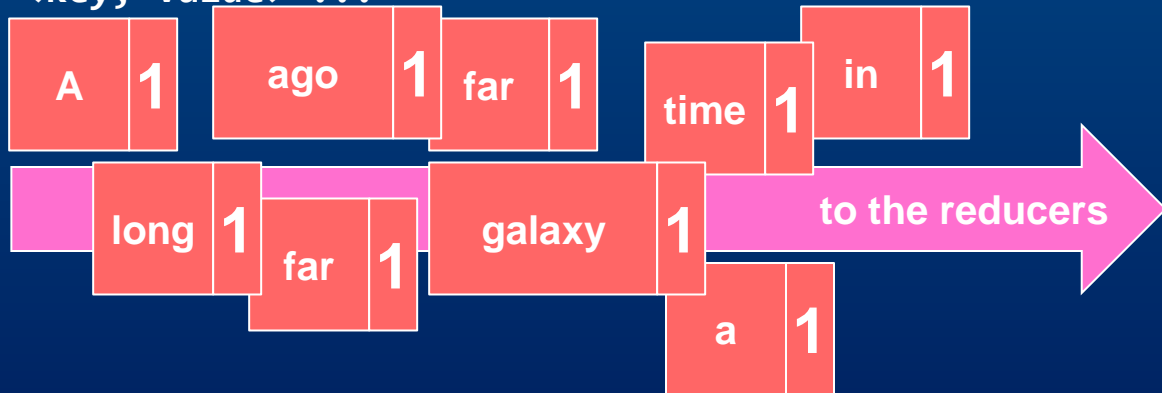
a

galaxy

far

far


Emit <key, value> ...



# Wordcount Map/Reduce:

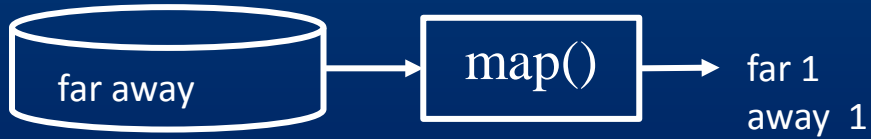
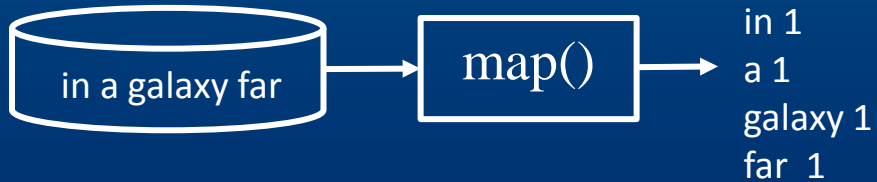
## *The Reducer:*

Loop  
Over  
key-  
values

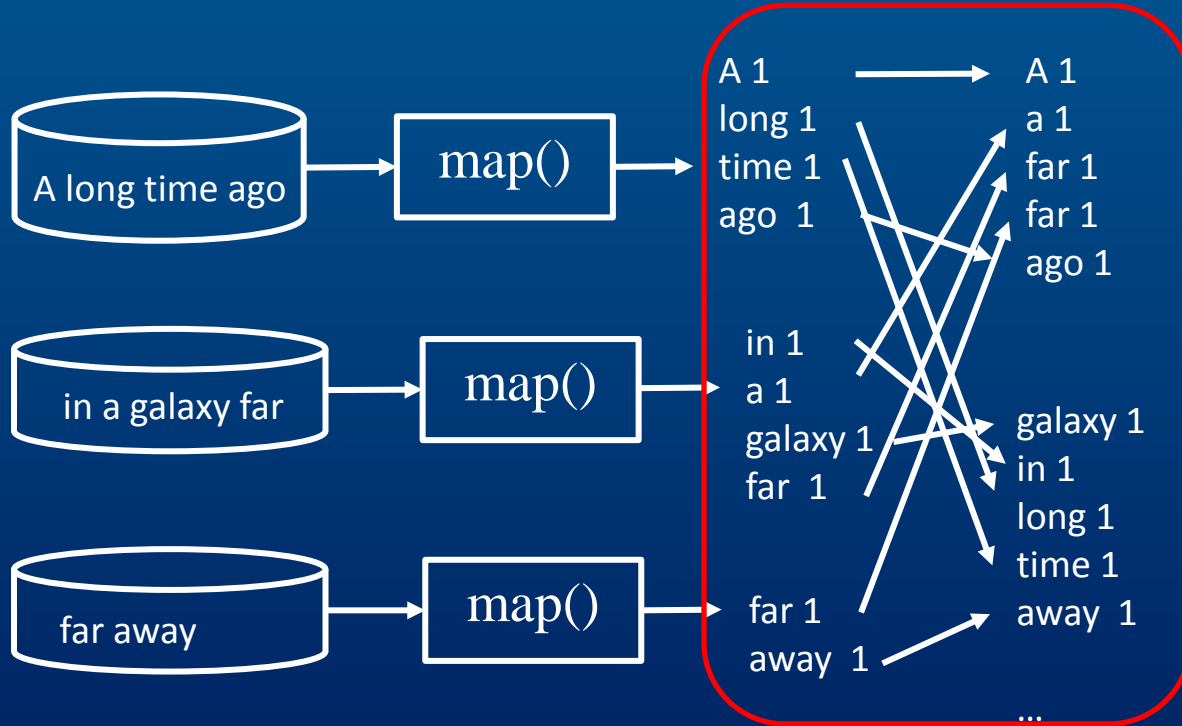


```
Get next <word><value>  
If <word> is same as previous word  
    add <value> to count  
else  
    emit <word> < count>  
    set count to 0
```

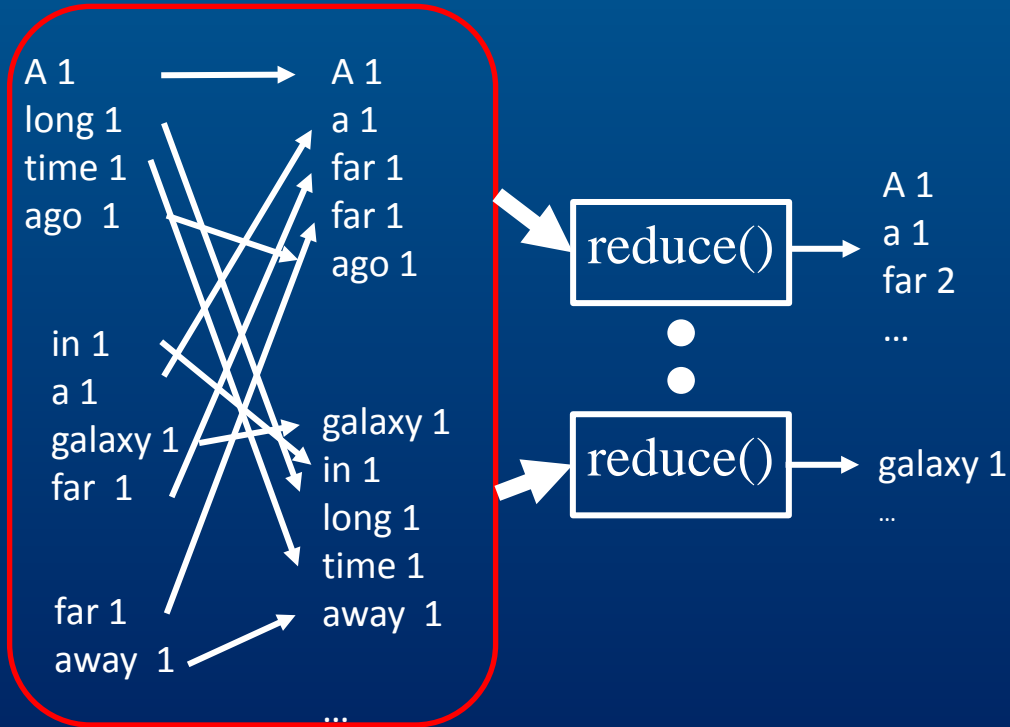
# map() output



# Hadoop shuffles, groups, and distributes



# reduce() aggregates



# Running Wordcount

```
> hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \  
  -input /user/cloudera/input \  
  -output /user/cloudera/output_new \  
  -mapper /home/cloudera/mapper.py \  
  -reducer /home/cloudera/reducer.py
```

functions



- Step 3: run streaming utility: