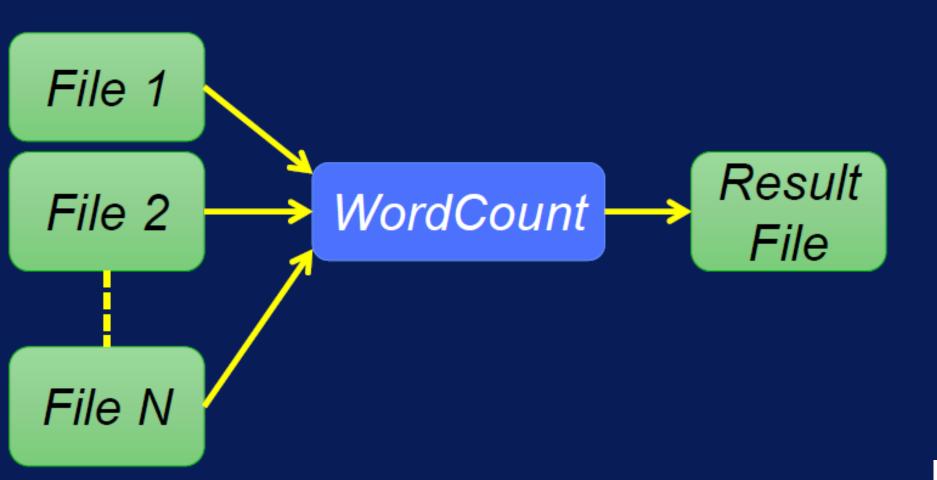
Based on Functional Programming

Map = apply operation to all elements

f(x) = y

Reduce = summarize operation on elements

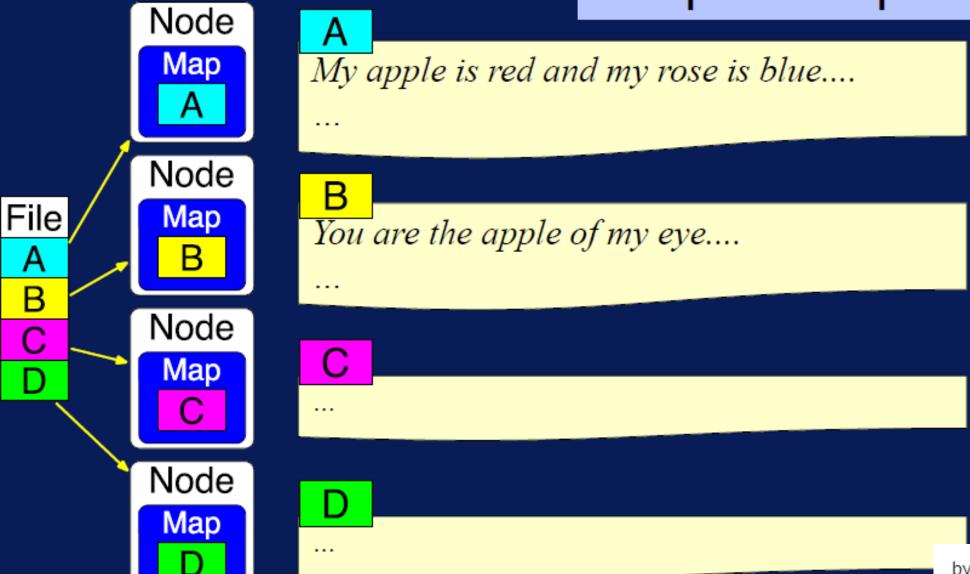
Example MapReduce Application: WordCount



Node Node File В Node Node

Step 0: File is stored in HDFS

Step 1: Map on each node



Node Map A

Α

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

...

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)

Map generates key-value pairs

File A B C

Map generates key-value pairs

B
You are the apple of my eye....

...

You \rightarrow (You, 1)

are \rightarrow (are, 1)

the \rightarrow (the, 1)

apple → (apple, 1)

of \rightarrow (of, 1)

 $my \rightarrow (my, 1)$

eye \rightarrow (eye, 1)

File A B

Node

Map

Node Node Map word1, 1 word1, 1 Node Node Map word2, 1 Node word2, 1 word2, 1 Map Node Node Map wordN, 1 wordN, 1

Step 2: Sort and Shuffle

Pairs with same key moved to same node

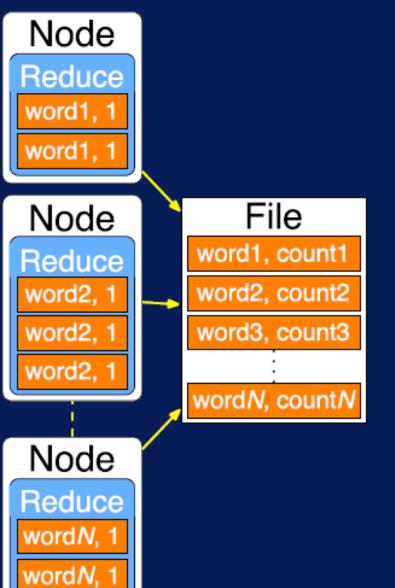
(You, 1) Node Node (apple, 1) Map word1, 1 (apple, 1) word1, 1 Node Node Map (is, 1) word2, 1 word2, 1 Node (is, 1) word2, 1 Map Node (rose, 1) Node Map wordN, 1 (red, 1) wordN, 1

Step 2: Sort and Shuffle

Pairs with same key moved to same node

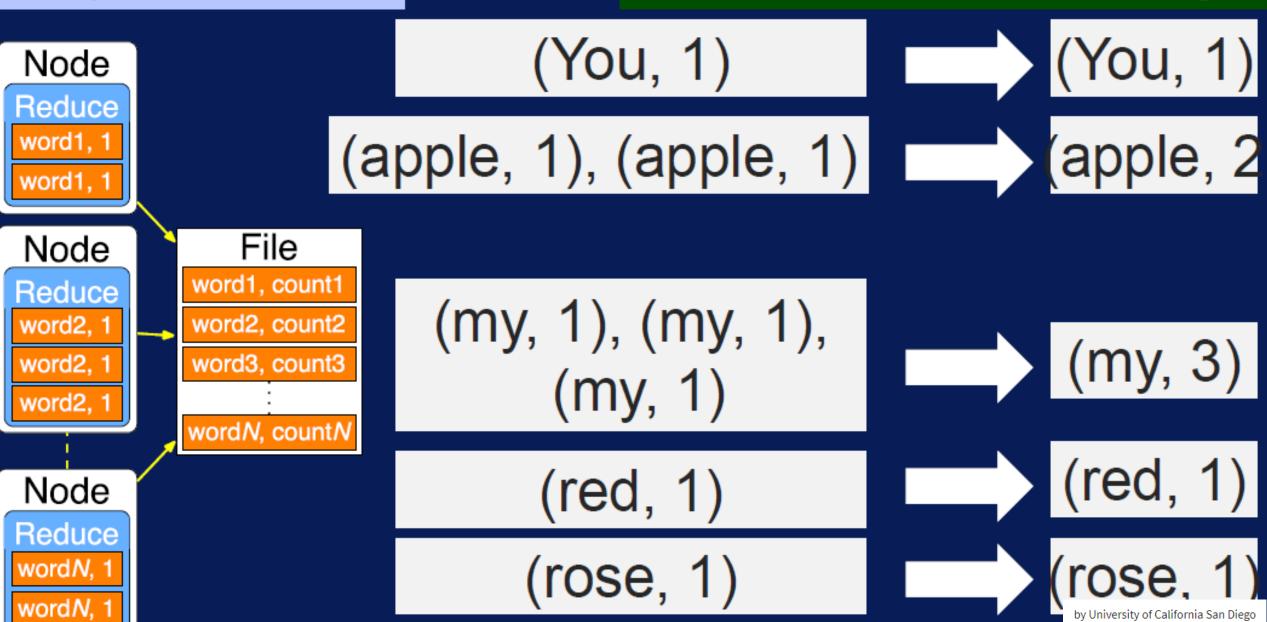
Step 3: Reduce

Add values for same keys

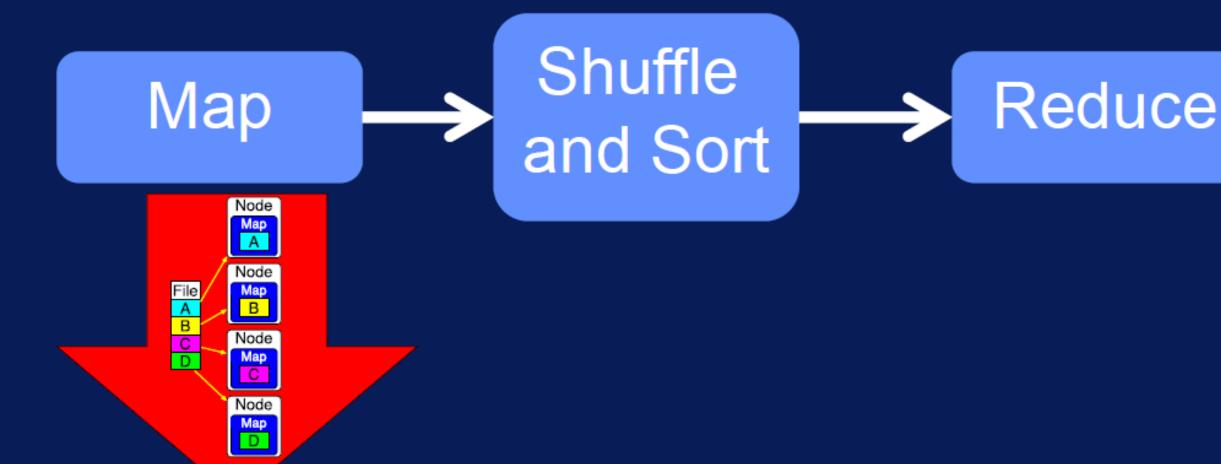


Step 3: Reduce

Add values for same keys



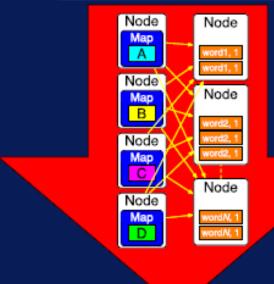
→ Shuffle and Sort → Reduce



Parallelization over the input



Node
Map
A
Node
Map
B
Node
Map
C
D
Node
Map
C
Node
Map
D
D

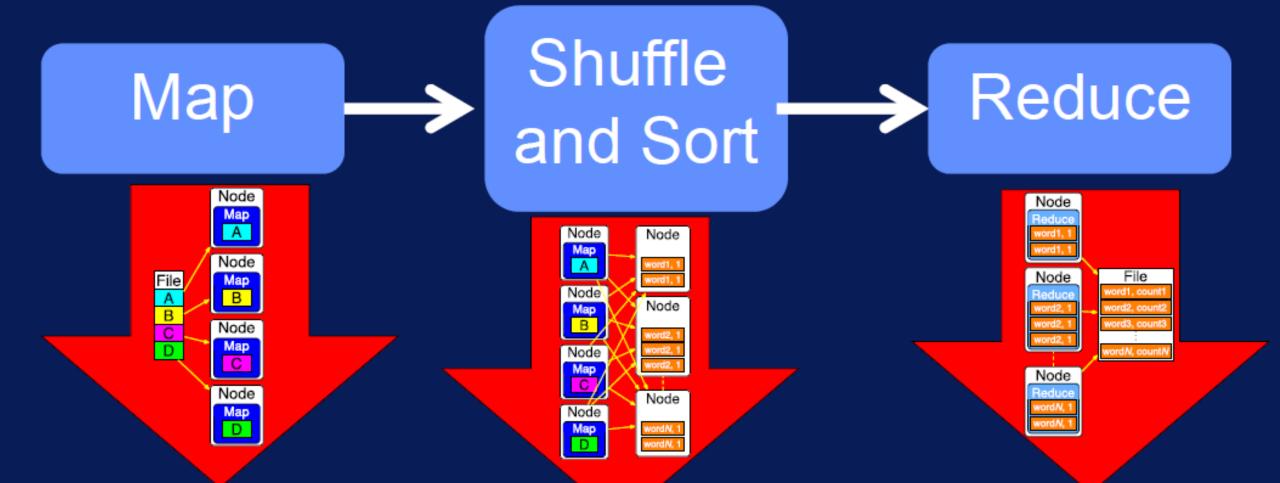


Parallelization over the input

Parallelization data sorting

by University of California San Diego

Reduce



Parallelization over the input

Parallelization over intermediate data

Parallelization over data groups