Computational Analysis of Big Data

Week 8

MapReduce

Upcoming deadlines

November 8, 2017, 23:59: Handin of assignment 2

November 9, 2017, in class: Project idea approvals

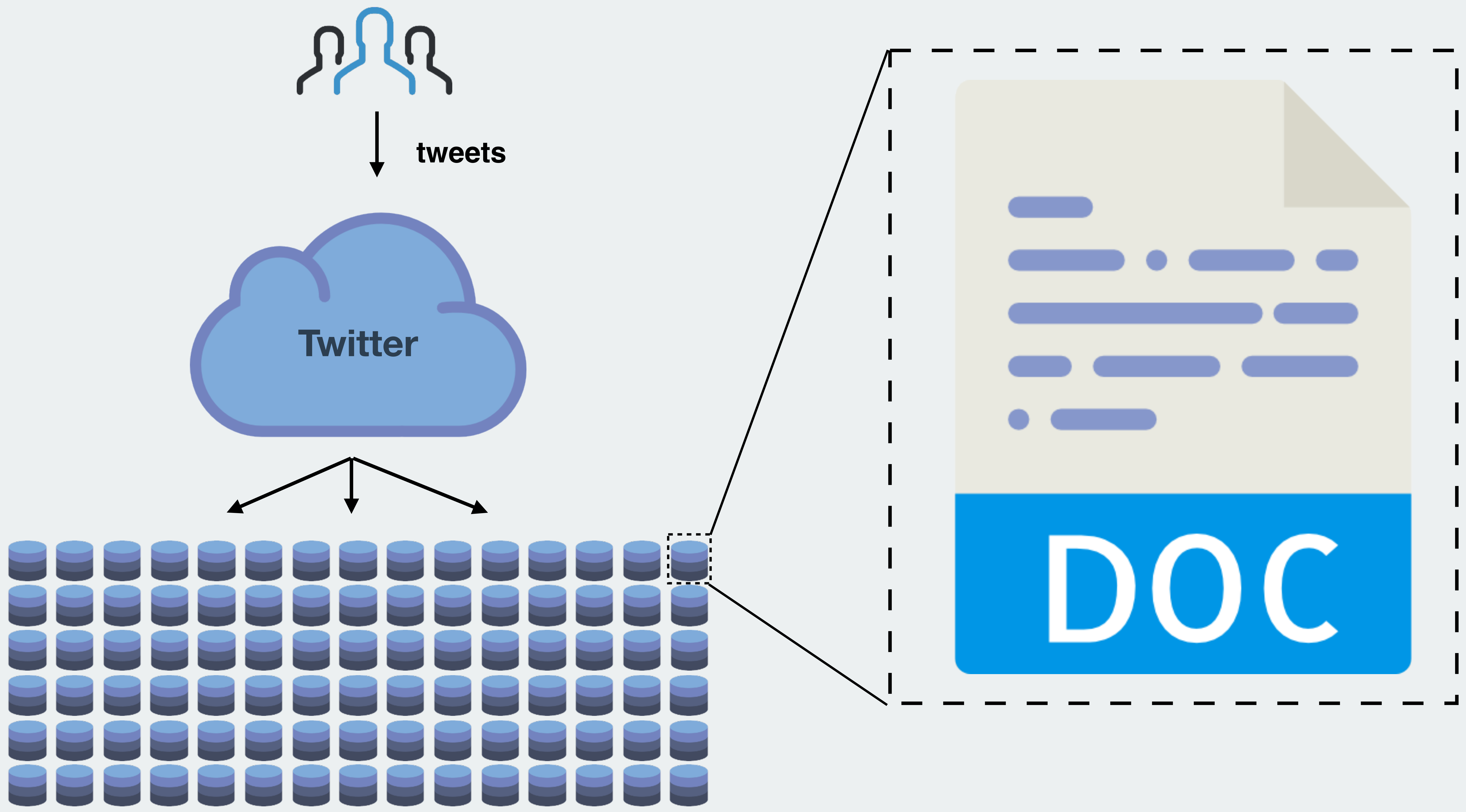
- Send your suggestions to me on email before class (short description)
- Individual talks in class for final approval

November 15, 2017, 23:59: Project A handin

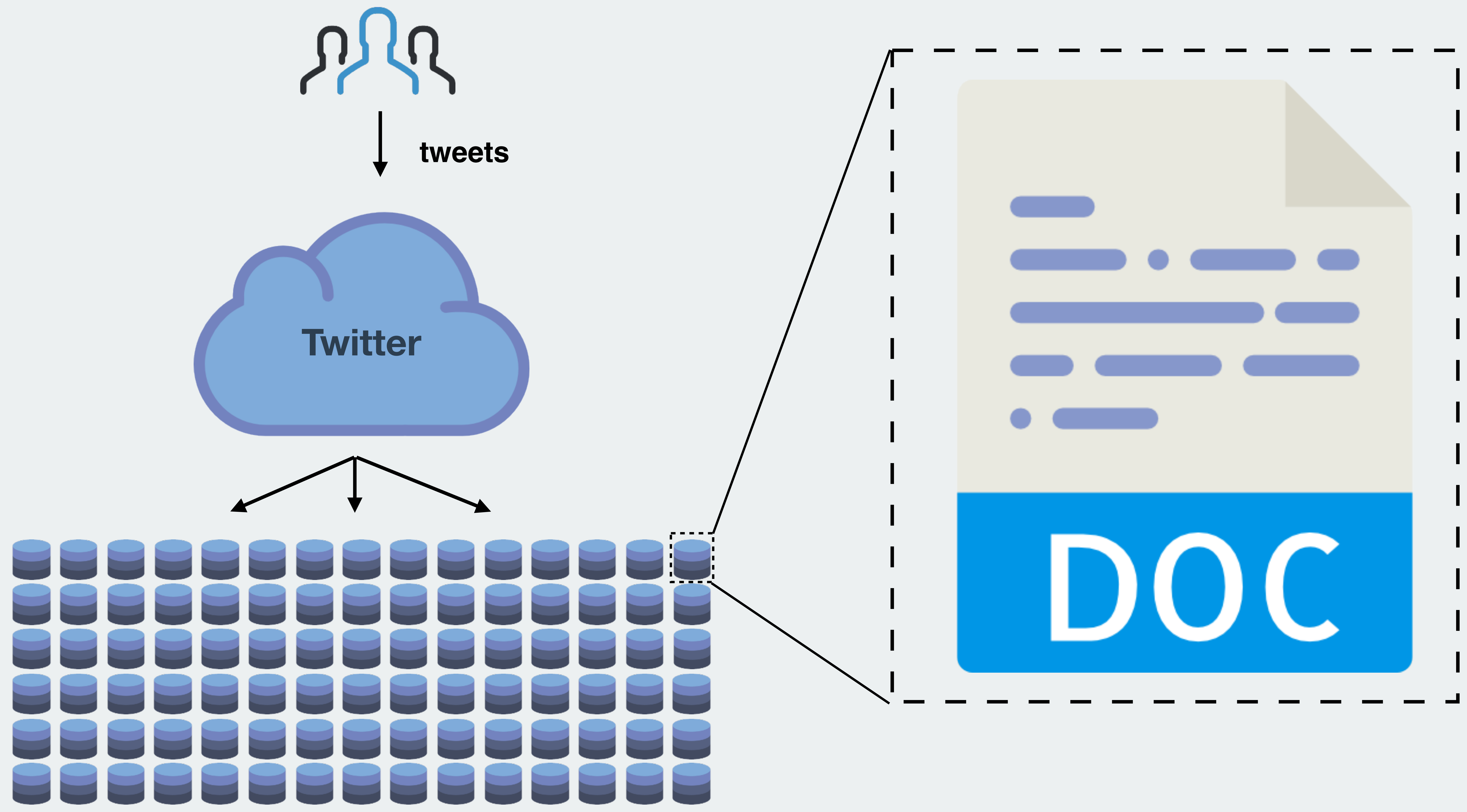
MapReduce

“MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster.”
- *Wikipedia*

Problem: Massive computation

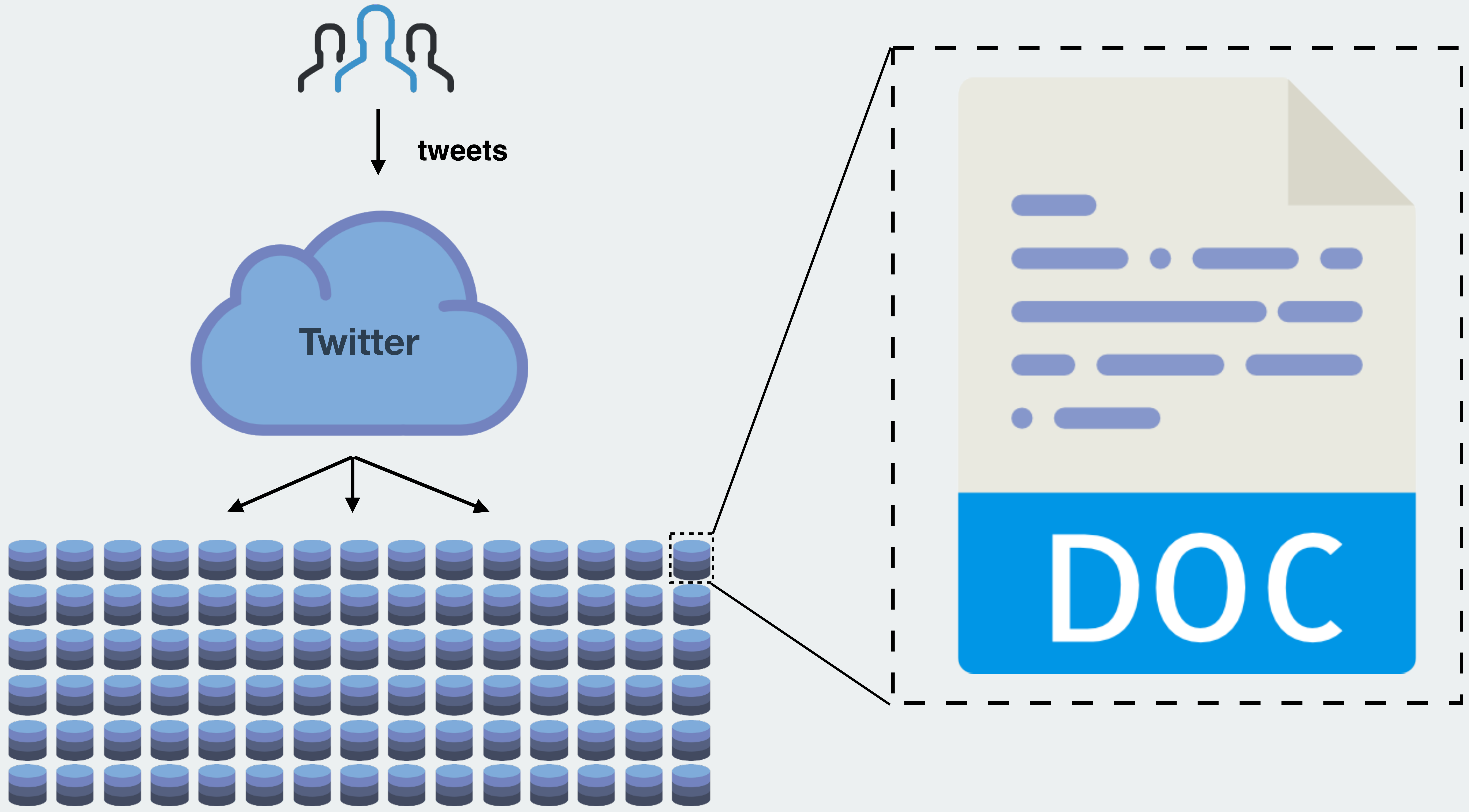


Problem: Massive computation



Objective: Count number of occurrences for each word used on twitter

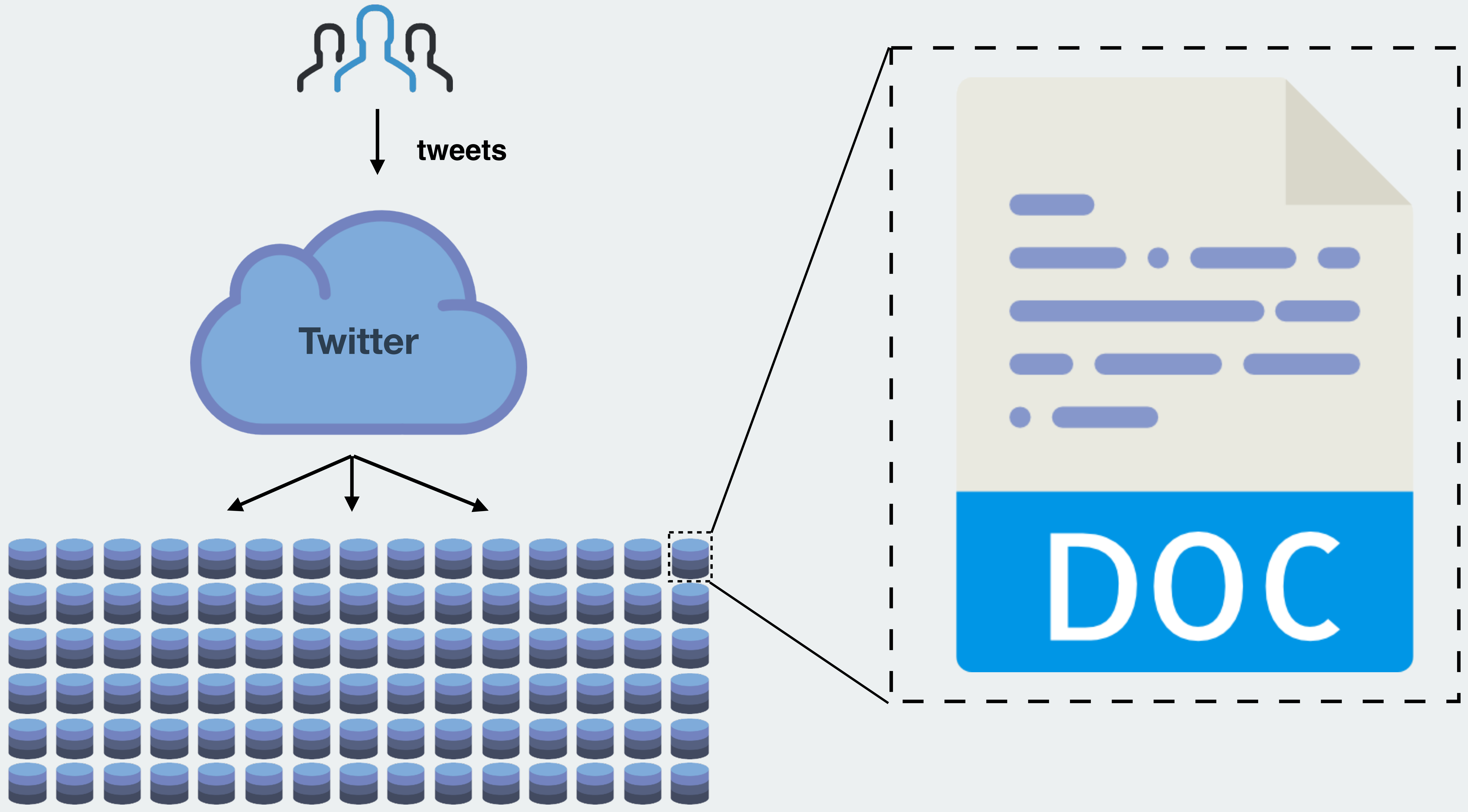
Problem: Massive computation



Objective: Count number of occurrences for each word used on twitter

Sequential: Set a Python script to loop over databases and for each word in each document, increment the word's `Counter`.

Problem: Massive computation

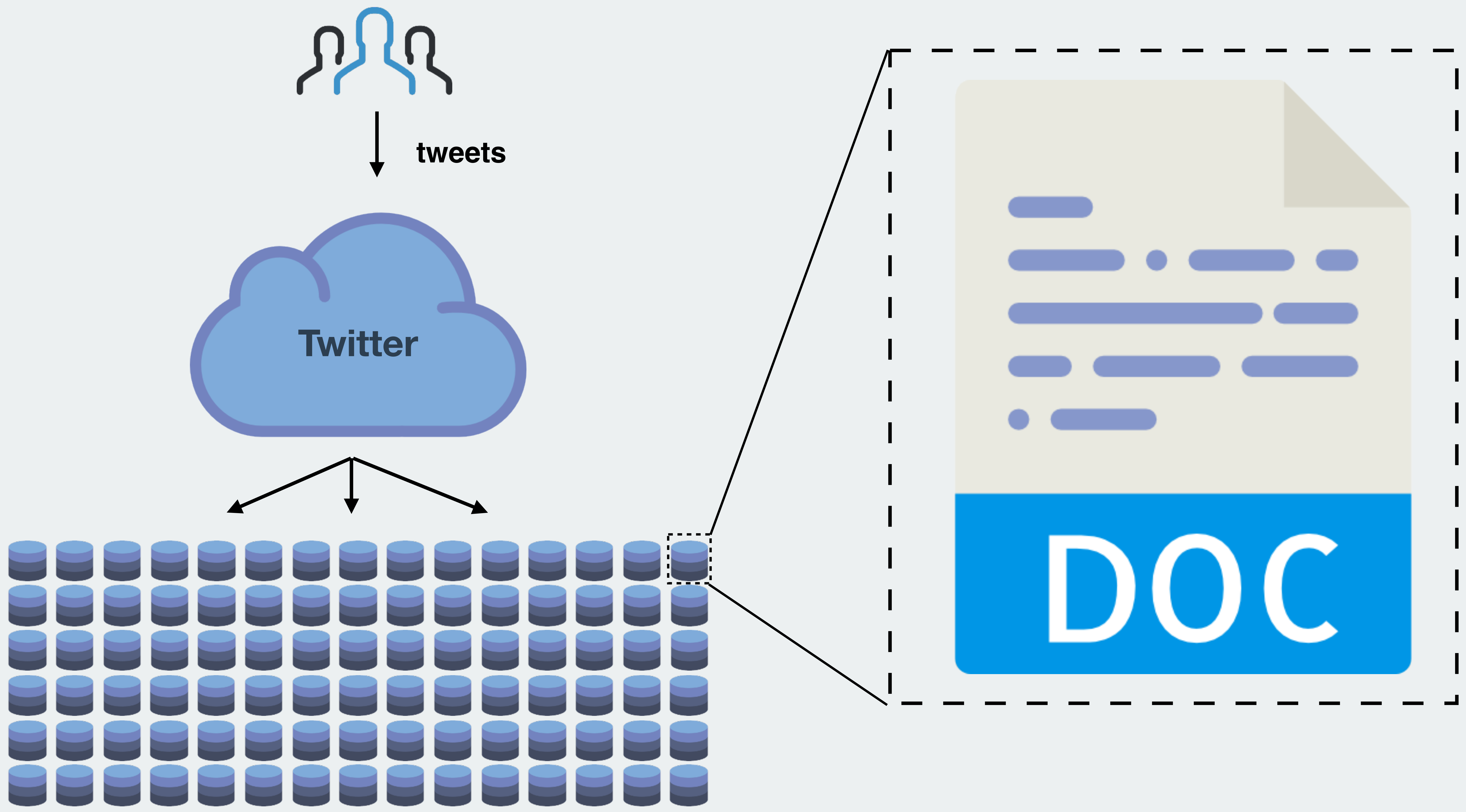


Objective: Count number of occurrences for each word used on twitter

Sequential: Set a Python script to loop over databases and for each word in each document, increment the word's `Counter`.

SLOW

Problem: Massive computation



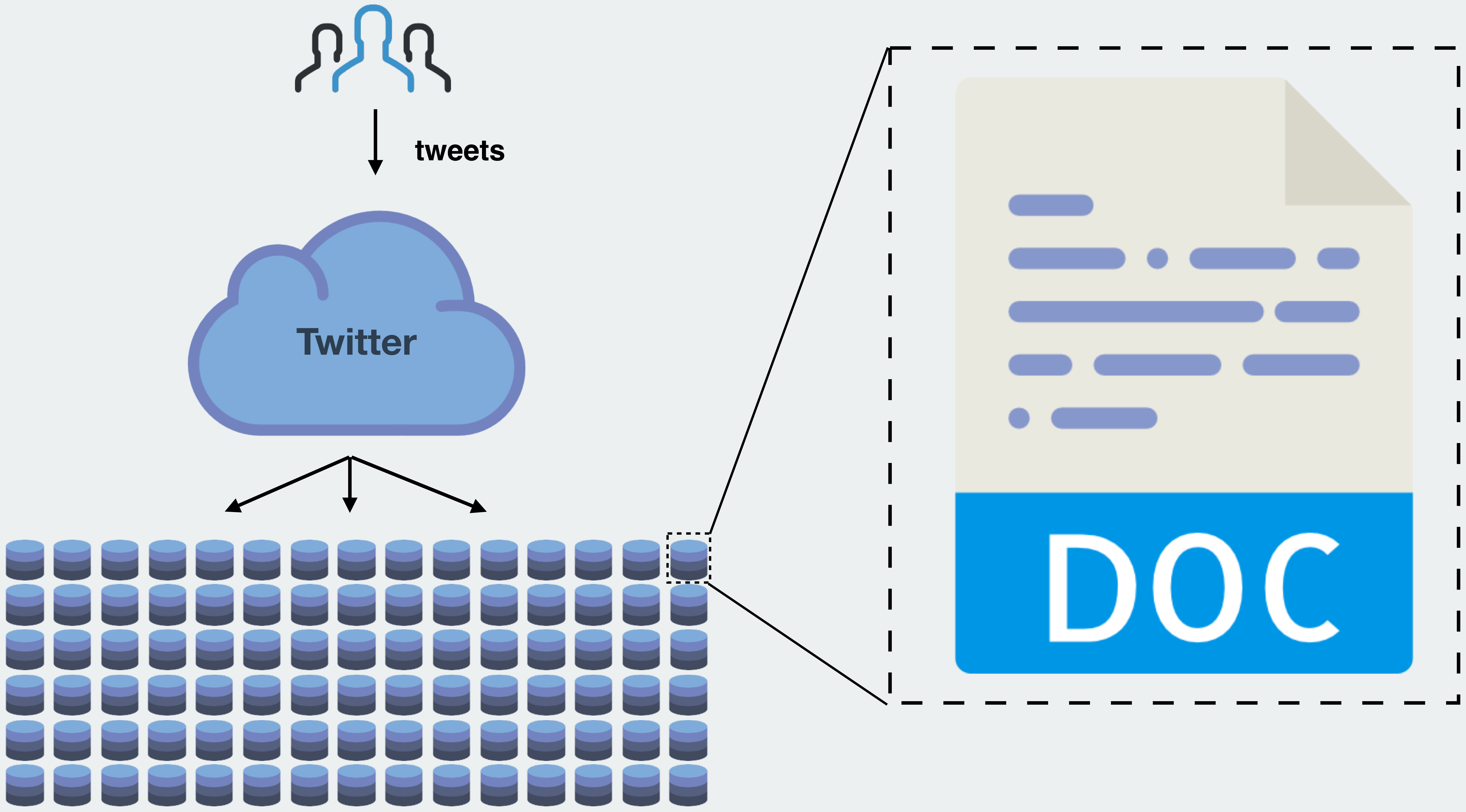
Objective: Count number of occurrences for each word used on twitter

Sequential: Set a Python script to loop over databases and for each word in each document, increment the word's `Counter`.

SLOW

Parallel: Send Python script to multiple databases at a time, which loops over words in the document and increments a central `Counter`.

Problem: Massive computation



Objective: Count number of occurrences for each word used on twitter

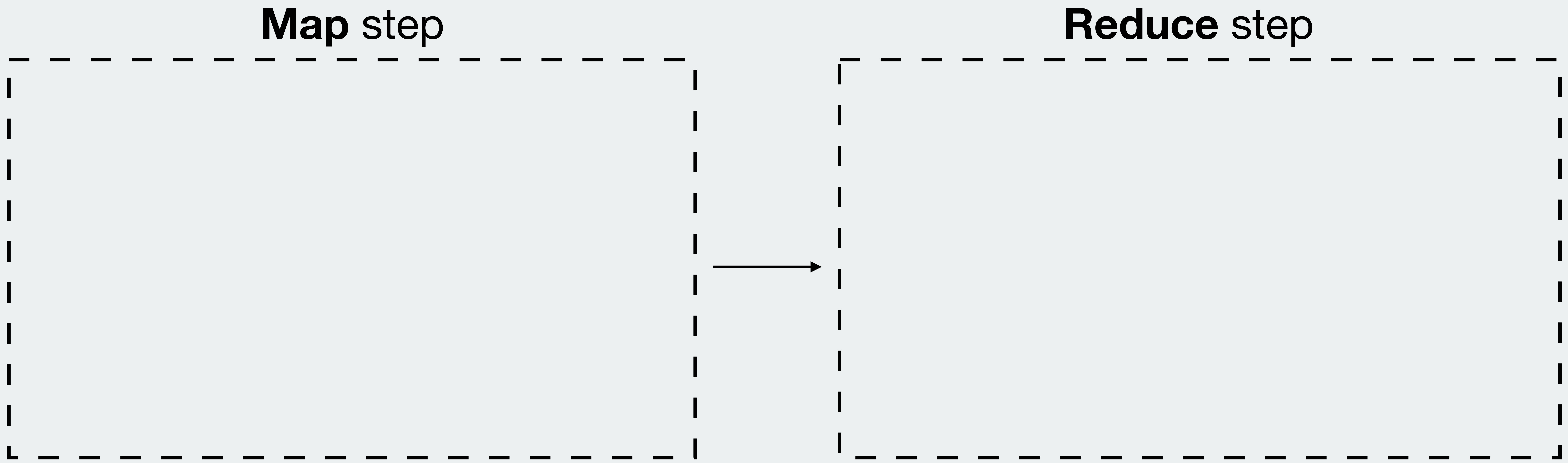
Sequential: Set a Python script to loop over databases and for each word in each document, increment the word's `Counter`.

SLOW

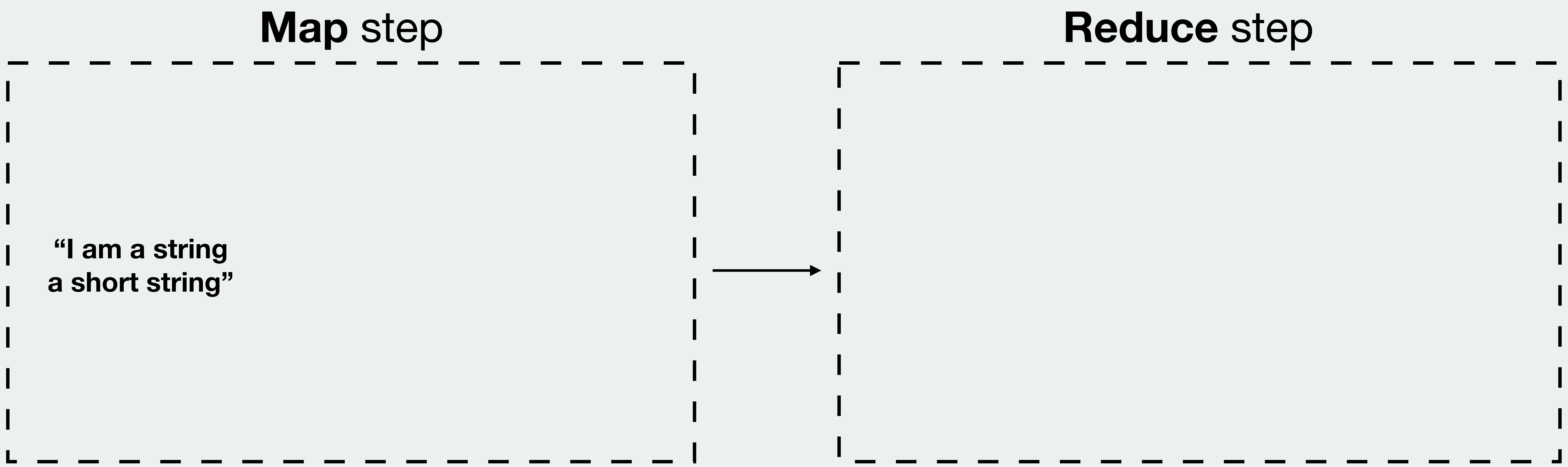
Parallel: Send Python script to multiple databases at a time, which loops over words in the document and increments a central `Counter`.

READ/WRITE ERRORS

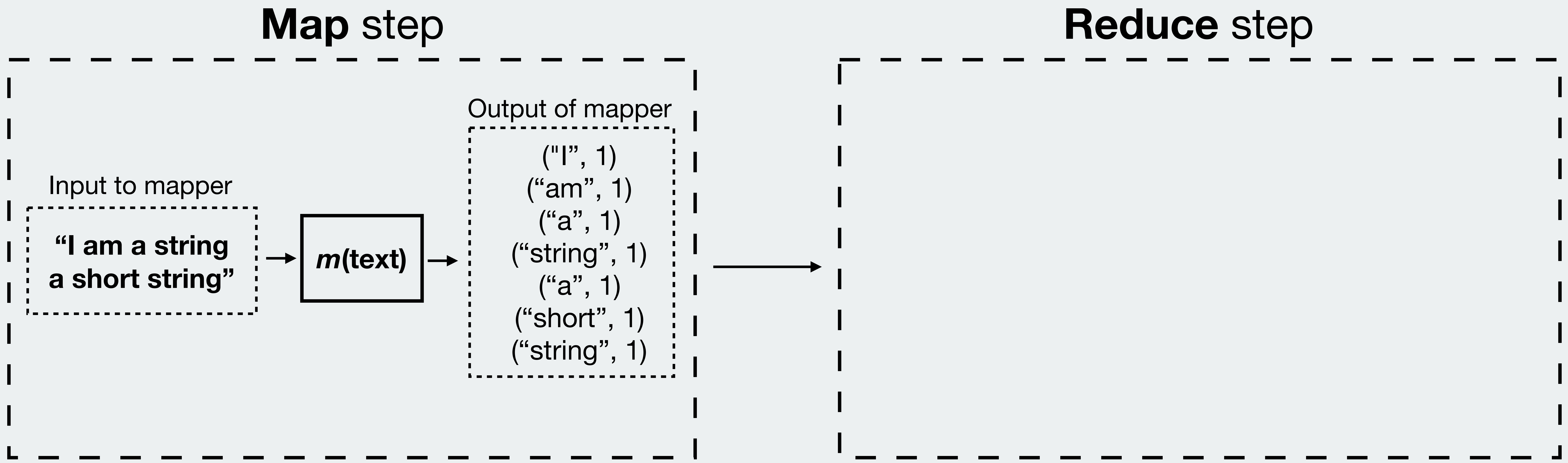
Solution: MapReduce



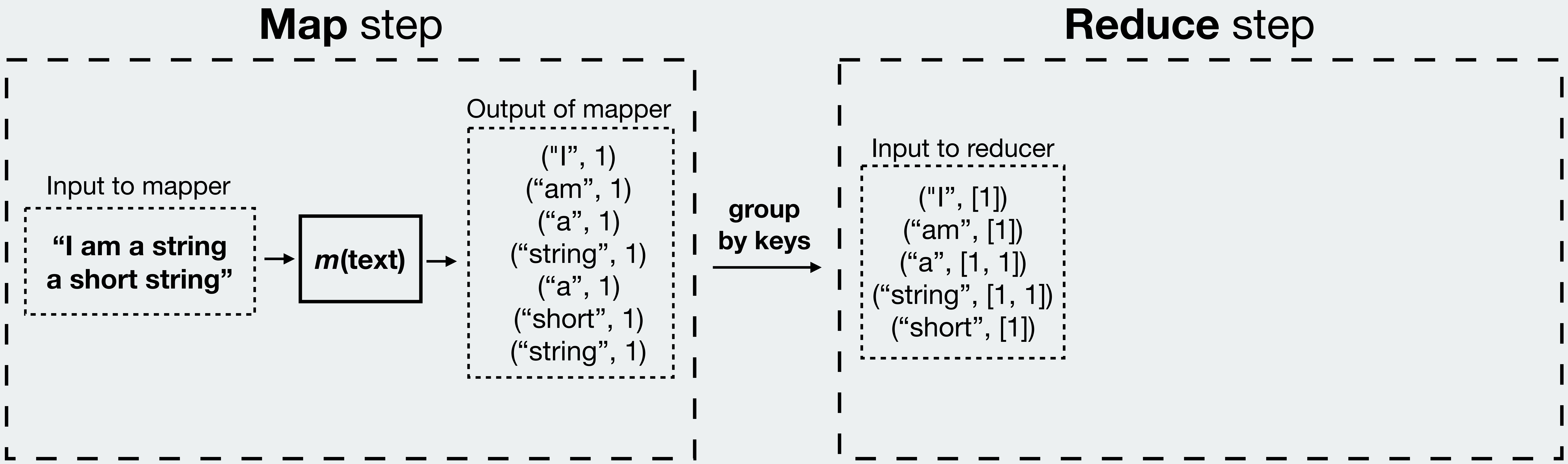
Solution: MapReduce



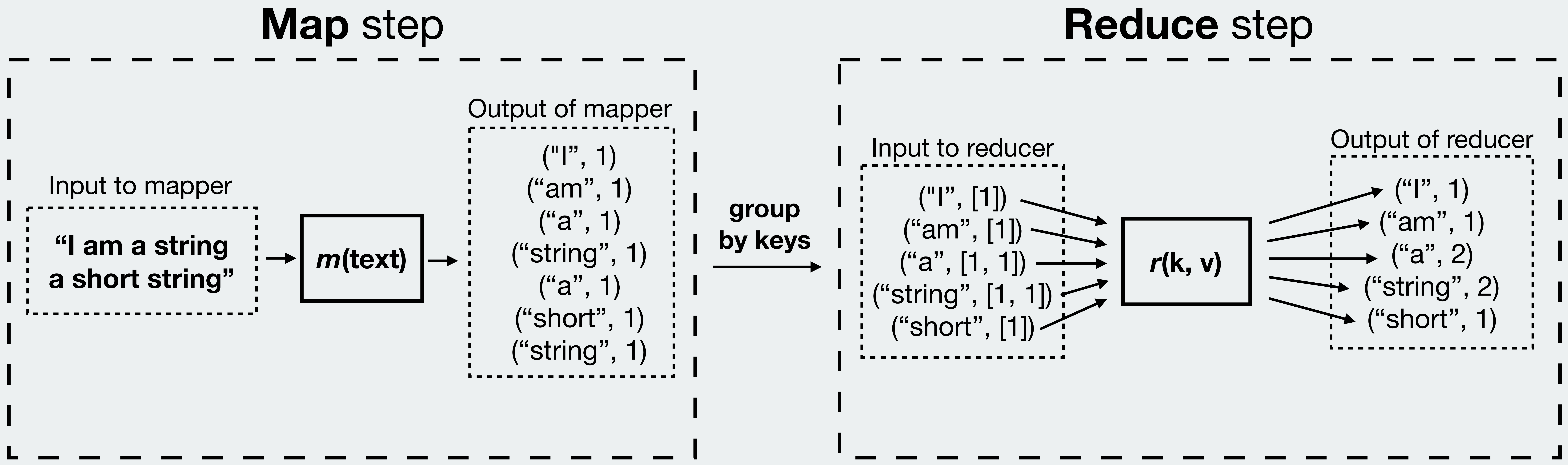
Solution: MapReduce



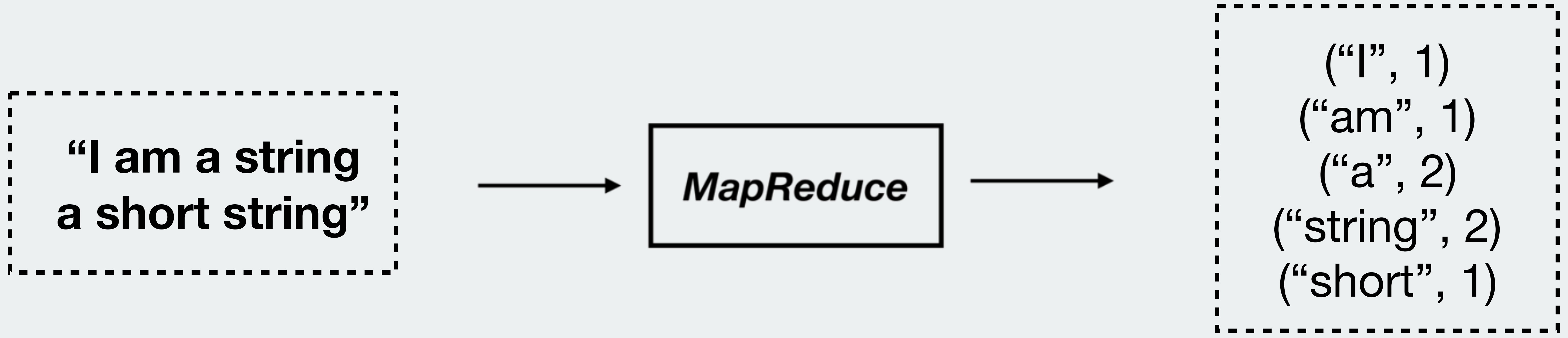
Solution: MapReduce



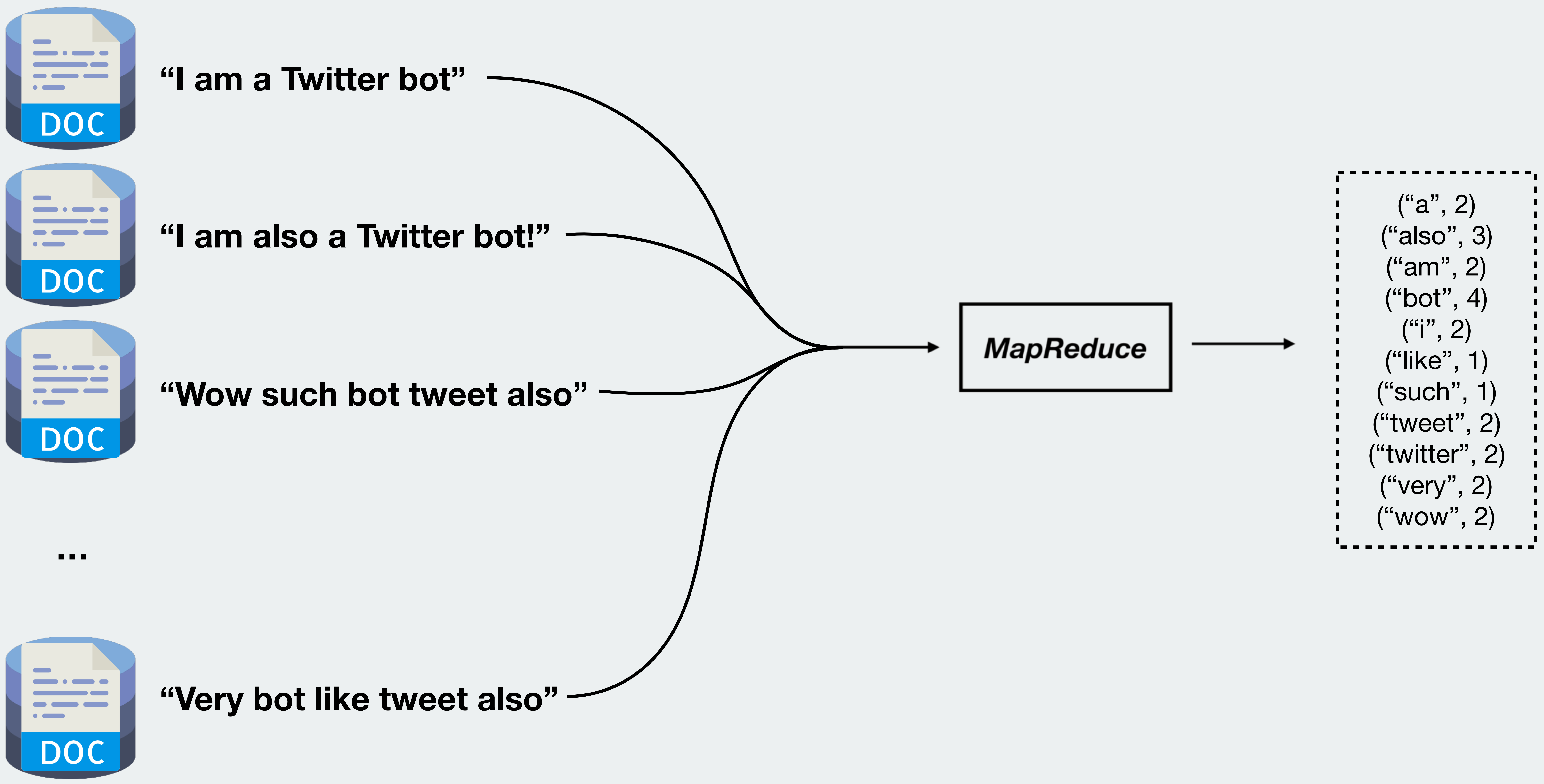
Solution: MapReduce



Solution: MapReduce



Word count with multiple documents



Map step

“I am a Twitter bot”

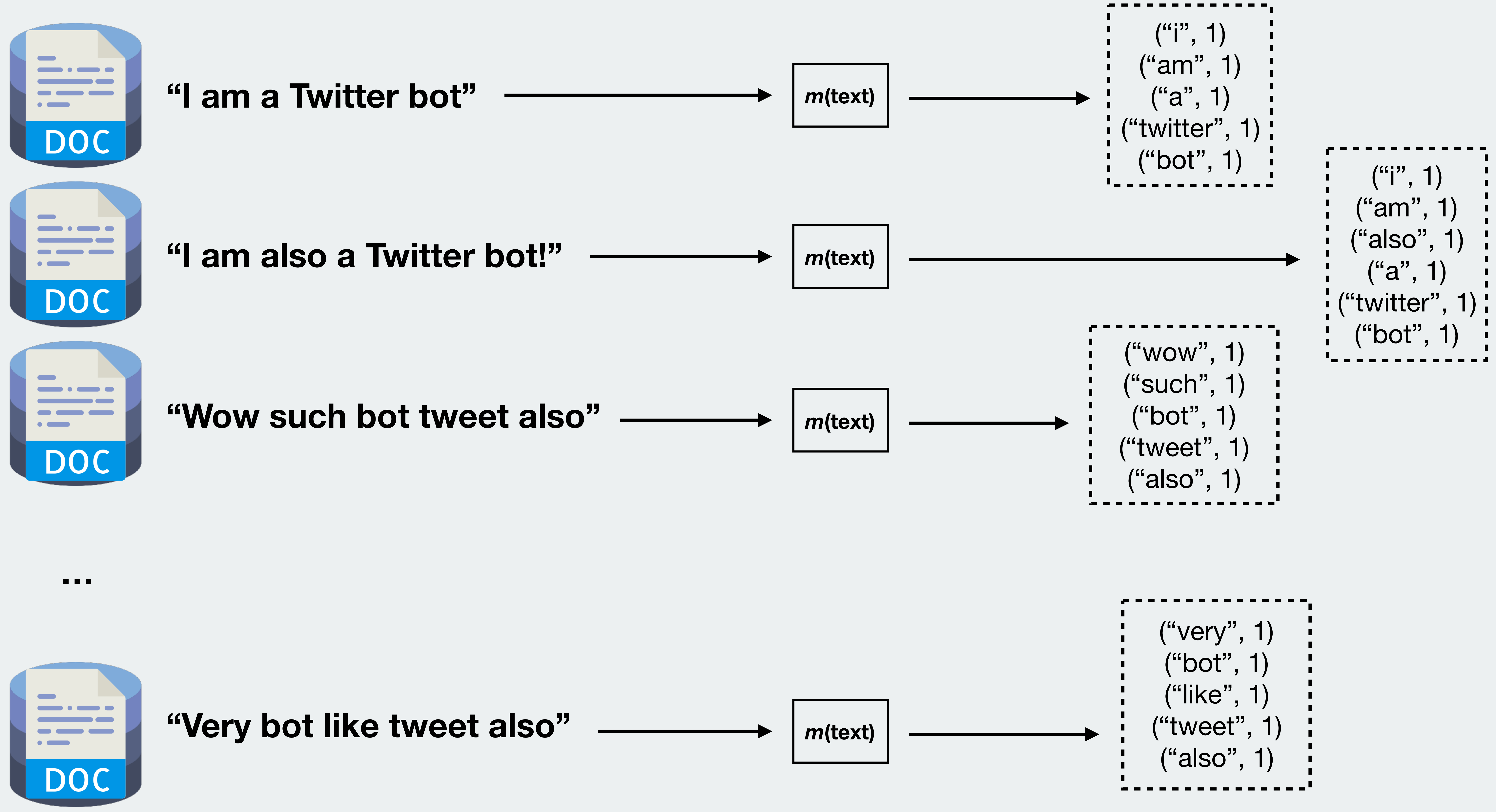
“I am also a Twitter bot!”

“Wow such bot tweet also”

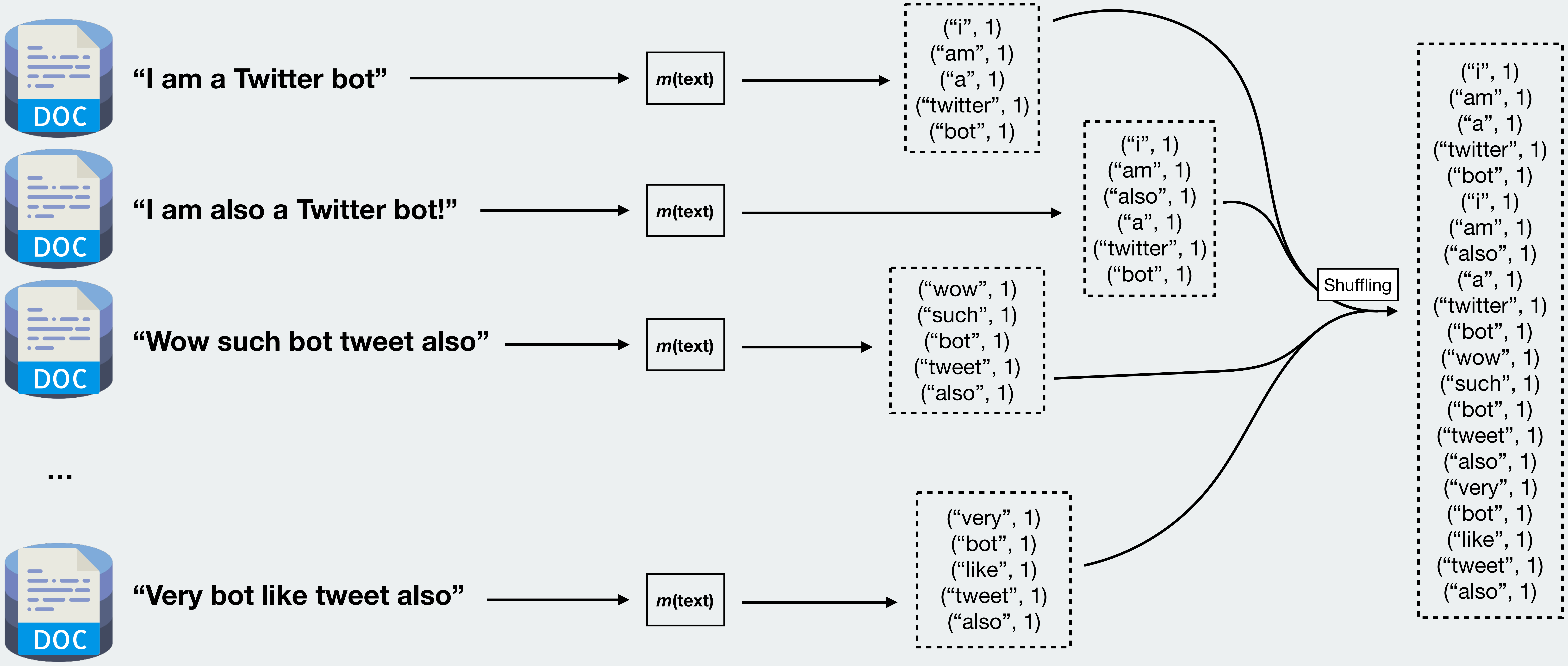
…

“Very bot like tweet also”

Map step


DOC
"I am a Twitter bot"
m(text)
("i", 1)
("am", 1)
("a", 1)
("twitter", 1)
("bot", 1)
DOC
"I am also a Twitter bot!"
m(text)
("i", 1)
("am", 1)
("also", 1)
("a", 1)
("twitter", 1)
("bot", 1)
DOC
"Wow such bot tweet also"
m(text)
("wow", 1)
("such", 1)
("bot", 1)
("tweet", 1)
("also", 1)
...
DOC
"Very bot like tweet also"
m(text)
("very", 1)
("bot", 1)
("like", 1)
("tweet", 1)
("also", 1)

Map step

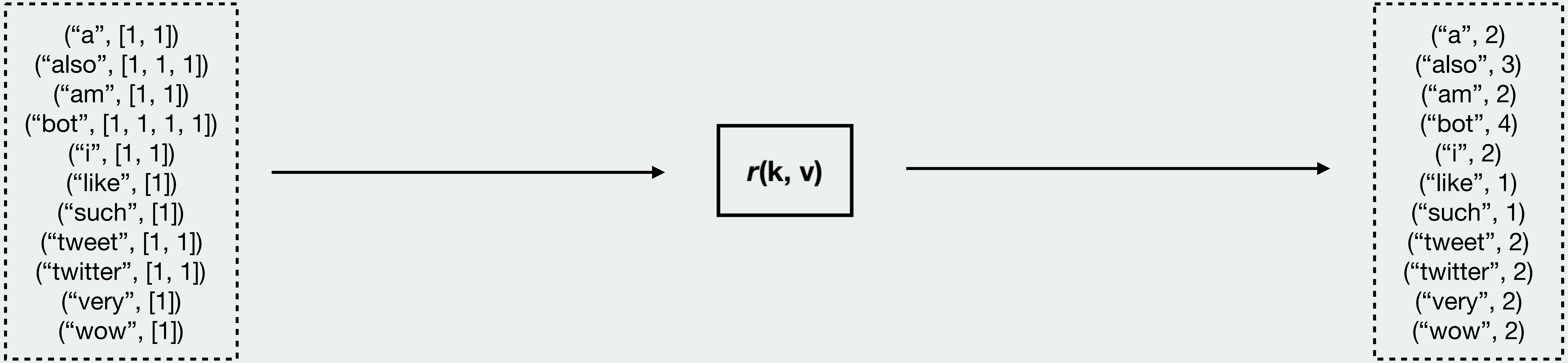

DOC
"I am a Twitter bot"
m(text)
("i", 1) ("am", 1) ("a", 1) ("twitter", 1) ("bot", 1)
DOC
"I am also a Twitter bot!"
m(text)
("i", 1) ("am", 1) ("also", 1) ("a", 1) ("twitter", 1) ("bot", 1)
DOC
"Wow such bot tweet also"
m(text)
("wow", 1) ("such", 1) ("bot", 1) ("tweet", 1) ("also", 1)
...
DOC
"Very bot like tweet also"
m(text)
("very", 1) ("bot", 1) ("like", 1) ("tweet", 1) ("also", 1)
Shuffling
("i", 1) ("am", 1) ("a", 1) ("twitter", 1) ("bot", 1) ("i", 1) ("am", 1) ("also", 1) ("a", 1) ("twitter", 1) ("bot", 1) ("wow", 1) ("such", 1) ("bot", 1) ("tweet", 1) ("also", 1) ("very", 1) ("bot", 1) ("like", 1) ("tweet", 1) ("also", 1)

```
("i", 1)
("am", 1)
("a", 1)
("twitter", 1)
("bot", 1)
("i", 1)
("am", 1)
("also", 1)
("a", 1)
("twitter", 1)
("bot", 1)
("wow", 1)
("such", 1)
("bot", 1)
("tweet", 1)
("also", 1)
("very", 1)
("bot", 1)
("like", 1)
("tweet", 1)
("also", 1)
```

$$\xrightarrow{\textbf{group by keys}}$$

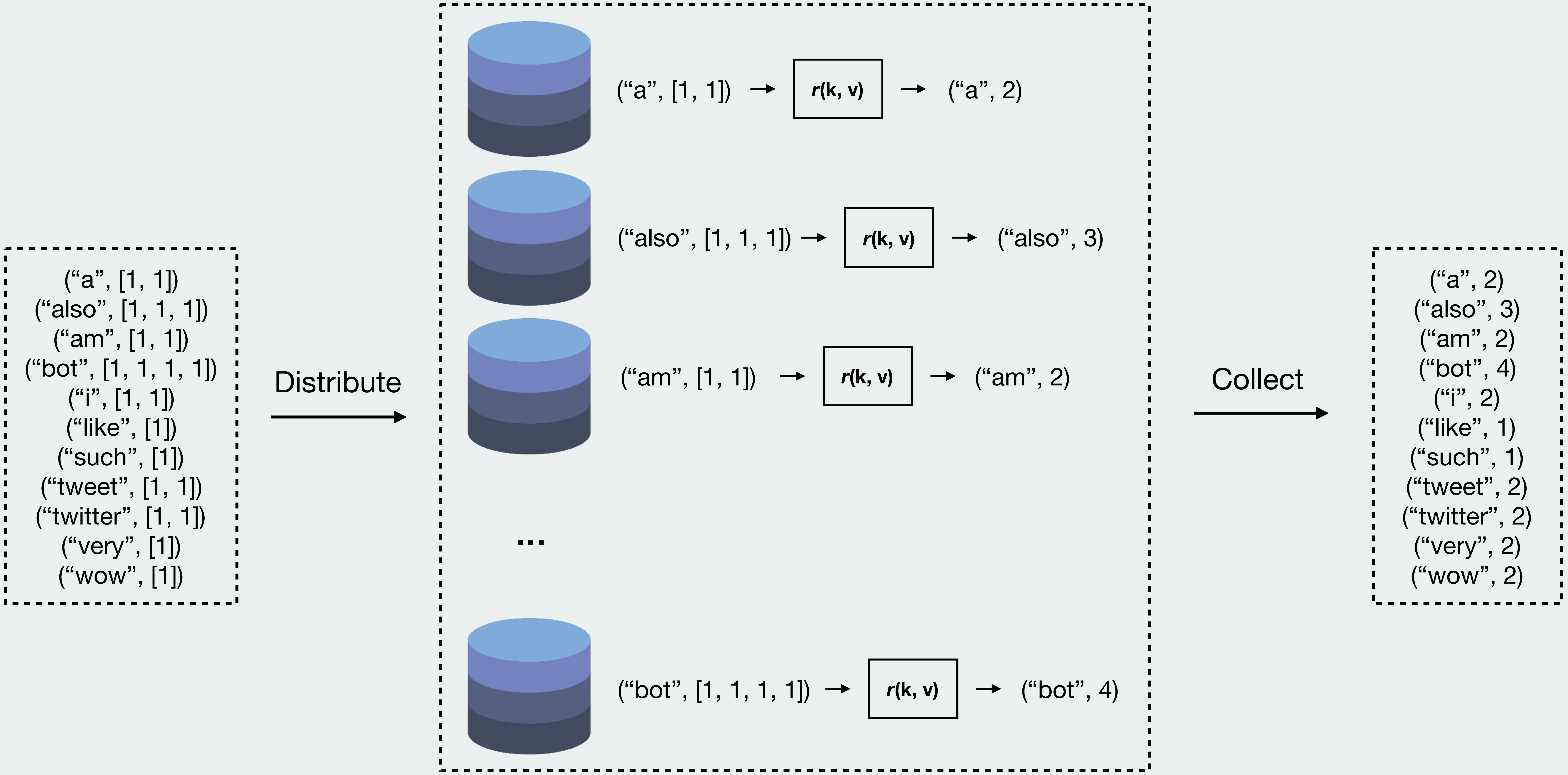
```
("a", [1, 1])
("also", [1, 1, 1])
("am", [1, 1])
("bot", [1, 1, 1, 1])
("i", [1, 1])
("like", [1])
("such", [1])
("tweet", [1, 1])
("twitter", [1, 1])
("very", [1])
("wow", [1])
```

Reduce step


(“a”, [1, 1])
(“also”, [1, 1, 1])
(“am”, [1, 1])
(“bot”, [1, 1, 1, 1])
(“i”, [1, 1])
(“like”, [1])
(“such”, [1])
(“tweet”, [1, 1])
(“twitter”, [1, 1])
(“very”, [1])
(“wow”, [1])

(“a”, 2)
(“also”, 3)
(“am”, 2)
(“bot”, 4)
(“i”, 2)
(“like”, 1)
(“such”, 1)
(“tweet”, 2)
(“twitter”, 2)
(“very”, 2)
(“wow”, 2)

Reduce step

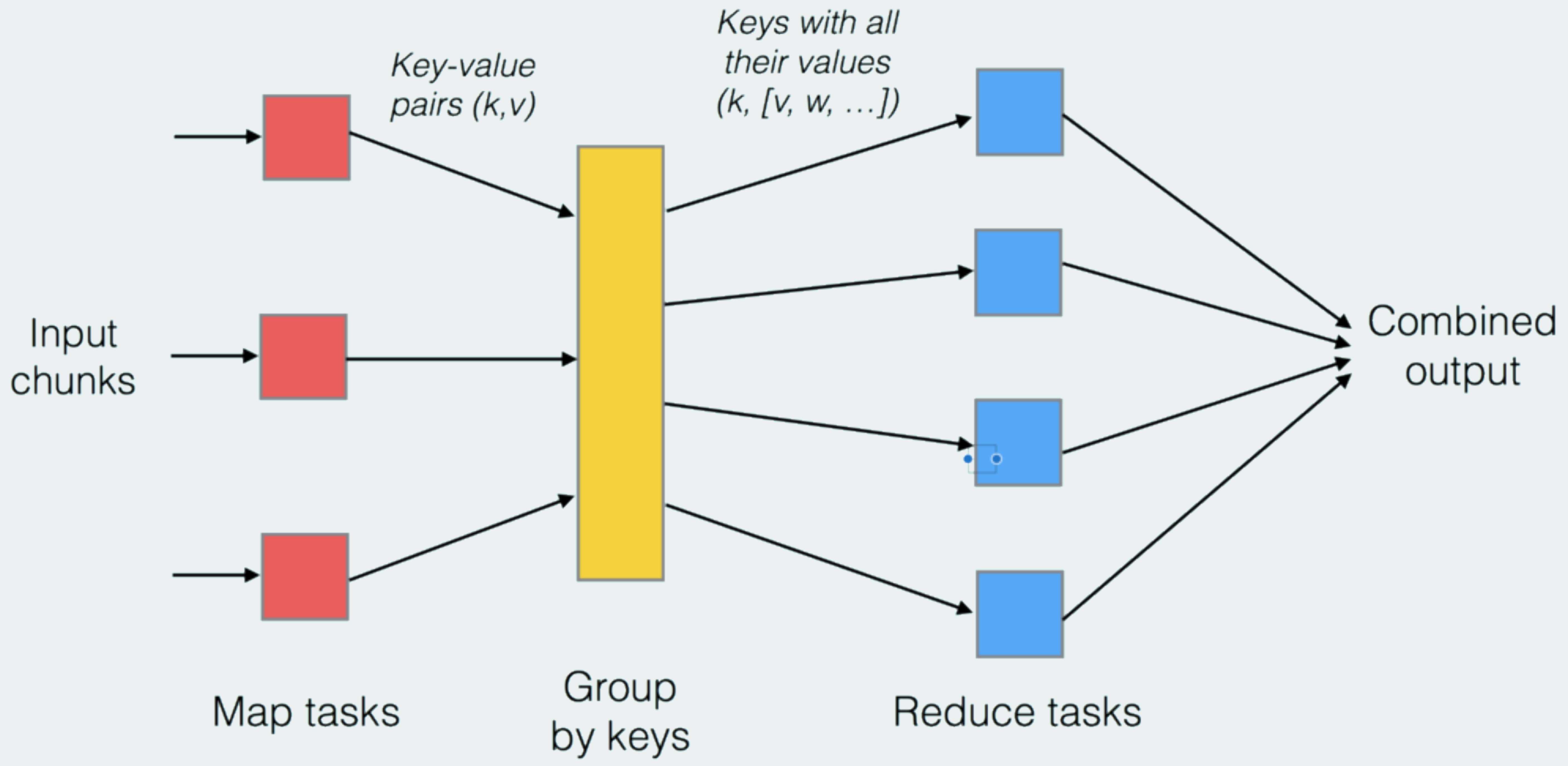

("a", [1, 1])
("also", [1, 1, 1])
("am", [1, 1])
("bot", [1, 1, 1, 1])
("i", [1, 1])
("like", [1])
("such", [1])
("tweet", [1, 1])
("twitter", [1, 1])
("very", [1])
("wow", [1])
Distribute
("a", [1, 1]) → r(k, v) → ("a", 2)
("also", [1, 1, 1]) → r(k, v) → ("also", 3)
("am", [1, 1]) → r(k, v) → ("am", 2)
...
("bot", [1, 1, 1, 1]) → r(k, v) → ("bot", 4)
Collect
("a", 2)
("also", 3)
("am", 2)
("bot", 4)
("i", 2)
("like", 1)
("such", 1)
("tweet", 2)
("twitter", 2)
("very", 2)
("wow", 2)

Summary

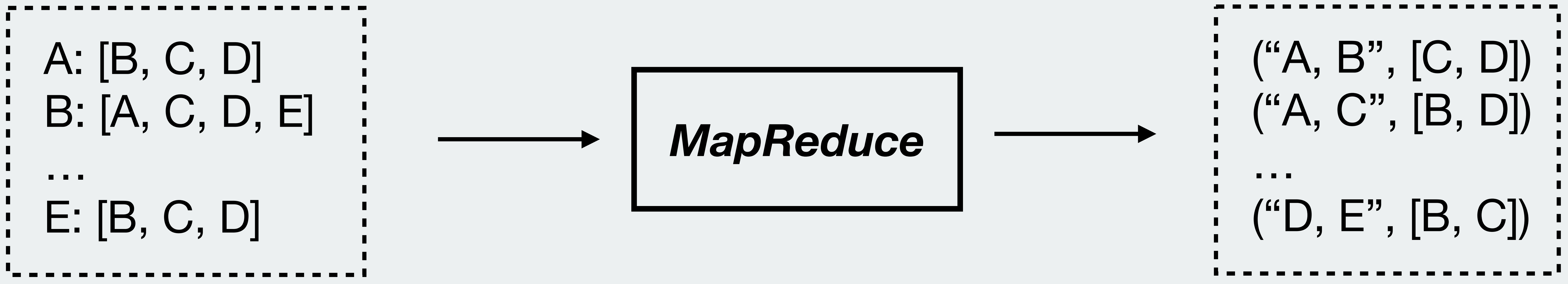
1. Map input **key-value** pairs

2. Group values by their keys

3. Perform an operation on each key's list of values



Advanced example: Common friends



Input: Friend lists

Output: For each pair of friends, a list of common friends

Map step

A: [B, C, D]
B: [A, C, D, E]
C: [A, B, D, E]
D: [A, B, C, E]
E: [B, C, D]

(Key , Value)

(“A, B”, [B, C, D])
(“A, C”, [B, C, D])
(“A, D”, [B, C, D])

(“A, B”, [A, C, D, E])
(“B, C”, [A, C, D, E])
(“B, D”, [A, C, D, E])
(“B, E”, [A, C, D, E])

(“A, C”, [A, B, D, E])
(“B, C”, [A, B, D, E])
(“C, D”, [A, B, D, E])
(“C, E”, [A, B, D, E])

(“A, D”, [A, B, C, E])
(“B, D”, [A, B, C, E])
(“C, D”, [A, B, C, E])
(“D, E”, [A, B, C, E])

(“B, E”, [B, C, D])
(“C, E”, [B, C, D])
(“D, E”, [B, C, D])

Group by key

(Key , Value)

(“A, B”, [B, C, D])
(“A, C”, [B, C, D])
(“A, D”, [B, C, D])
(“A, B”, [A, C, D, E])
(“B, C”, [A, C, D, E])
(“B, D”, [A, C, D, E])
(“B, E”, [A, C, D, E])
(“A, C”, [A, B, D, E])
(“B, C”, [A, B, D, E])
(“C, D”, [A, B, D, E])
(“C, E”, [A, B, D, E])
(“A, D”, [A, B, C, E])
(“B, D”, [A, B, C, E])
(“C, D”, [A, B, C, E])
(“D, E”, [A, B, C, E])
(“B, E”, [B, C, D])
(“C, E”, [B, C, D])
(“D, E”, [B, C, D])

$\longrightarrow$

(Key , [Value1, Value2])

(“A, B”, [[B, C, D], [A, C, D, E]])
(“A, C”, [[A, C, D, E], [A, B, D, E]])
(“A, D”, [[B, C, D], [A, B, C, E]])
(“B, C”, [[A, C, D, E], [A, B, D, E]])
(“B, D”, [[A, C, D, E], [A, B, C, E]])
(“B, E”, [[A, C, D, E], [B, C, D]])
(“C, D”, [[A, B, D, E], [A, B, C, E]])
(“C, E”, [[A, B, D, E], [B, C, D]])
(“D, E”, [[A, B, C, E], [B, C, D]])

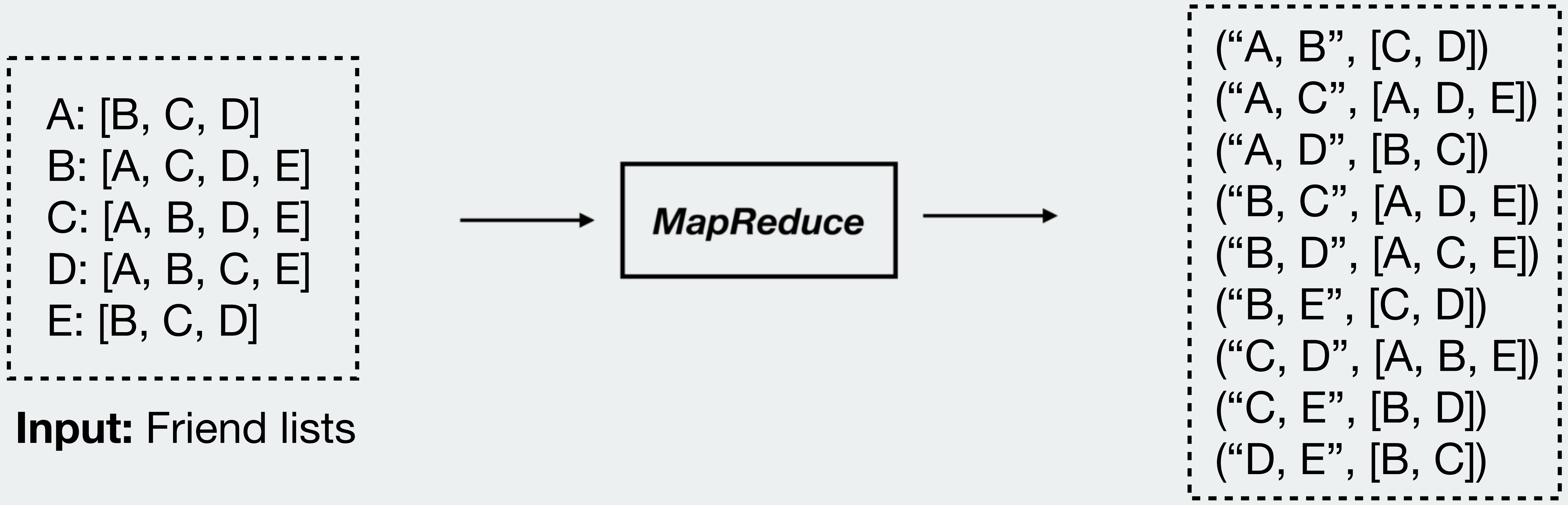
Reduce step

(“A, B”, [[B, **C**, **D**], [A, **C**, **D**, E]])
(“A, C”, [[**A**, C, **D**, **E**], [**A**, B, **D**, **E**]])
(“A, D”, [[**B**, **C**, D], [A, **B**, **C**, E]])
(“B, C”, [[**A**, C, **D**, **E**], [**A**, B, **D**, **E**]])
(“B, D”, [[**A**, **C**, D, **E**], [**A**, B, **C**, **E**]])
(“B, E”, [[A, **C**, **D**, E], [B, **C**, **D**]])
(“C, D”, [[**A**, **B**, D, **E**], [**A**, **B**, C, **E**]])
(“C, E”, [[A, **B**, **D**, E], [**B**, C, **D**]])
(“D, E”, [[A, **B**, **C**, E], [**B**, **C**, D]])

$\xrightarrow{\textbf{Intersection}}$

(“A, B”, [C, D])
(“A, C”, [A, D, E])
(“A, D”, [B, C])
(“B, C”, [A, D, E])
(“B, D”, [A, C, E])
(“B, E”, [C, D])
(“C, D”, [A, B, E])
(“C, E”, [B, D])
(“D, E”, [B, C])

Advanced example: Common friends



A: [B, C, D]
B: [A, C, D, E]
C: [A, B, D, E]
D: [A, B, C, E]
E: [B, C, D]

Input: Friend lists

(“A, B”, [C, D])
(“A, C”, [A, D, E])
(“A, D”, [B, C])
(“B, C”, [A, D, E])
(“B, D”, [A, C, E])
(“B, E”, [C, D])
(“C, D”, [A, B, E])
(“C, E”, [B, D])
(“D, E”, [B, C])

Output: For each pair of friends, a list of common friends

MapReduce in Python: `mrjob`

- Python package that lets you write MapReduce jobs in pure Python.
- Runs on your local machine as well as a Hadoop cluster
- Can also be used to write Spark jobs.

```
from mrjob.job import MRJob

class MRWordCounter(MRJob):

    def mapper(self, _, line):
        for word in line.split():
            yield word, 1

    def reducer(self, key, values):
        yield key, sum(values)

if __name__ == '__main__':
    MRWordCounter.run()
```

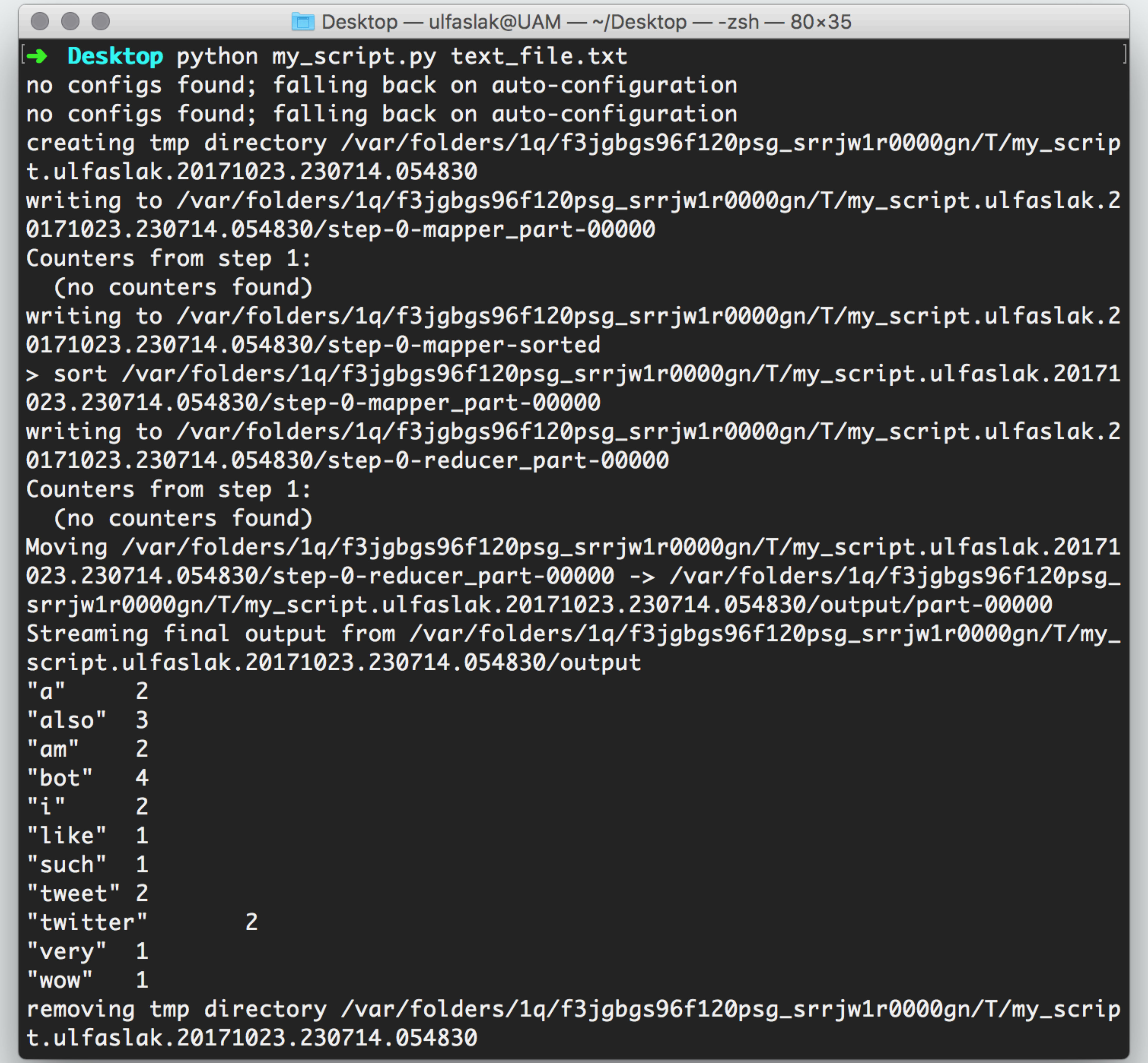
MapReduce in Python: `mrjob`

```python
from mrjob.job import MRJob

class MRWordCounter(MRJob):

    def mapper(self, _, line):
        for word in line.split():
            yield word, 1

    def reducer(self, key, values):
        yield key, sum(values)

if __name__ == '__main__':
    MRWordCounter.run()
```

my_script.py

```
i am a twitter bot
i am also a twitter bot
wow such bot tweet also
very bot like tweet also
```

text_file.txt



```
➜  Desktop python my_script.py text_file.txt
no configs found; falling back on auto-configuration
no configs found; falling back on auto-configuration
creating tmp directory /var/folders/1q/f3jgbgs96f120psg_srrjw1r0000gn/T/my_scrip
t.ulfaslak.20171023.230714.054830
writing to /var/folders/1q/f3jgbgs96f120psg_srrjw1r0000gn/T/my_script.ulfaslak.2
0171023.230714.054830/step-0-mapper_part-00000
Counters from step 1:
  (no counters found)
writing to /var/folders/1q/f3jgbgs96f120psg_srrjw1r0000gn/T/my_script.ulfaslak.2
0171023.230714.054830/step-0-mapper-sorted
> sort /var/folders/1q/f3jgbgs96f120psg_srrjw1r0000gn/T/my_script.ulfaslak.20171
023.230714.054830/step-0-mapper_part-00000
writing to /var/folders/1q/f3jgbgs96f120psg_srrjw1r0000gn/T/my_script.ulfaslak.2
0171023.230714.054830/step-0-reducer_part-00000
Counters from step 1:
  (no counters found)
Moving /var/folders/1q/f3jgbgs96f120psg_srrjw1r0000gn/T/my_script.ulfaslak.20171
023.230714.054830/step-0-reducer_part-00000 -> /var/folders/1q/f3jgbgs96f120psg_
srrjw1r0000gn/T/my_script.ulfaslak.20171023.230714.054830/output/part-00000
Streaming final output from /var/folders/1q/f3jgbgs96f120psg_srrjw1r0000gn/T/my_
script.ulfaslak.20171023.230714.054830/output
"a"     2
"also"  3
"am"    2
"bot"   4
"i"     2
"like"  1
"such"  1
"tweet" 2
"twitter"       2
"very"  1
"wow"   1
removing tmp directory /var/folders/1q/f3jgbgs96f120psg_srrjw1r0000gn/T/my_scrip
t.ulfaslak.20171023.230714.054830
```