Based on Functional Programming

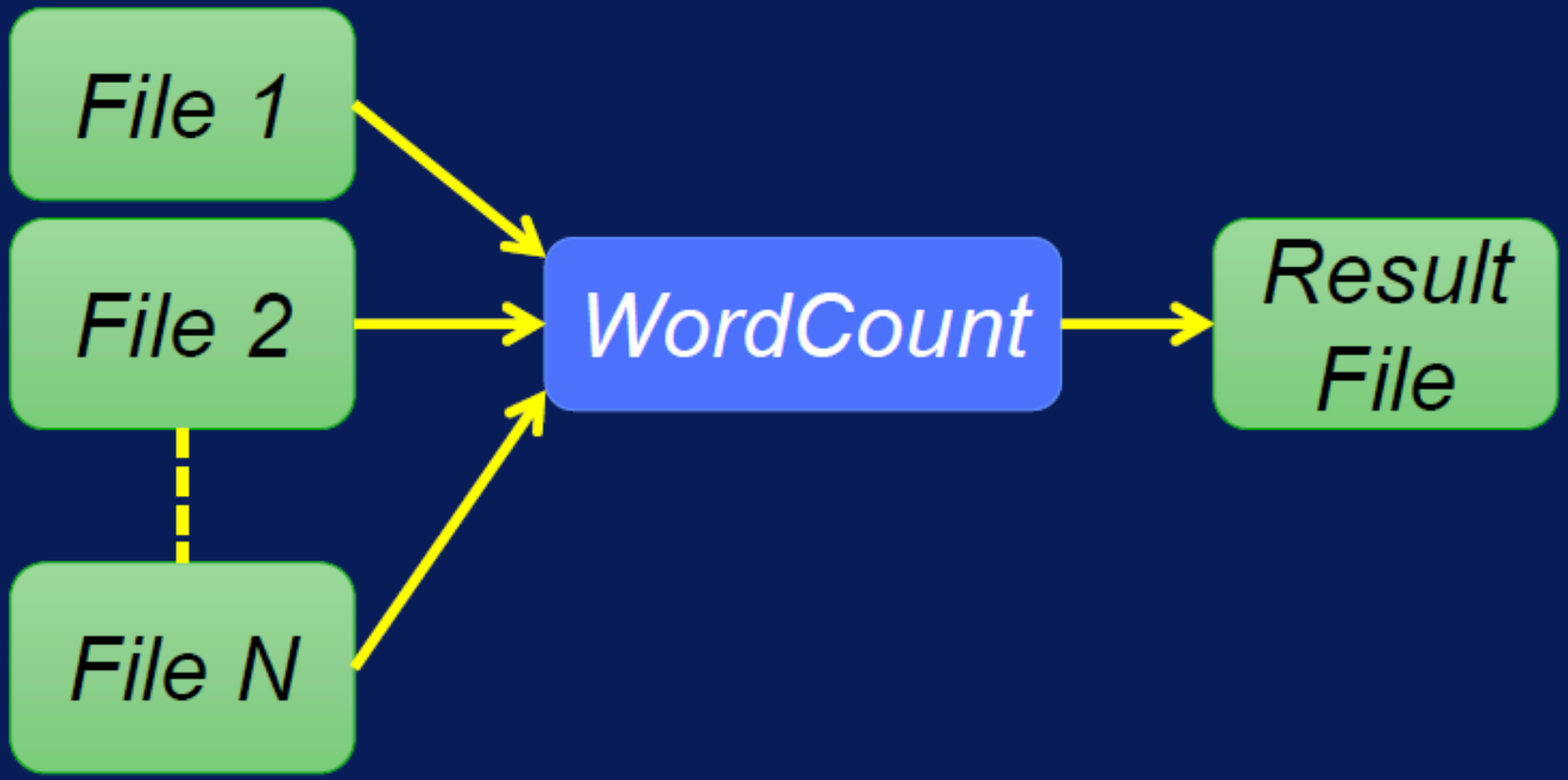
Map = apply operation to all elements

Reduce = summarize operation on elements

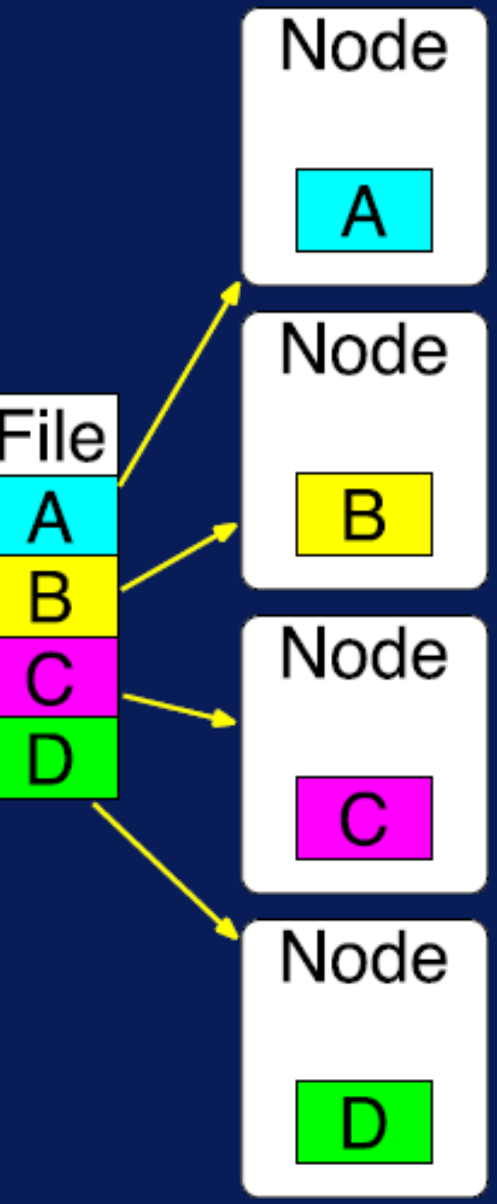
f (x) = y

by University of California San Diego

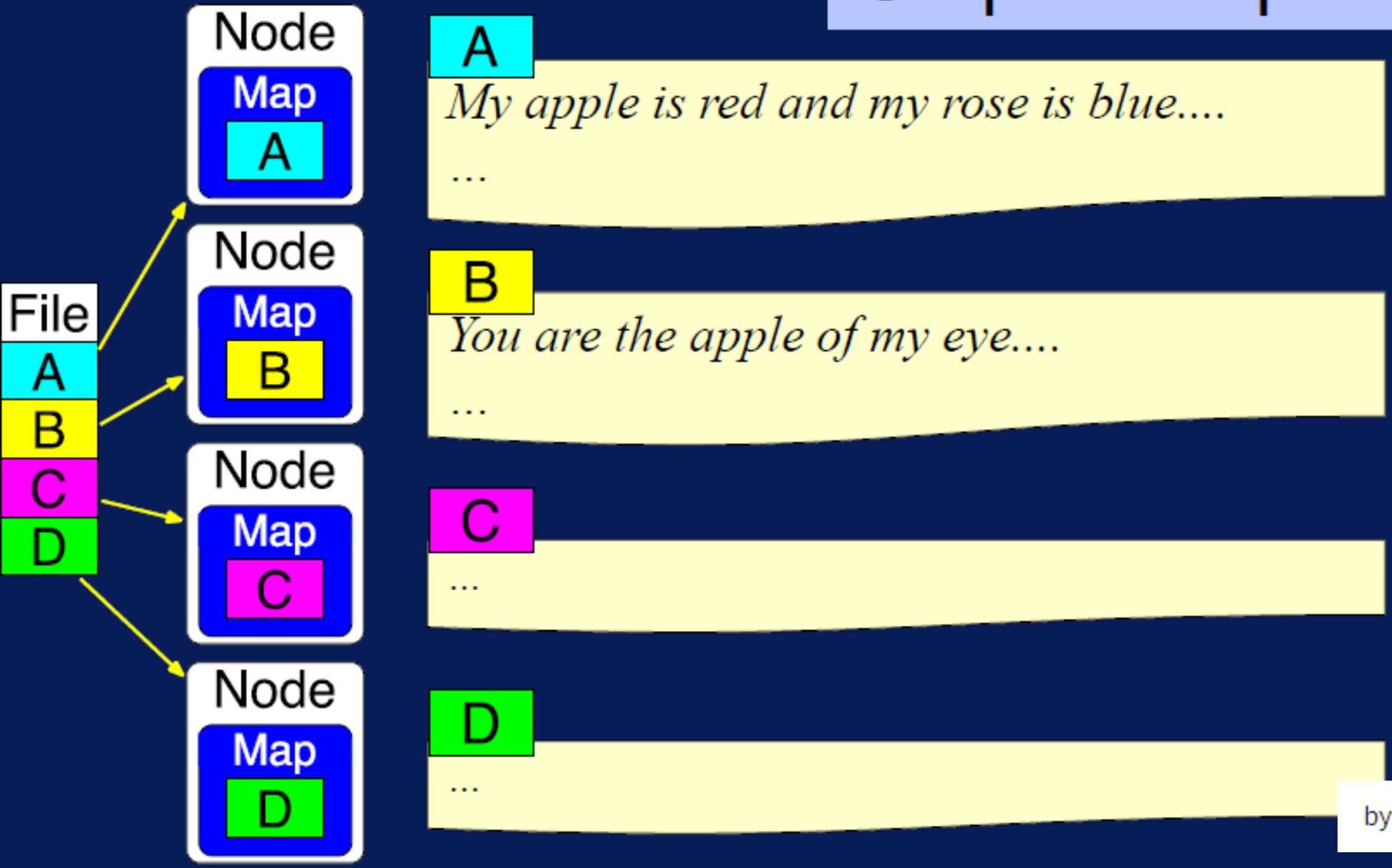
Example MapReduce Application: WordCount



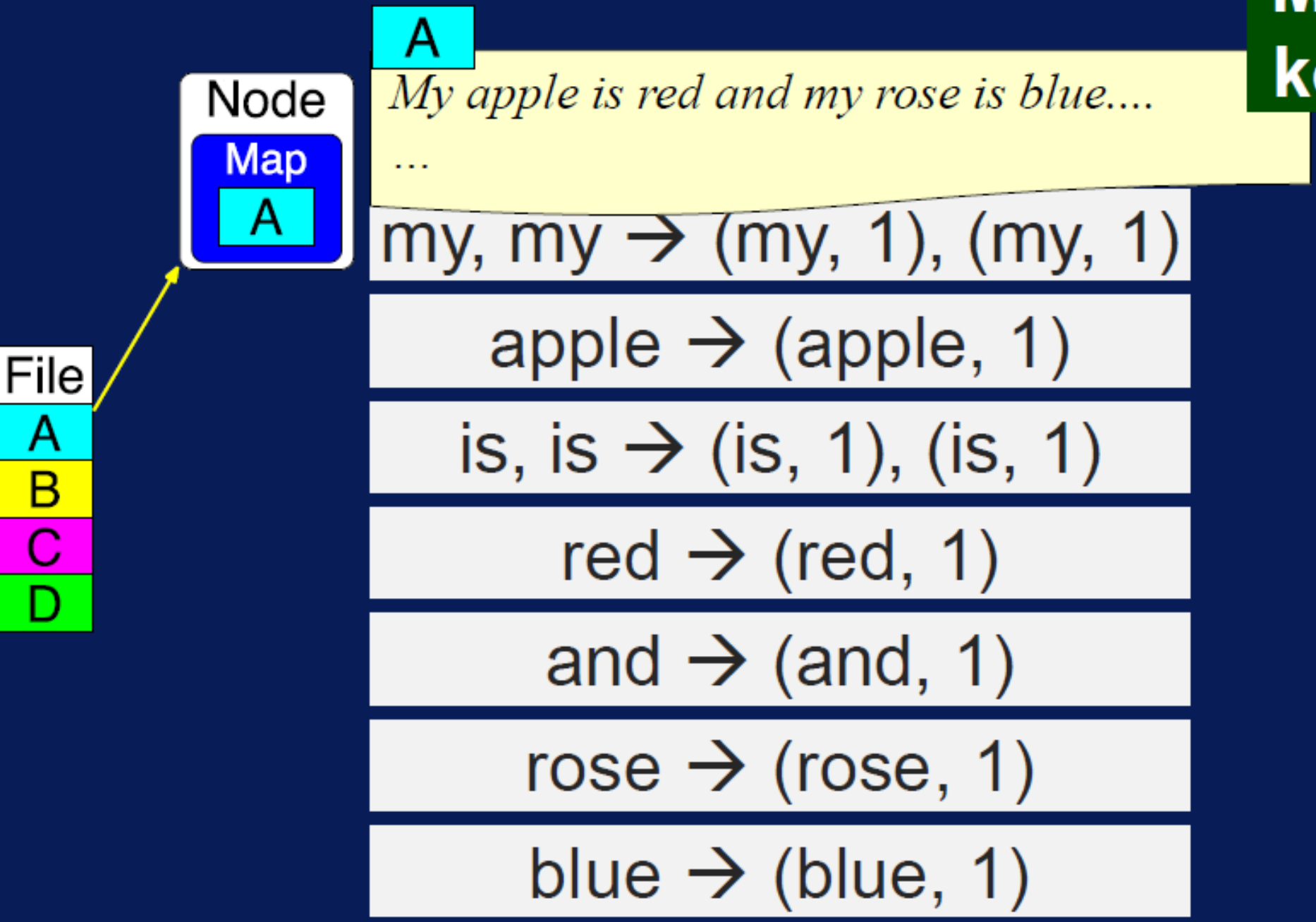
Step 0: File is stored in HDFS



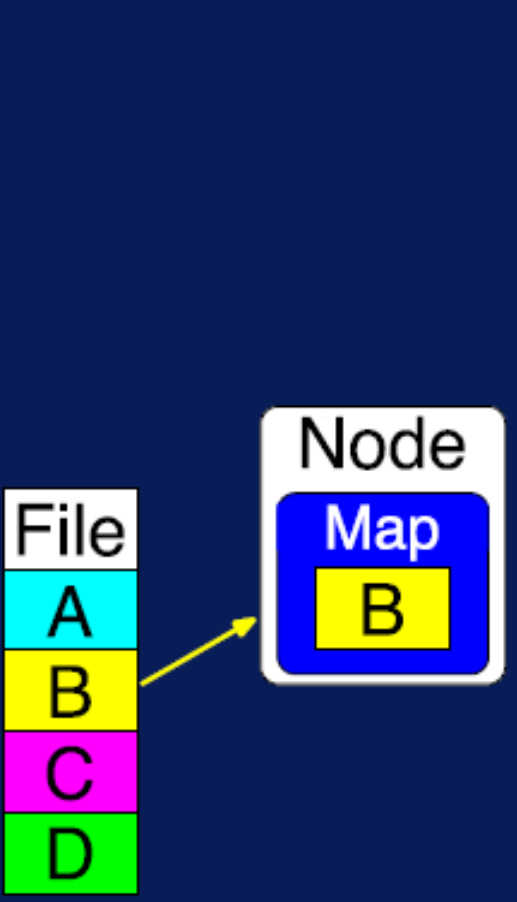
Step 1: Map on each node



Map generates key-value pairs



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)

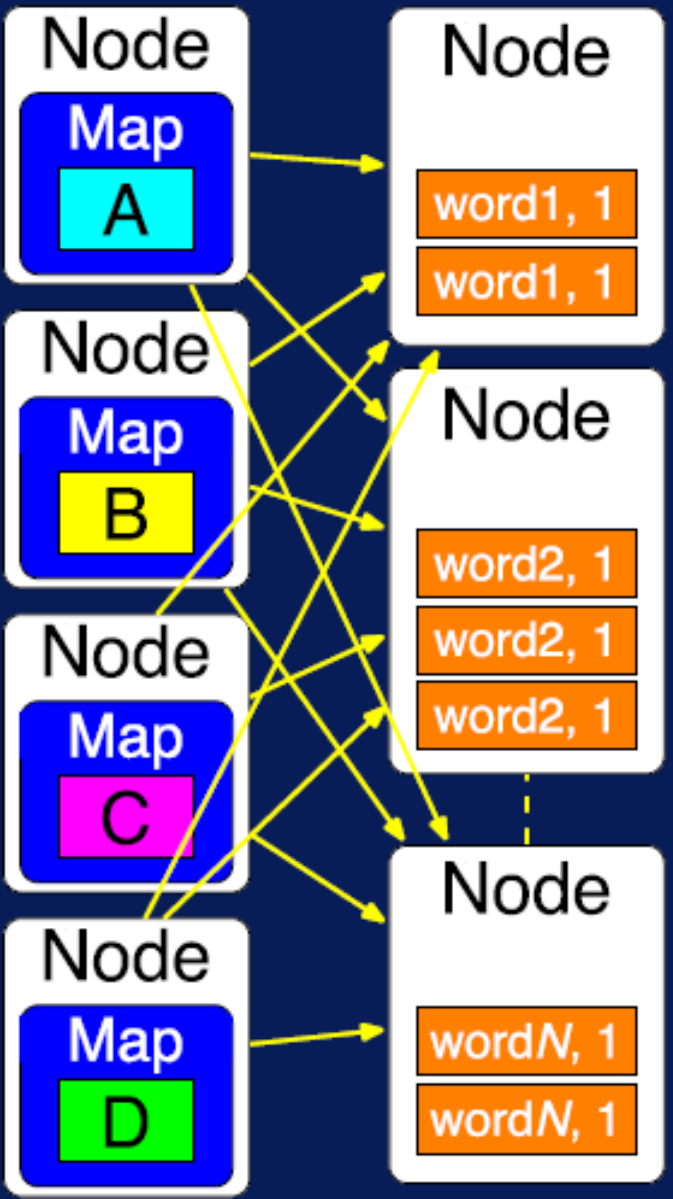
of → (of, 1)

my → (my, 1)

eye → (eye, 1)

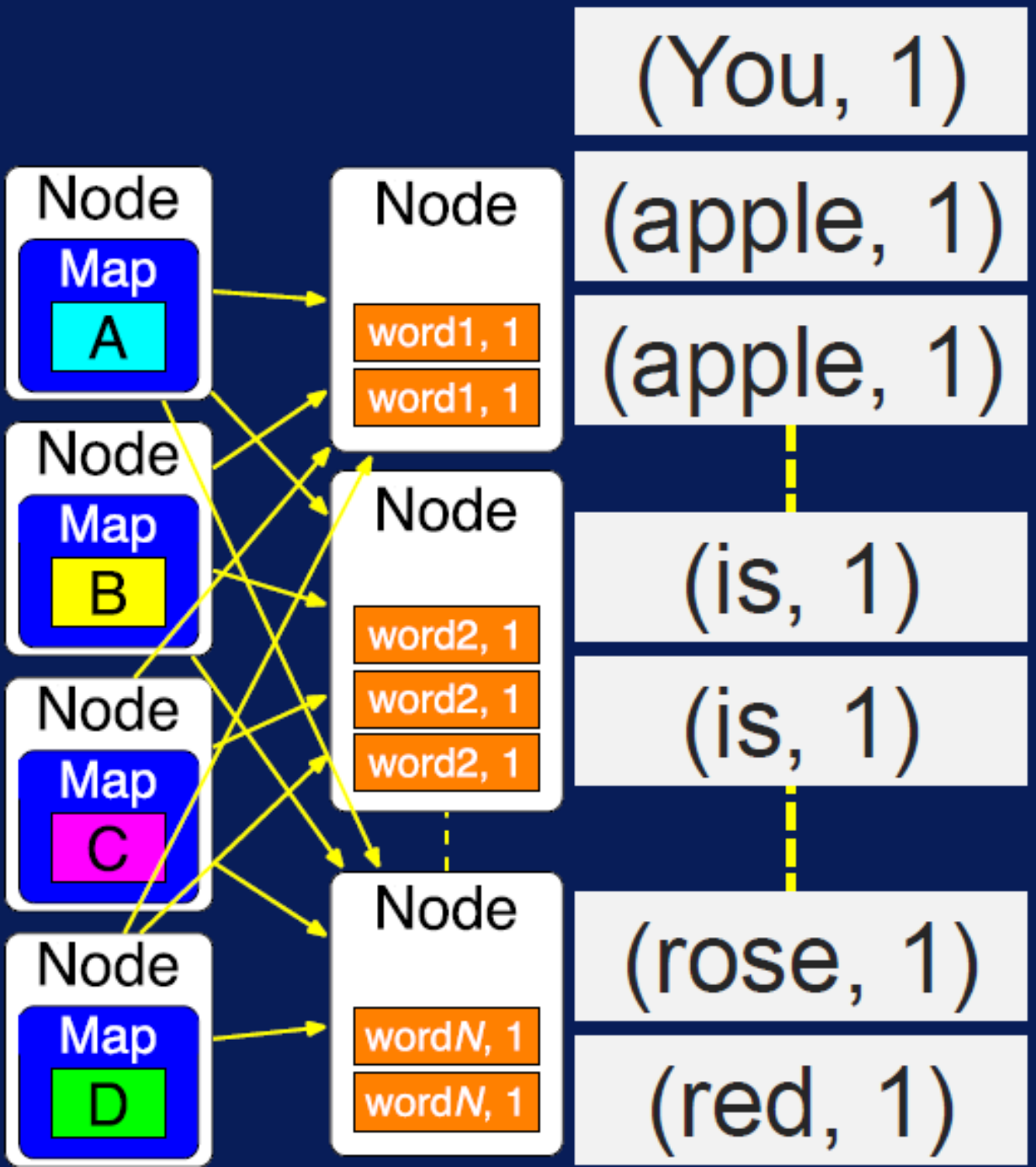
Step 2: Sort and Shuffle

Pairs with same key moved to same node



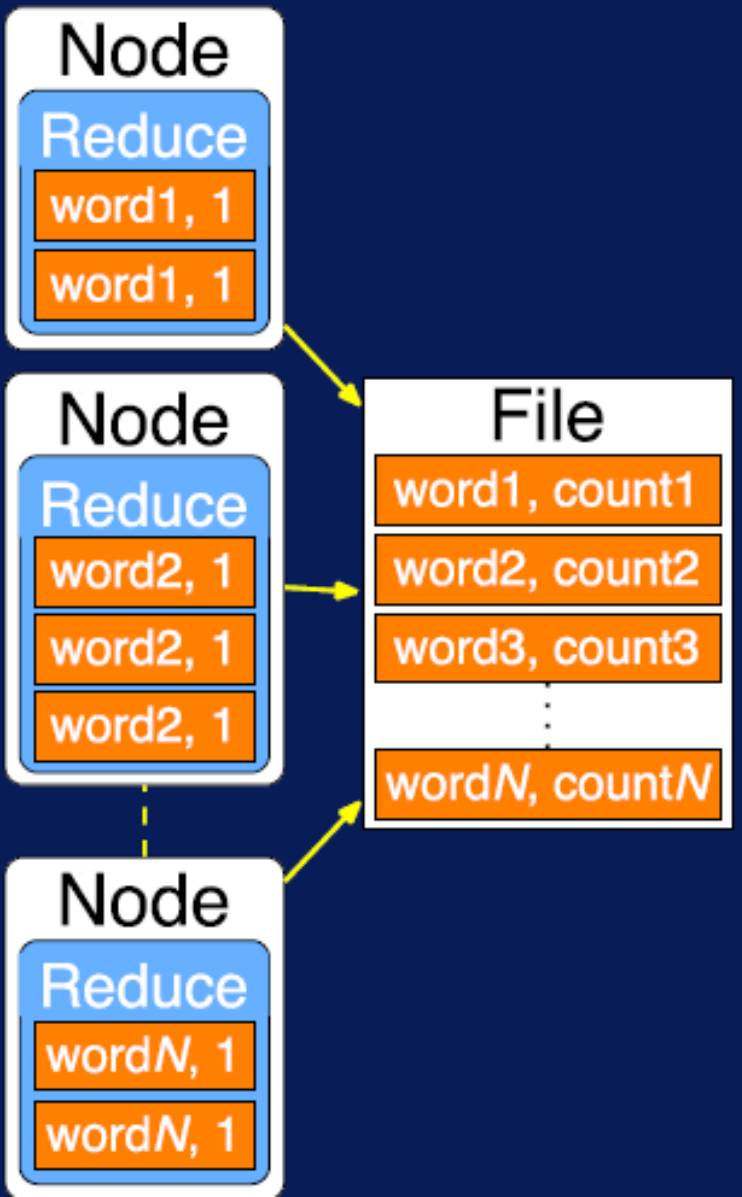
Step 2: Sort and Shuffle

Pairs with same key moved to same node



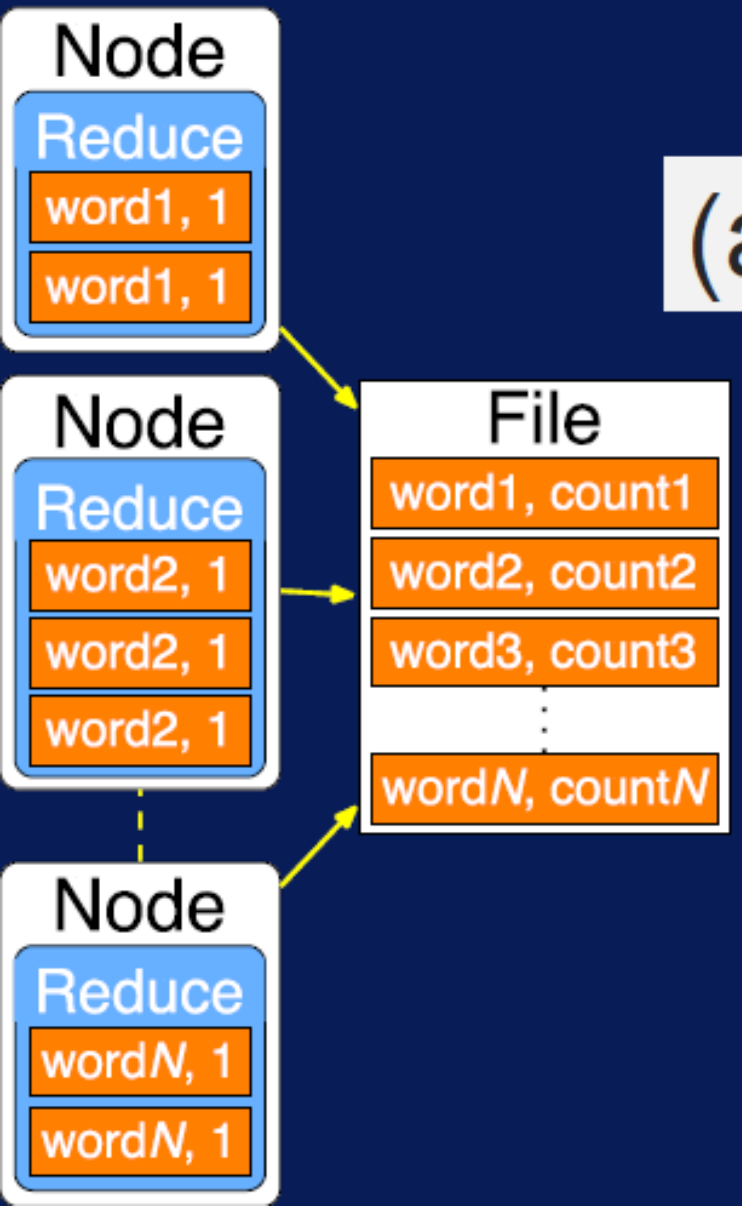
Step 3: Reduce

Add values for same keys



Step 3: Reduce

Add values for same keys



(You, 1) → (You, 1)

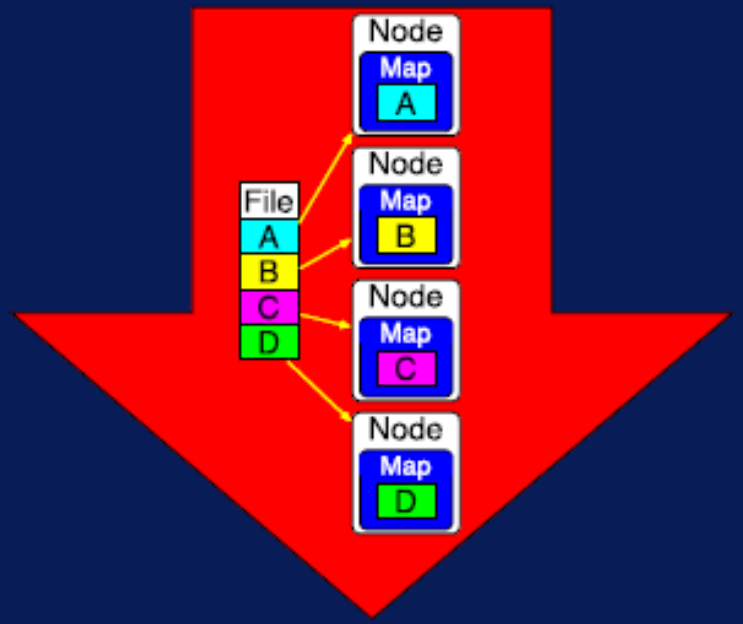
(apple, 1), (apple, 1) → (apple, 2)

(my, 1), (my, 1), (my, 1) → (my, 3)

(red, 1) → (red, 1)

(rose, 1) → (rose, 1)

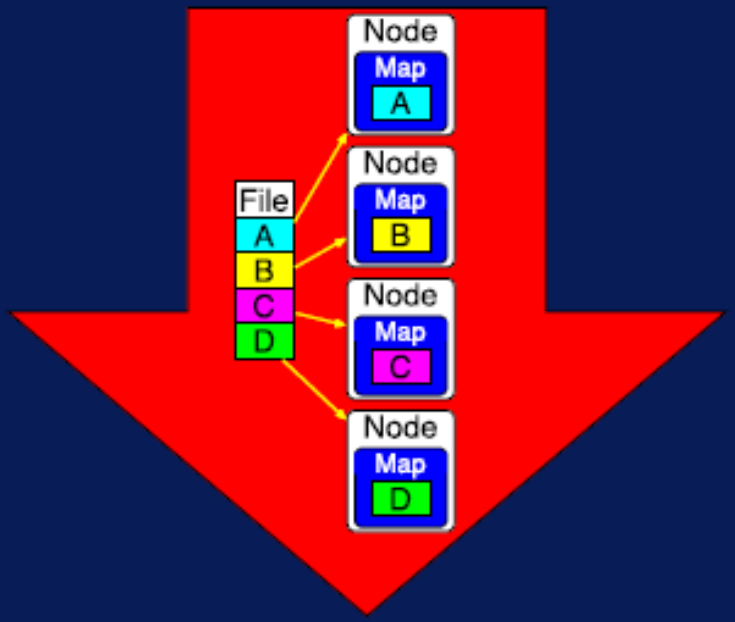
Map → Shuffle and Sort → Reduce



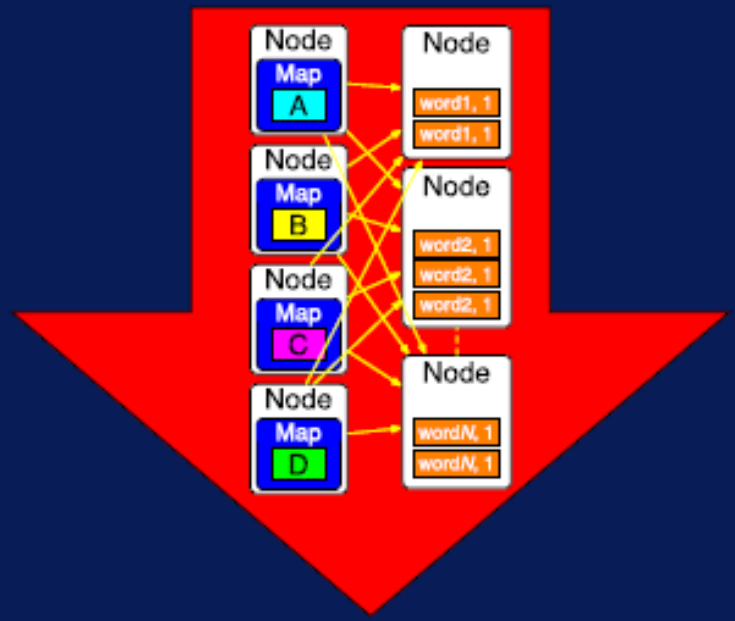
Parallelization over the input

by University of California San Diego

Map → Shuffle and Sort → Reduce



Parallelization over the input



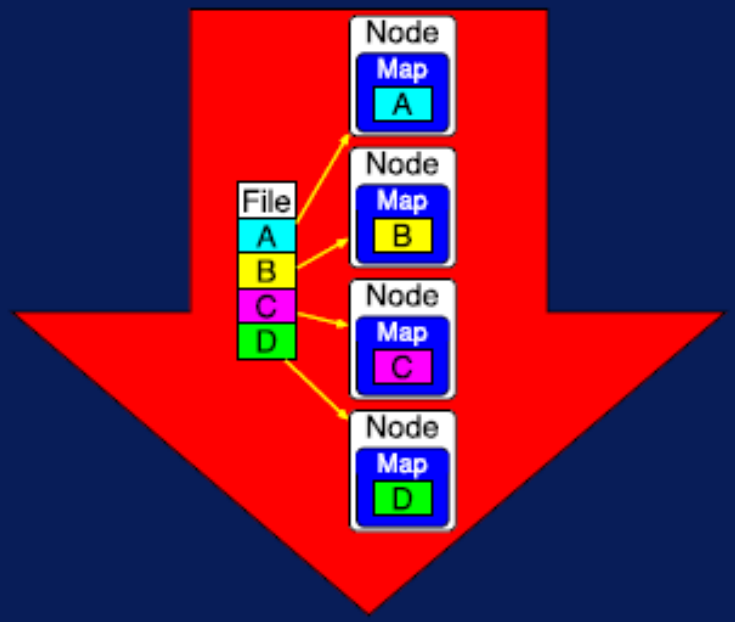
Parallelization data sorting

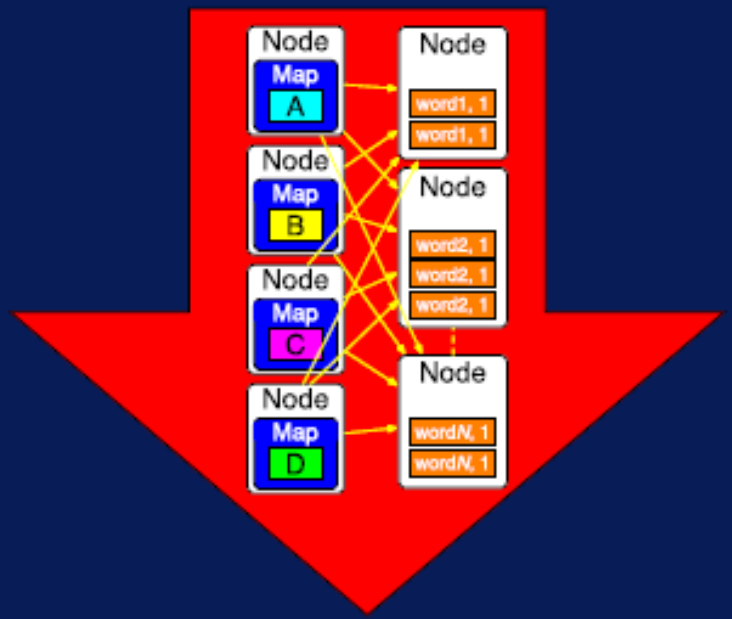
by University of California San Diego

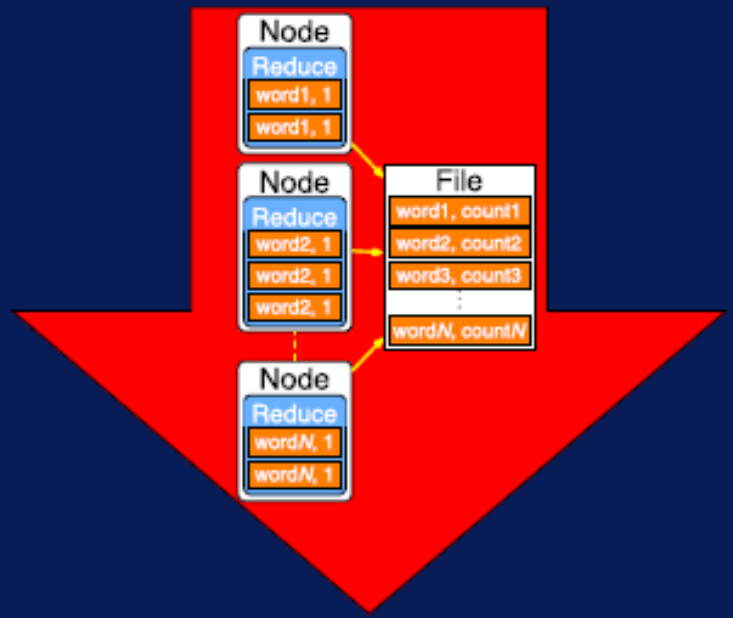
Map

Shuffle and Sort

Reduce







Parallelization over the input

Parallelization over intermediate data

Parallelization over data groups

by University of California San Diego