Computational Tools for Big Data

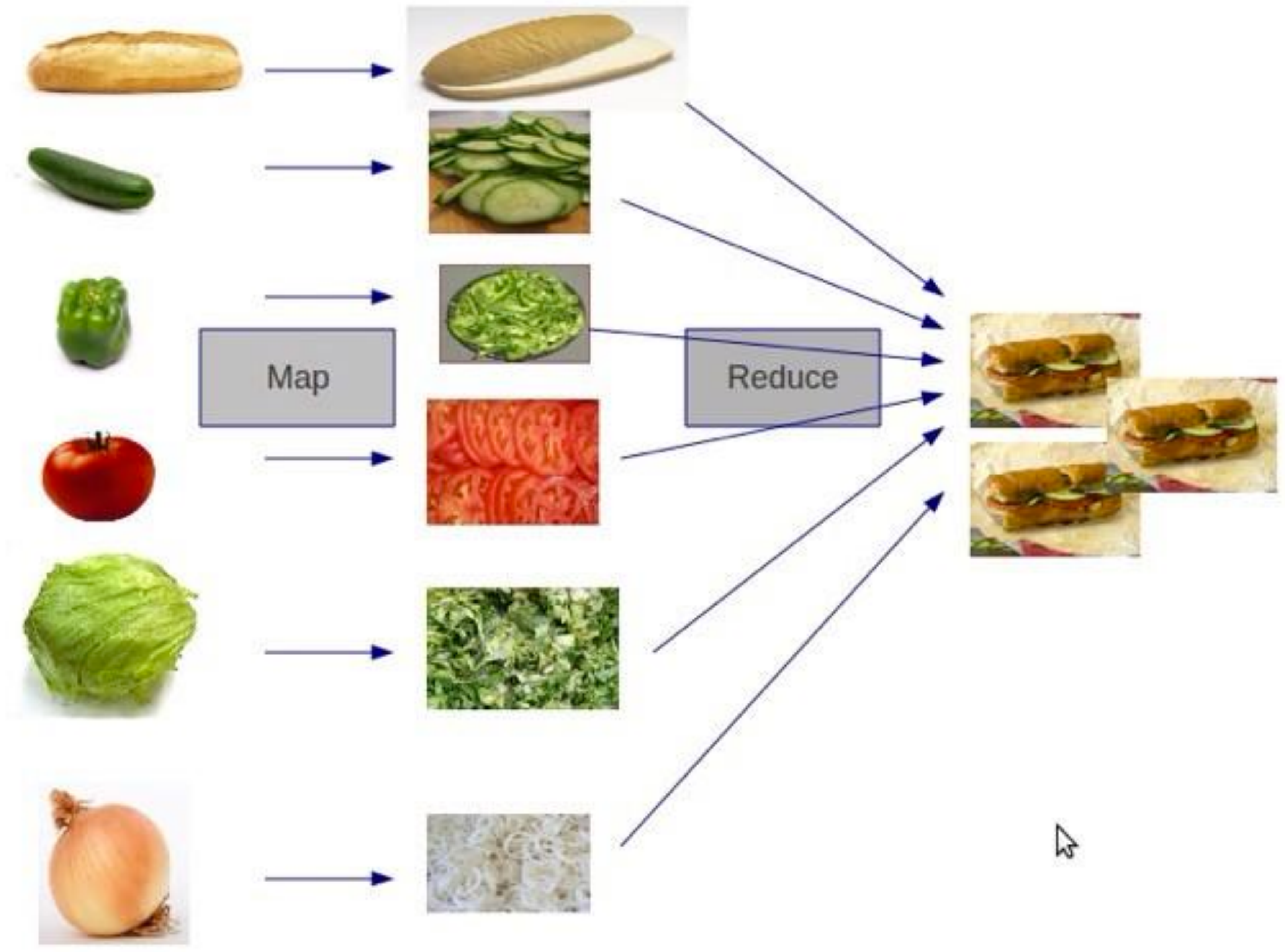
Week 8: MapReduce

What is MapReduce?

"MapReduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster."

- *Wikipedia*

Why MapReduce?



What is MapReduce?

Map

Group

Distribute

Reduce

Combine

Example: Word count

Input

- Documents of words

Output

- Number of occurrences of each word

Example: Word count

Map task

Key Value

Document 1
"John likes to watch movies. Mary likes movies too."

$\longrightarrow$

(John, 1), (likes, 1), (to, 1),
(watch, 1), (movies, 1), (Mary, 1),
(likes, 1), (movies, 1), (too, 1)

Document 2
"John also likes to watch football games."

$\longrightarrow$

(John, 1), (also, 1), (likes, 1), (to, 1),
(watch, 1), (football, 1), (games, 1)

Example: Word count

Group by key

(John, 1), (likes, 1), (to, 1),
(watch, 1), (movies, 1), (Mary, 1),
(likes, 1), (movies, 1), (too, 1)

(John, 1), (also, 1), (likes, 1), (to, 1),
(watch, 1), (football, 1), (games, 1)

(John, [1, 1]),
(likes, [1, 1, 1]),
(to, [1, 1]),
(watch, [1, 1]),
(movies, [1, 1]),
(Mary, [1]),
(too, [1]),
(also, [1]),
(football, [1]),
(games, [1])

Example: Word count

Distribute to reducers

(John, [1, 1]),
(likes, [1, 1, 1]),
(to, [1, 1]),
(watch, [1, 1]),
(movies, [1, 1]),
(Mary, [1]),
(too, [1]),
(also, [1]),
(football, [1]),
(games, [1])

Hash by first letter in key

Reducer 1: *A-I*
(also, [1]),
(football, [1]),
(games, [1])

Reducer 2: *J-S*
(John, [1, 1]),
(likes, [1, 1, 1]),
(movies, [1, 1]),
(Mary, [1]),

Reducer 3: *T-Z*
(to, [1, 1]),
(watch, [1, 1]),
(too, [1]),

Example: Word count

Reduce task

Reducer 1: *A-I* (also, [1]), (football, [1]), (games, [1])	$\xrightarrow{\textit{sum}}$	**Reducer 1:** *A-I* (also, 1), (football, 1), (games, 1)
Reducer 2: *J-S* (John, [1, 1]), (likes, [1, 1, 1]), (movies, [1, 1]), (Mary, [1]),	$\longrightarrow$	**Reducer 2:** *J-S* (John, 2), (likes, 3), (movies, 2), (Mary, 1),
Reducer 3: *T-Z* (to, [1, 1]), (watch, [1, 1]), (too, [1]),	$\longrightarrow$	**Reducer 3:** *T-Z* (to, 2), (watch, 2), (too, 1),

Example: Word count

Combine results

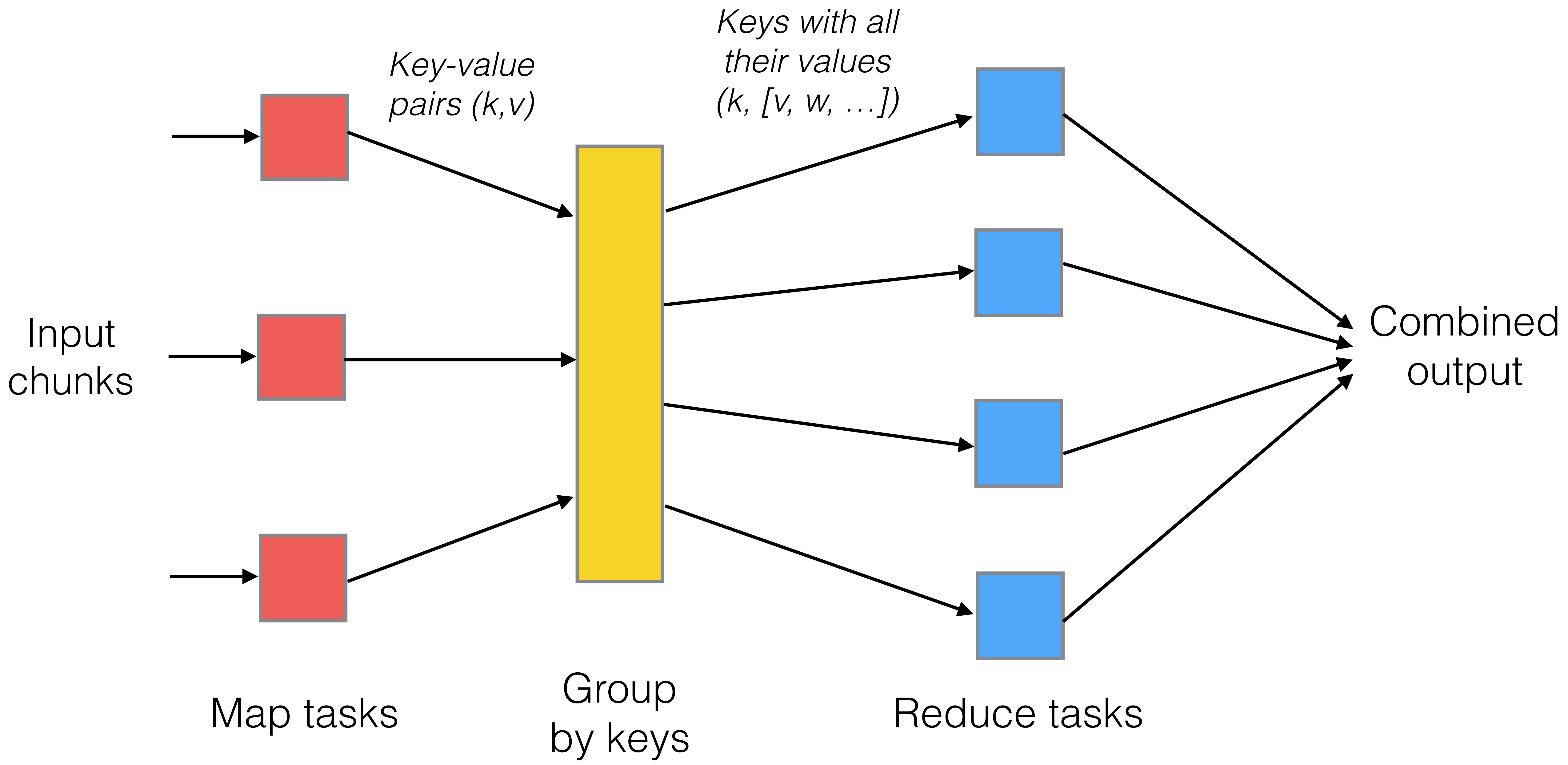
Reducer 1: *A-I*
(also, 1),
(football, 1),
(games, 1)

Reducer 2: *J-S*
(John, 2),
(likes, 3),
(movies, 2),
(Mary, 1),

Reducer 3: *T-Z*
(to, 2),
(watch, 2),
(too, 1),

(also, 1),
(football, 1),
(games, 1),
(John, 2),
(likes, 3),
(movies, 2),
(Mary, 1),
(to, 2),
(watch, 2),
(too, 1)

What is MapReduce?



What is MapReduce?

You only need to specify a map-function and a reduce-function

Complicated example

Input

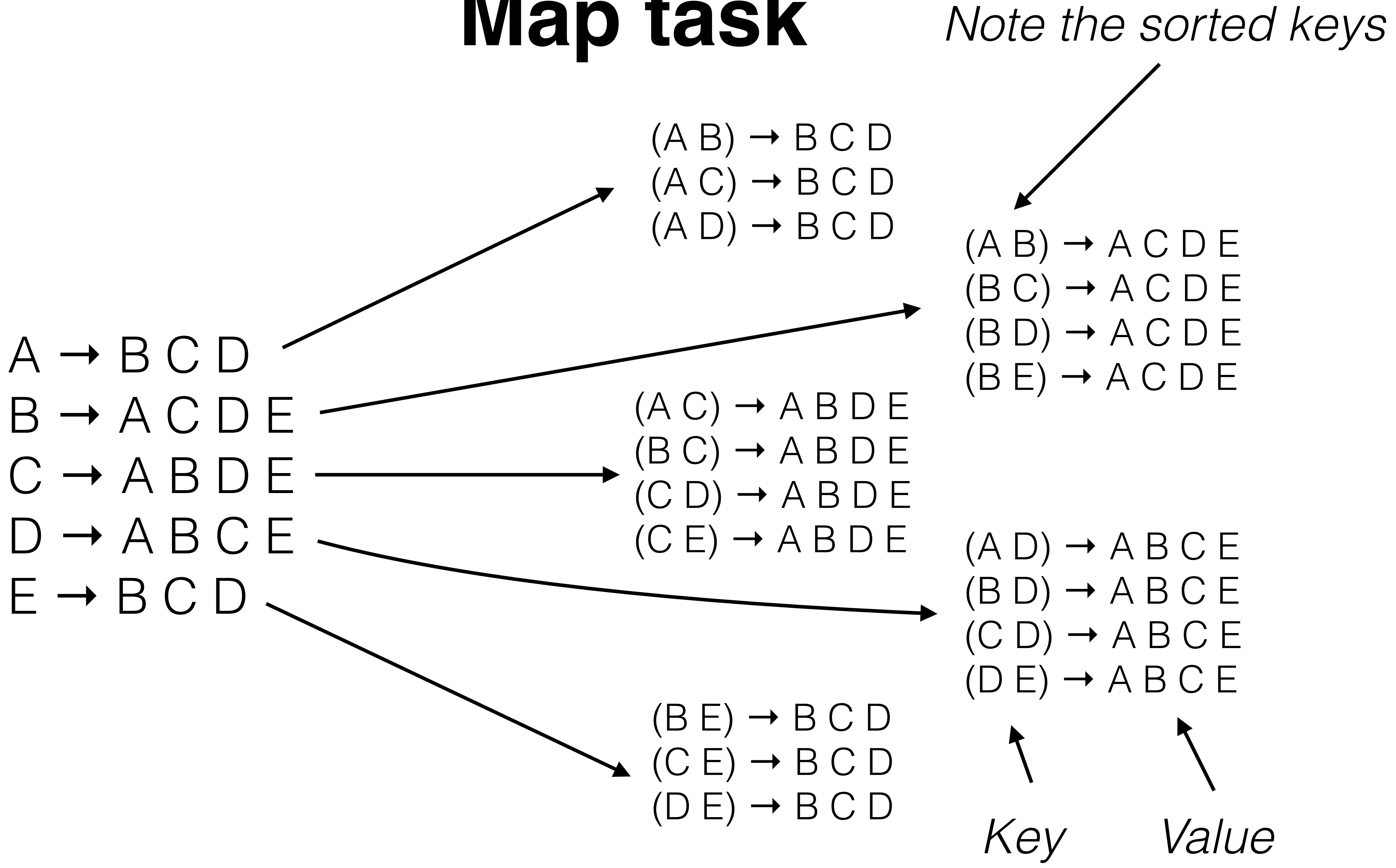
- Friend lists

Output

- For pairs of friends, a list of their common friends

Complicated example

Map task



A → B C D
B → A C D E
C → A B D E
D → A B C E
E → B C D

(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

Complicated example

Group by key

(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

(A B) → (A C D E) (B C D)
(A C) → (A B D E) (B C D)
(A D) → (A B C E) (B C D)
(B C) → (A B D E) (A C D E)
(B D) → (A B C E) (A C D E)
(B E) → (A C D E) (B C D)
(C D) → (A B C E) (A B D E)
(C E) → (A B D E) (B C D)
(D E) → (A B C E) (B C D)

Complicated example

Reduce task

(A B) → (A C D E) (B C D)		(A B) → (C D)
(A C) → (A B D E) (B C D)		(A C) → (B D)
(A D) → (A B C E) (B C D)		(A D) → (B C)
(B C) → (A B D E) (A C D E)		(B C) → (A D E)
(B D) → (A B C E) (A C D E)	$\xrightarrow{\text{Intersection}}$	(B D) → (A C E)
(B E) → (A C D E) (B C D)		(B E) → (C D)
(C D) → (A B C E) (A B D E)		(C D) → (A B E)
(C E) → (A B D E) (B C D)		(C E) → (B D)
(D E) → (A B C E) (B C D)		(D E) → (B C)

yield

Iterators

- Anything you can do “for … in …” on
- Lists, tuples, strings, files, …

Generators

- You can iterate over them, but only once!
- Like a tape
- Generates values on the fly

yield

- Like return, but returns a generator
- Code will be run each time the for uses the generator

```python
# Build and return a list
def firstn(n):
    num, nums = 0, []
    while num < n:
        nums.append(num)
        num += 1
    return nums

sum_of_first_n = sum(firstn(1000000))
```

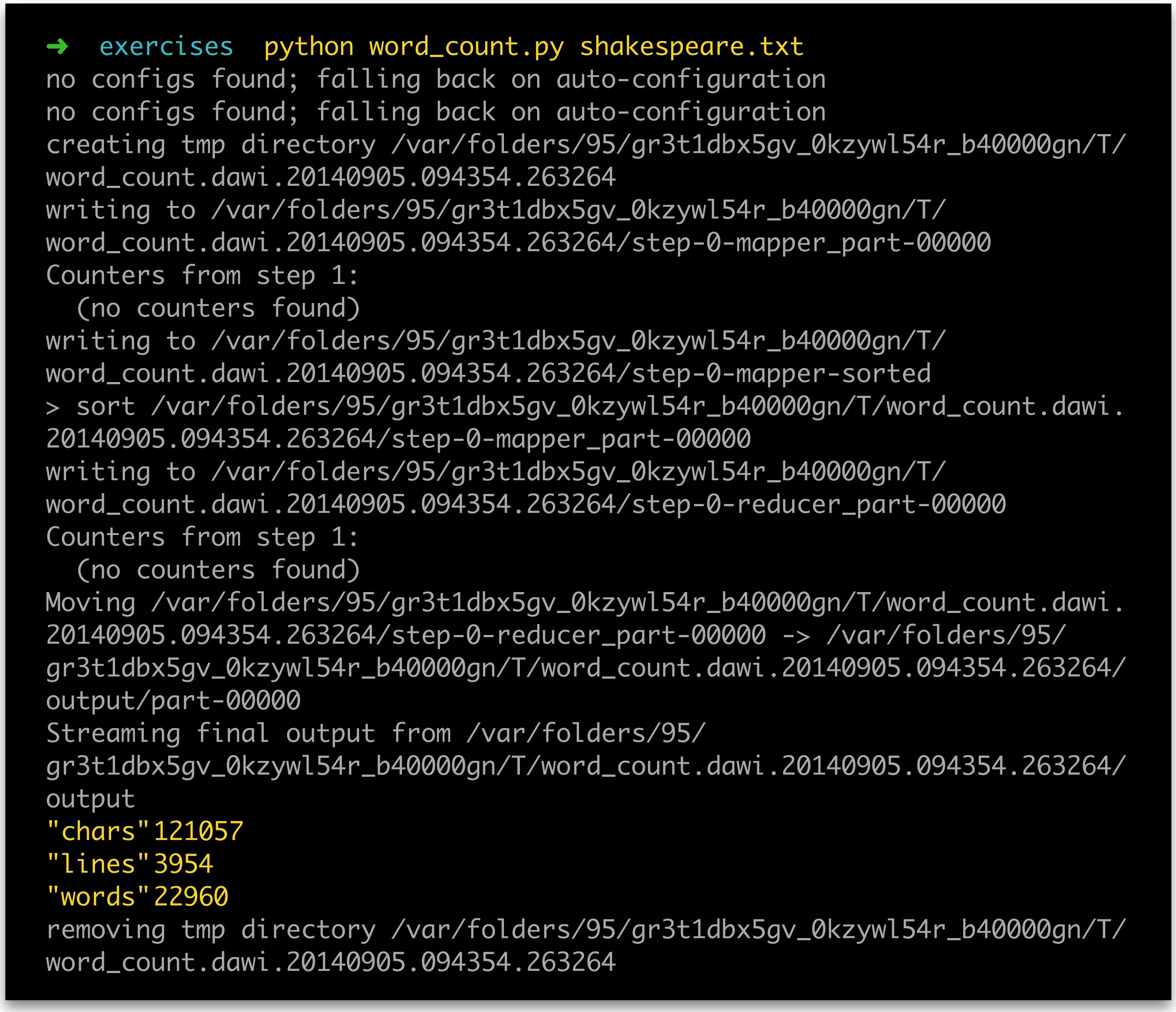
```python
# a generator that yields items instead of returning a list
def firstn(n):
    num = 0
    while num < n:
        yield num
        num += 1

sum_of_first_n = sum(firstn(1000000))
```

mrjob

word_count.py

```python
from mrjob.job import MRJob

class MRWordCount(MRJob):

    def mapper(self, key, line):
        yield "chars", len(line)
        yield "words", len(line.split())
        yield "lines", 1

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

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

mrjob



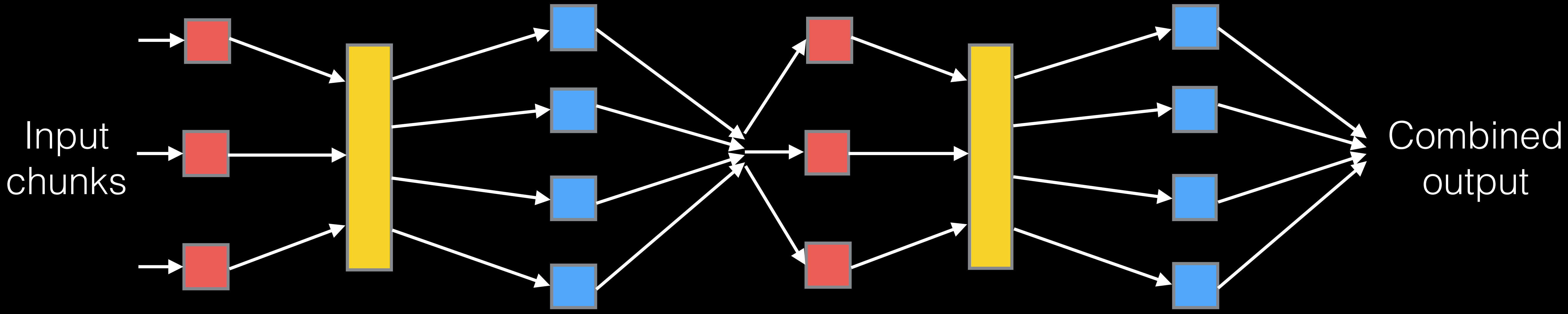
```
➜  exercises  python word_count.py shakespeare.txt
no configs found; falling back on auto-configuration
no configs found; falling back on auto-configuration
creating tmp directory /var/folders/95/gr3t1dbx5gv_0kzywl54r_b40000gn/T/
word_count.dawi.20140905.094354.263264
writing to /var/folders/95/gr3t1dbx5gv_0kzywl54r_b40000gn/T/
word_count.dawi.20140905.094354.263264/step-0-mapper_part-00000
Counters from step 1:
  (no counters found)
writing to /var/folders/95/gr3t1dbx5gv_0kzywl54r_b40000gn/T/
word_count.dawi.20140905.094354.263264/step-0-mapper-sorted
> sort /var/folders/95/gr3t1dbx5gv_0kzywl54r_b40000gn/T/word_count.dawi.
20140905.094354.263264/step-0-mapper_part-00000
writing to /var/folders/95/gr3t1dbx5gv_0kzywl54r_b40000gn/T/
word_count.dawi.20140905.094354.263264/step-0-reducer_part-00000
Counters from step 1:
  (no counters found)
Moving /var/folders/95/gr3t1dbx5gv_0kzywl54r_b40000gn/T/word_count.dawi.
20140905.094354.263264/step-0-reducer_part-00000 -> /var/folders/95/
gr3t1dbx5gv_0kzywl54r_b40000gn/T/word_count.dawi.20140905.094354.263264/
output/part-00000
Streaming final output from /var/folders/95/
gr3t1dbx5gv_0kzywl54r_b40000gn/T/word_count.dawi.20140905.094354.263264/
output
"chars"121057
"lines"3954
"words"22960
removing tmp directory /var/folders/95/gr3t1dbx5gv_0kzywl54r_b40000gn/T/
word_count.dawi.20140905.094354.263264
```

mrjob

Documentation at:
https://pythonhosted.org/mrjob/

What else?

You can extend MapReduce by having multiple mappers and reducers.



Worker failure

On worker failure

- Detect failure via periodic heartbeats
- Re-execute tasks on a new worker node

Master failure

- Could be handled, but is not (master failure unlikely)
- At Google they lost 1600 of 1800 machines once, but finished fine

Slow workers

Slow workers significantly lengthen completion time

- We can not reduce before all mappers are done
- For example other jobs consuming resources on machine

Solution

- Near end of phase, spawn backup copies of tasks
- Whichever one finishes first "wins"