

MapReduce:

**Simple Programming for
Big Results**

After this video you will be able to..

- Explain how MapReduce simplifies creating parallel programs
- Design a WordCount application using the MapReduce programming model

MapReduce = Programming Model for Hadoop Ecosystem



Parallel Programming = Requires Expertise

Semaphores

Threads

Monitors

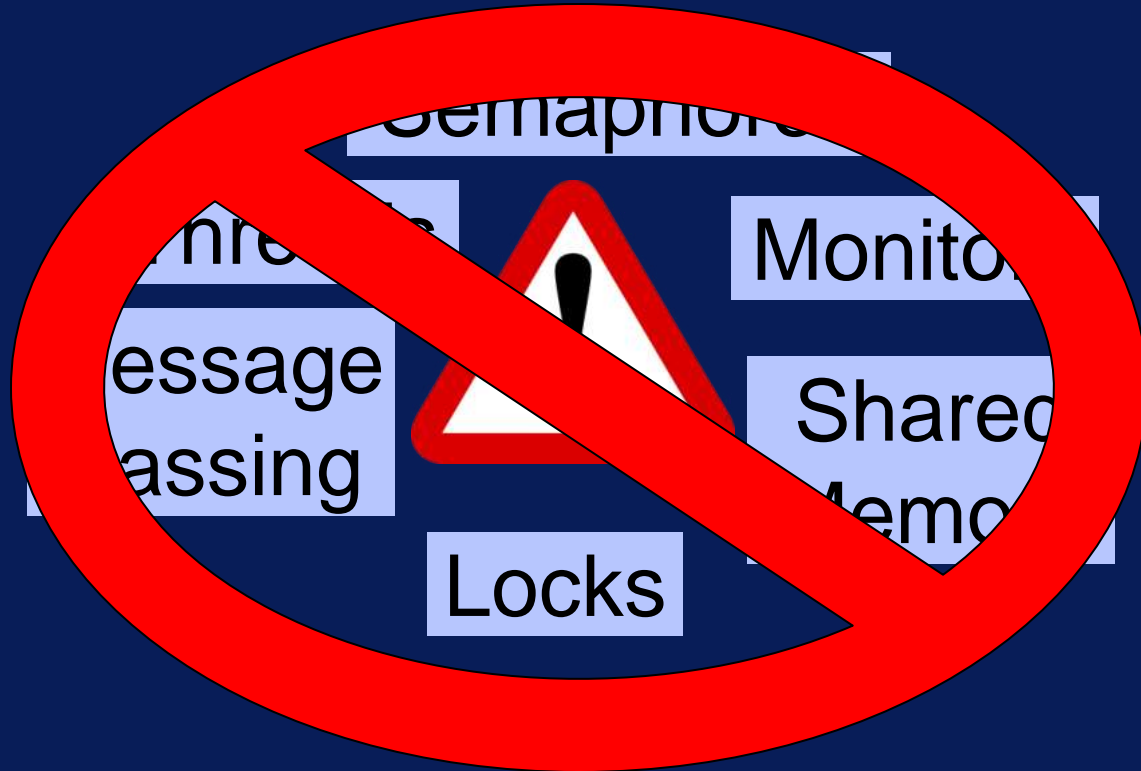
Message
Passing

Shared
Memory

Locks



MapReduce = Only Map and Reduce!



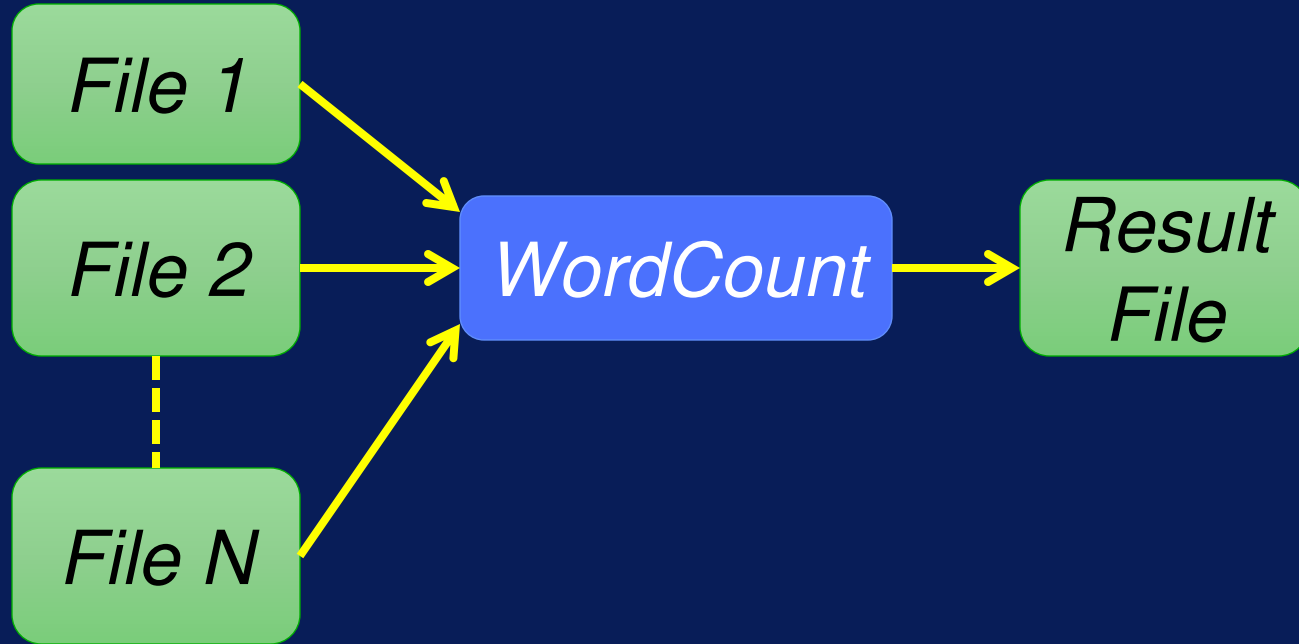
Based on Functional Programming

Map = apply operation
to all elements

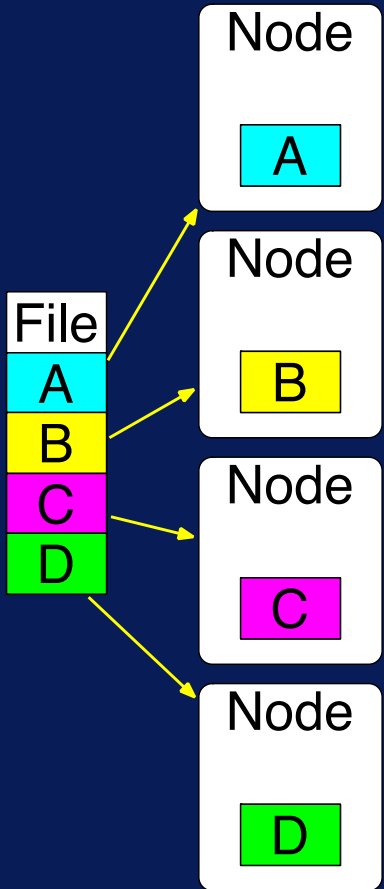
$$f(x) = y$$

Reduce = summarize
operation on elements

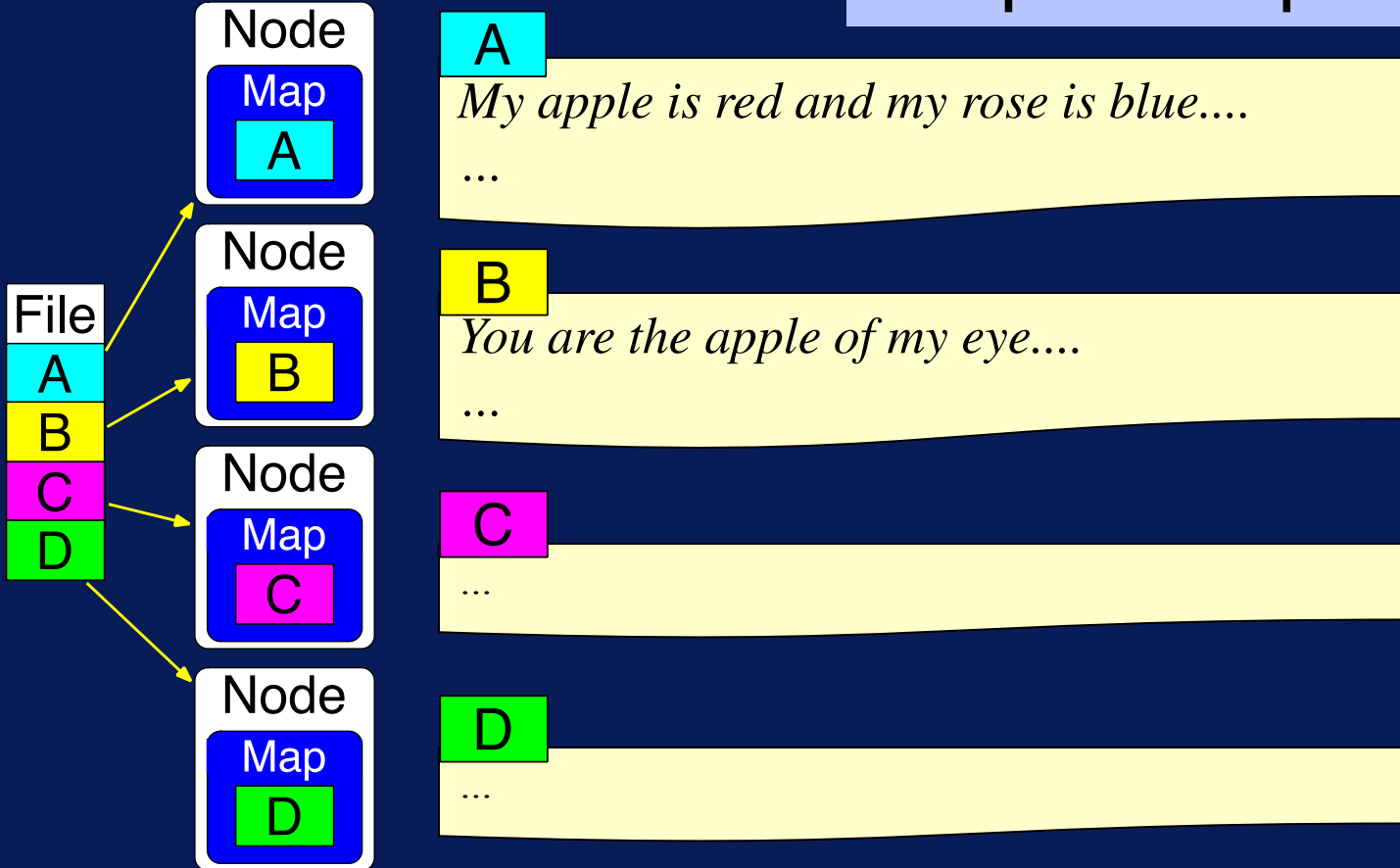
Example MapReduce Application: WordCount



Step 0: File is stored in HDFS



Step 1: Map on each node



**Map generates
key-value pairs**

A

My apple is red and my rose is blue....

...

Node

Map

A

my, my \rightarrow (my, 1), (my, 1)

apple \rightarrow (apple, 1)

is, is \rightarrow (is, 1), (is, 1)

red \rightarrow (red, 1)

and \rightarrow (and, 1)

rose \rightarrow (rose, 1)

blue \rightarrow (blue, 1)

File

A

B

C

D

**Map generates
key-value pairs**

B

You are the apple of my eye....

...

You → (You, 1)

are → (are, 1)

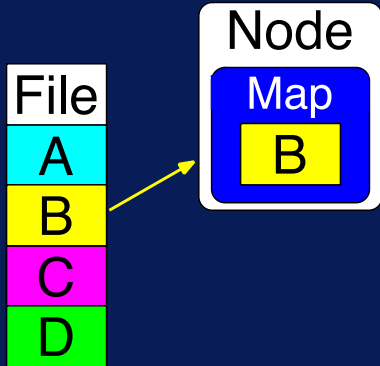
the → (the, 1)

apple → (apple, 1)

of → (of, 1)

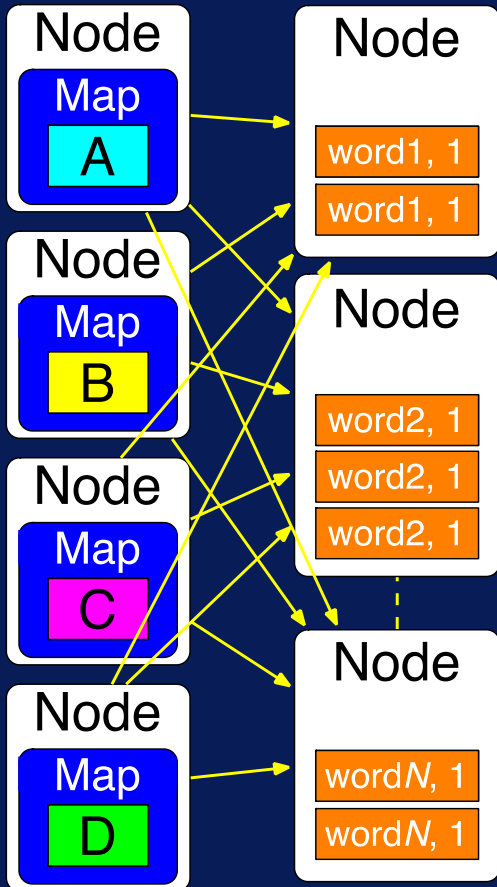
my → (my, 1)

eye → (eye, 1)



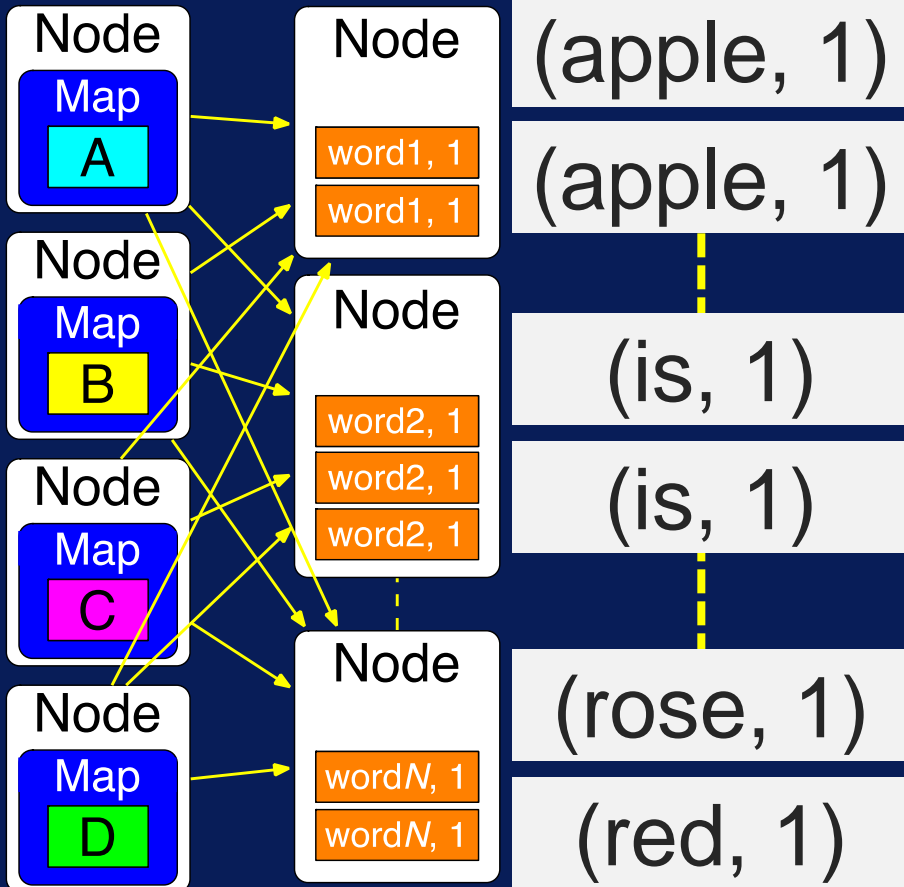
Step 2: Sort and Shuffle

**Pairs with same key
moved to same node**



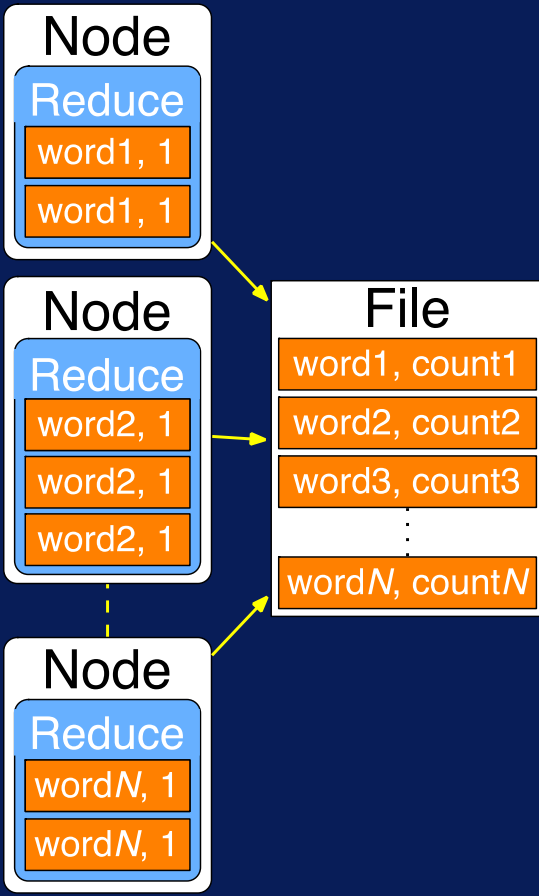
Step 2: Sort and Shuffle

**Pairs with same key
moved to same node**



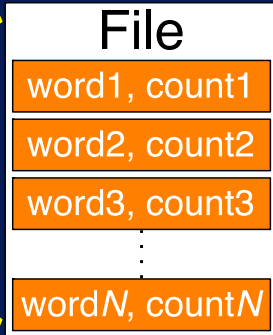
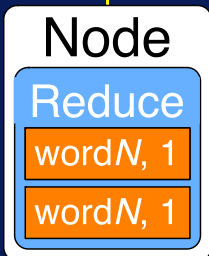
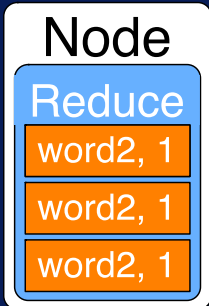
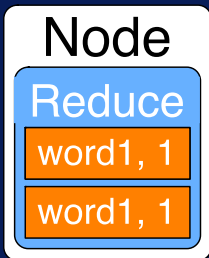
Step 3: Reduce

Add values for same keys



Step 3: Reduce

Add values for same keys



(You, 1)

(apple, 1), (apple, 1)

(my, 1), (my, 1),
(my, 1)

(red, 1)

(rose, 1)

(You, 1)

(apple, 2)

(my, 3)

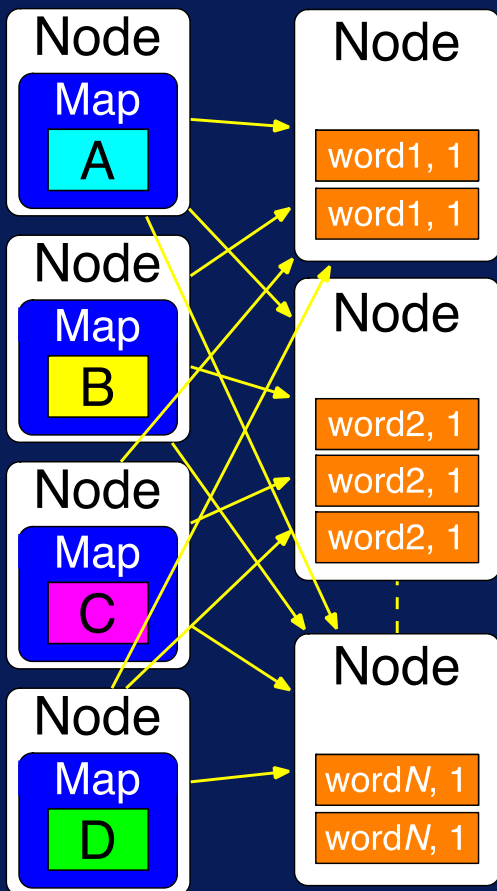
(red, 1)

(rose, 1)



Represents a large
number of applications.

Sort and Shuffle



(You, <http://you1.fake>)

(apple, <http://apple1.fake>)

(apple, <http://apple2.fake>)

(is, <http://apple2.fake>)

(is, <http://apple2.fake>)

(rose, <http://apple2.fake>)

(red, <http://apple2.fake>)

Reduce Results for “apple”

```
(apple -> http://apple1.fake,  
         http://apple2.fake)
```

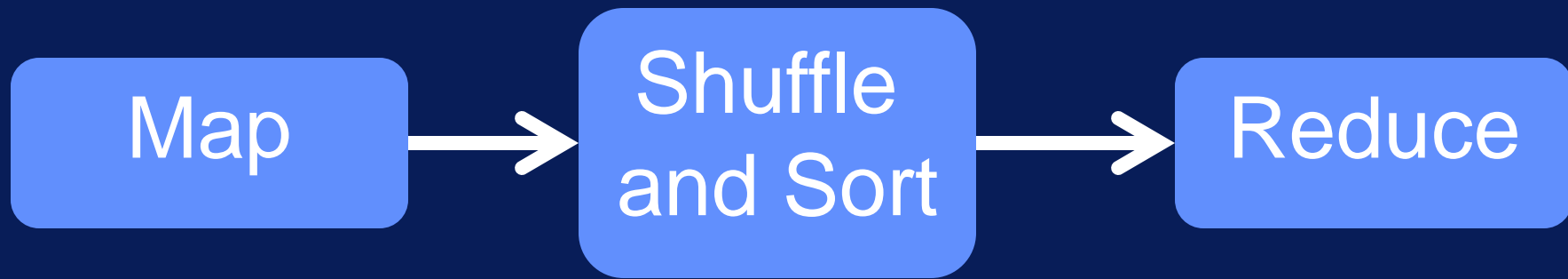
Reduce Results for “apple”

Key

Value

apple -> <http://apple1.fake>,
<http://apple2.fake>)

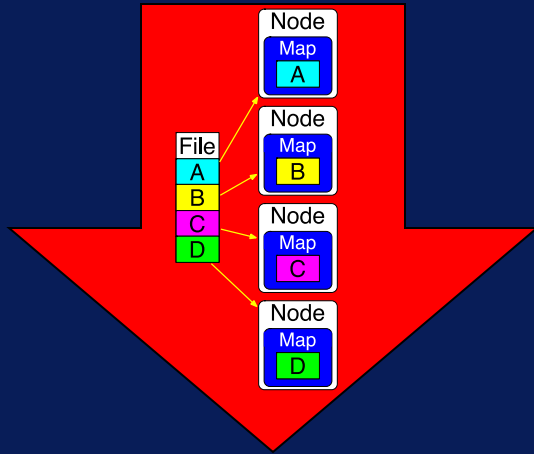




Map

Shuffle
and Sort

Reduce

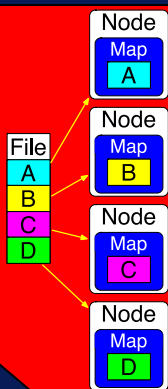


Parallelization
over the input

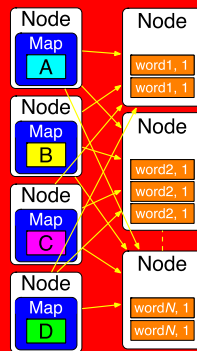
Map

Shuffle
and Sort

Reduce



Parallelization
over the input

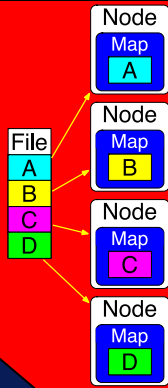


Parallelization
data sorting

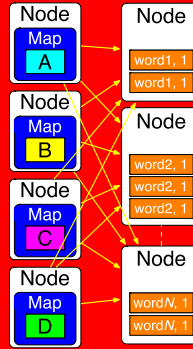
Map

Shuffle
and Sort

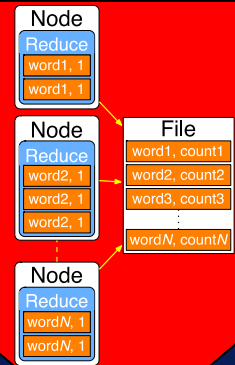
Reduce



Parallelization
over the input



Parallelization over
intermediate data



Parallelization
over data groups

MapReduce is bad for:

MapReduce is bad for:

Frequently **changing** data

MapReduce is bad for:

Frequently **changing** data

Dependent tasks

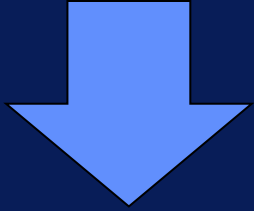
MapReduce is bad for:

Frequently **changing** data

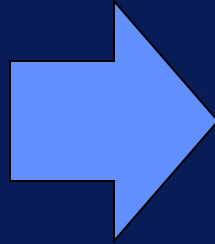
Dependent tasks

Interactive analysis

MapReduce



Simplified parallel
programming



Applications with
independent data-
parallel tasks