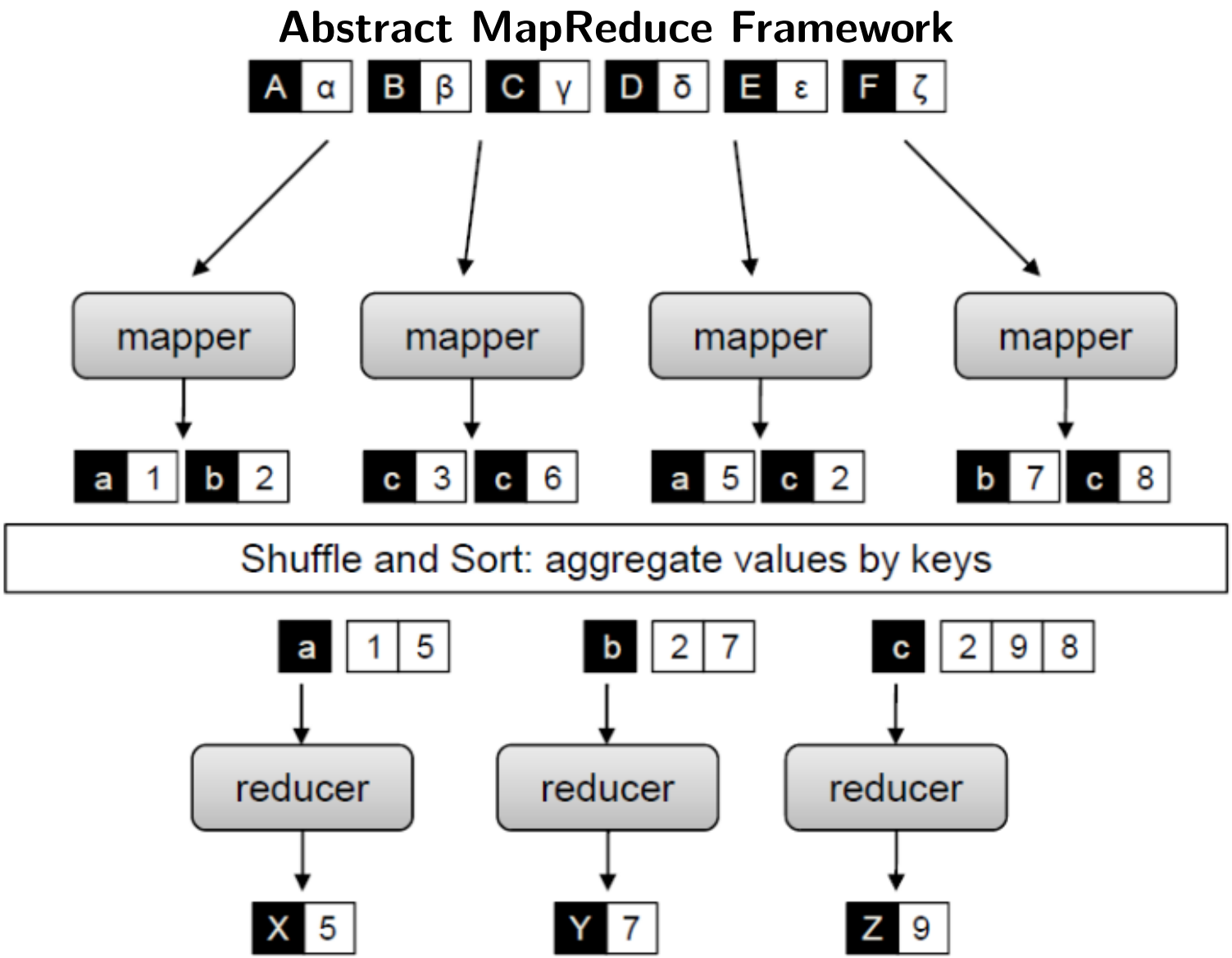
Large Scale Data Processing
CSE3025

Dr. Ramesh Ragala

School of Computer Science and Engineering
VIT Chennai

April 9, 2021


Abstract MapReduce Framework
A α
B β
C γ
D δ
E ε
F ζ
mapper
mapper
mapper
mapper
a 1
b 2
c 3
c 6
a 5
c 2
b 7
c 8
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8
reducer
reducer
reducer
X 5
Y 7
Z 9

Recall

- As the Map operation is parallelized the input dataset is first split to several pieces called **FileSplits**.
- Then a new map task is created per FileSplit.
- When an individual map task starts it will open a new output writer per configured reduce task.
- It will then proceed to read its FileSplit using the RecordReader it gets from the specified InputFormat.
- InputFormat parses the input and generates key-value pairs.
- As key-value pairs are read from the RecordReader they are passed to the configured Mapper.
- The user supplied Mapper does whatever it wants with the input pair and calls OutputCollector.collect with key-value pairs of its own choosing. The Map output is written into a SequenceFile.

Paritioner in MapReduce

- A partitioner works like a condition in processing an input dataset.
- The partition phase takes place after the Map phase and before the Reduce phase.
- The number of partitioners is equal to the number of reducers i.e partitioner will divides the data according to the number of reducers.
- A partitioner partitions the key-value pairs of intermediate Map-outputs. $\rightarrow$ It controls the partitioning of the key of the intermediate mapper output.
- It partitions the data using a **user-defined condition**, which works like a hash function.
- By hash function, key (or a subset of the key) is used to derive the partition.
- According to the key-value each mapper output is partitioned and records having the same key value go into the same partition (within each mapper), and then each partition is sent to a reducer.
- Partition class determines which partition a given (key, value) pair will go.
- Partition phase takes place after map phase and before reduce phase.
- The total number of partitions is same as the number of Reducer tasks for the job.

Paritioner in MapReduce

- Partitioner runs on the same machine where the mapper had completed its execution by consuming the mapper output.
- Entire mapper output sent to the partitioner
- The Hash partitioner partitions the key space by using the hash code

```
public int getPartition(K key, V value,
   int numReduceTasks) {
         return(key.hashCode()
    & Integer.MAX_VALUE) % numReduceTasks;
}
```

- Hadoop allows custom partitioners also
- The partitioner abstact class with a single method used for the partitioner in Hadoop.
- The partitioner abstact class with a single method can be extended to write the custom partitioner

```
public abstract class Partitioner<KEY, VALUE> {
      public abstract int getPartition
    (KEY key, VALUE value, int numPartitions){
            return;

    }
}
```

Combiner

- **Issue:** large number of key-value pairs $\rightarrow$ If copy across network $\rightarrow$ network congestion is more $\rightarrow$ then the intermediate data is greater than input data
- **Solution:** Optimization is required
- Allow local aggregation (after mapper) before shuffle sort
- Example: Word Count - Aggregate (count each word locally) $\rightarrow$ intermediate = Number of unique words
- Executed on same machine as mapper – no output from other mappers
- Results in a "mini-reduce" right after the map phas
- (k,v) of same type as input/output
- If operation associative and commutative, reduce can be combiner
- Reduces key-value pairs to save bandwidth

Combiner

- Combiners are an optimization in MapReduce that allow for local aggregation before the shuffle and sort phase.
- When the map operation outputs its pairs they are already available in memory.
- For efficiency reasons, sometimes it makes sense to take advantage of this fact by supplying a combiner class to perform a reduce-type function.
- If a combiner is used then the map key-value pairs are not immediately written to the output.
- Instead they will be collected in lists, one list per each key value.
- When a certain number of key-value pairs have been written, this buffer is flushed by passing all the values of each key to the combiner's reduce method and outputting the key-value pairs of the combine operation as if they were created by the original map operation.

Combiner – Example

- A word count MapReduce application whose map operation outputs (word,1) pairs as words are encountered in the input can use a combiner to speed up processing.
- A combine operation will start gathering the output in in-memory lists (instead of on disk), one list per word.
- Once a certain number of pairs is output, the combine operation will be called once per unique word with the list available as an iterator.
- The combiner then emits (word, count-in-this-part-of-the-input) pairs.
- From the viewpoint of the Reduce operation this contains the same information as the original Map output, but there should be far fewer pairs output to disk and read from disk.

When combiner is used:

- Combiner helps to reduce the amount of data transferred from the mapper to the reducers and save as much priceless bandwidth as possible.
- In many cases it can dramatically increase the efficiency of mapreduce job.
- Because combiner is an optimization, Hadoop does not guarantee how many times a combiner will be called.
- It can be executed zero, one or many times, so that a given mapreduce job should not depend on the combiner executions and should always produce the same results.
- How many times combiner is called depends on the size of a particular intermediate map output and a value of min.num.spills.for.combine property.
- By default 3 spill files produced by a mapper are needed for the combiner to run, if combiner is specified.

Commutative and associative

- A binary operation is commutative if changing the order of the operands does not change the result.
- A binary operation on a set S is called commutative if: x * y = y * x for all x, y in S
- A binary function f : A $\times$ A $\rightarrow$ B is called commutative if: f(x,y) = f(y,x) for all x,y in A
- A binary operation on a set S is called associative if it satisfies the associative law: (x * y) * z = x * (y * z)
- Example of functions both commutative and associative include: sum(), max().
- What about mean? median?

MapReduce with Partitioner and Combiner