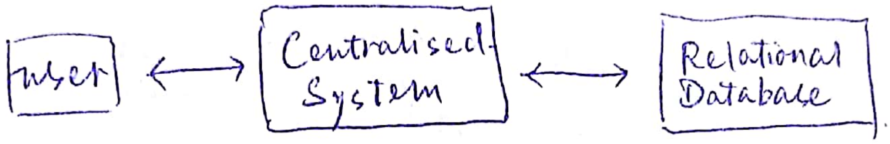
MapReduce Tutorial.

I. Introduction.

☒ Why MapReduce?

Traditional Enterprise System

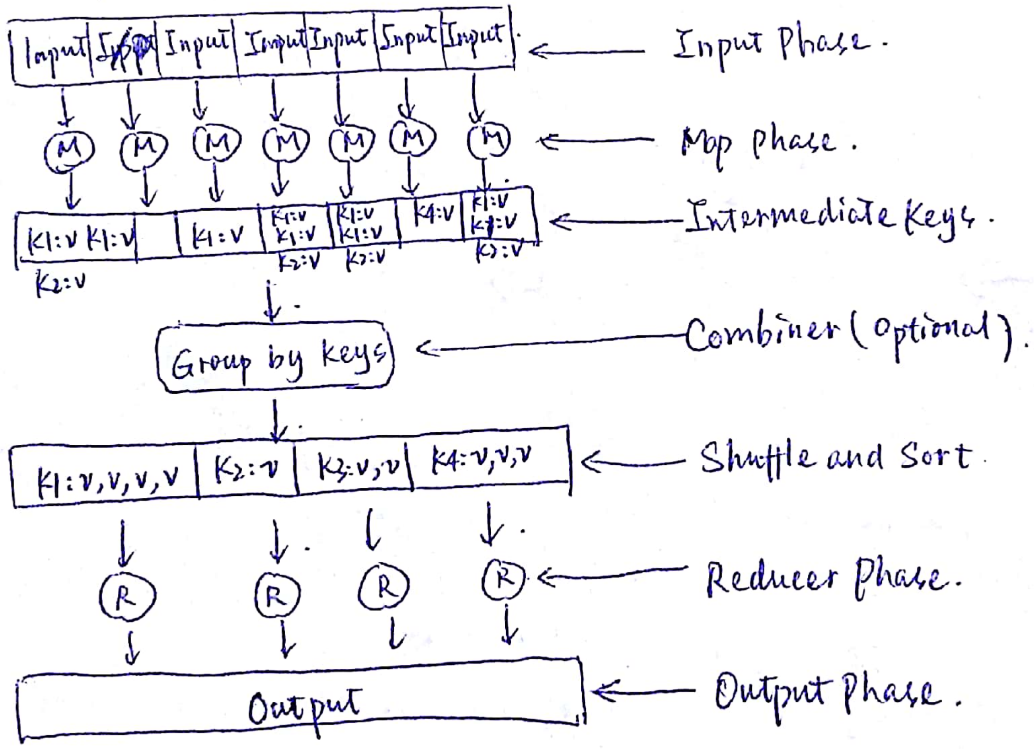


Limitation: Centralised system creates Too much of a bottleneck while processing multiple files simultaneously.

Solution: MapReduce Algorithm.

MR. divides a Task into small parts andassign them to many. computers. Results are collected at one place and integrated To form the result dataset.

☒ How MapReduce works?



Input Phase: Record Reader. parses data to the mapper in <Key, value>

Map: a user-defined function, <key, value> → new <key, value>

Combiner: In a mapper, combiner is a local reducer.
<intermediate key, value> → combiner does aggregation.

Shuffle and Sort: Reducer task starts here.
Grouped <k, v> pairs → local machine (Reducer here).
<k, v> sorted by key → data list.

Reducer : grouped. key-value paired data ~~as input~~.

↓.

Reducer function on each of them.

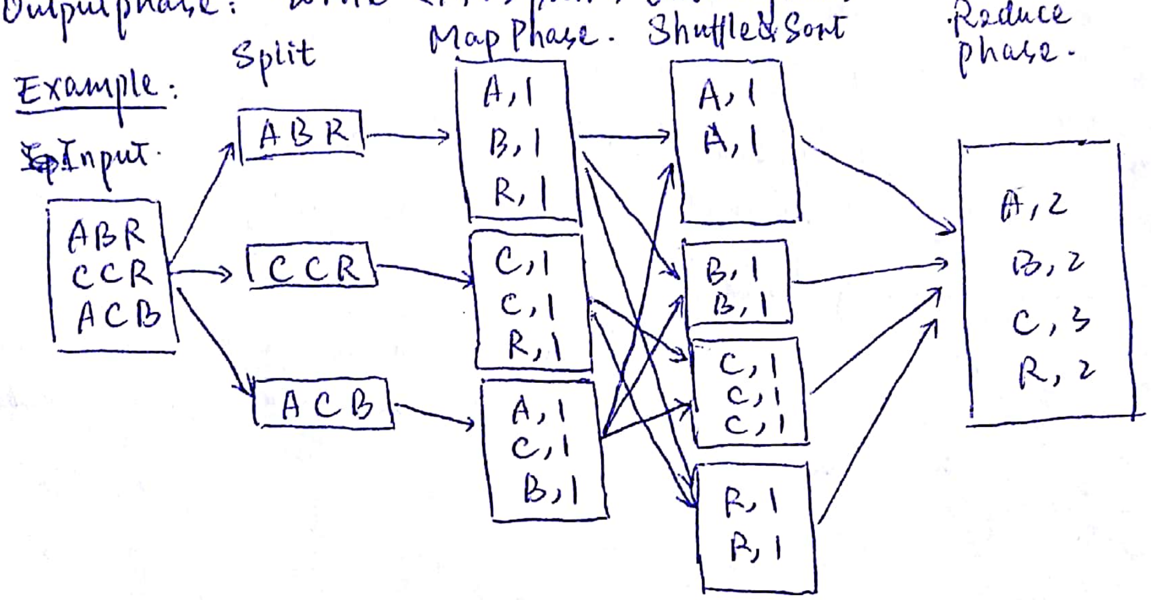
aggregated, filtered, combined.

↓.

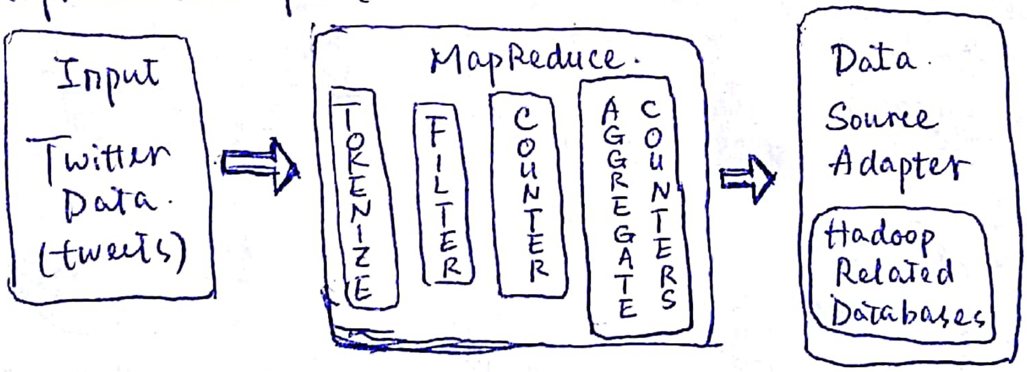
new <k,v> pairs.

Outputphase: write <k,v> pairs onto a file.

Example:



MapReduce Example (twitter).



Tokenize : tokenizes tweets into maps of tokens. <k,v> pairs

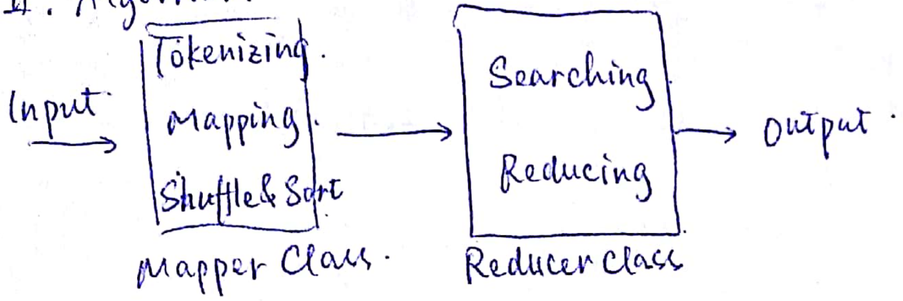
Filter : filters unwanted words, writes filtered words → <k,v>.

Count : generates a token counter per word.

Aggregate counters: prepare an aggregate of similar counter values.

MapReduce Tutorial

II. Algorithm.



1. Sorting: output of mapper class by their keys. <k,v>.
2. Searching: combiner phase & reducer phase.
3. Indexing: inverted index, TF-IDF. (Term freq. - Inverted doc. freq.).

$$TF_{(the)} = \frac{\text{\# of 'the' terms in the doc.}}{\text{\# of terms in the doc.}}$$

$$IDF(the) = \log_e \frac{\text{total \# of documents}}{\text{\# of documents with term 'the'}}$$

III Installation (skip).

IV. API.

1. JobContext Interface:
   - super interface for all classes.
   - defines different jobs in mapreduce.

   Sub interfaces:
   - MapContext < KEYIN, VALUEIN, KEYOUT, VALUEOUT >.
   - ReduceContext < . . . >.

2. Job Class:
   - most important API in MapReduce.
   - configure the job, submit it, control its execution, query the state.

3. ~~Constructors of Job Class~~.

3. Mapper Class.
   - defines map job. input <key, value> ⟶ intermediate <key, value>

4. Reducer Class:
   - defines reduce job.
   - 3 primary phases

     (1) Shuffle: copies the sorted output from each mapper using HTTP across the network.

     (2) Sort: merge-sorts reducer inputs by keys. Shuffle and sort phases occur simultaneously.

(3). Reduce : In this phase, the reduce (object, iterable, Context) method is called for each < key, (collection of values) > in the sorted inputs.

- reduce method is called once for each key value.

V. Hadoop Implementation.

1. MapReduce Algorithm.
   - Hadoop sends Map and Reduce tasks to appropriate servers in the cluster
   - most computing comple takes place on nodes with data locally.
   - after computation completes, collect to form appropriate result and sends it back to the Hadoop server.

2. Input and Outputs.
   - <K,V> pairs as input and output.
   - Serializable → Writable interface.
   - key classes implement the Writable Comparable interface to facilitate sorting.

Input <K1,V1> → map → Intermediate <K2,V2> → reduce ↓ output <K3,V3>

3. MapReduce Implementation. (skip).

VI. Partitioner.

Properties :

- takes place before after ~~Mapper Class~~ Map phase and before Reduce phase.
- # partitioner = # reducer, determine # partitioner according to # reducer.
- partitions intermediate <K,V> from mapper.
- works like a hash fn.

VII. Combiners.

Properties:

(1) semi-reducer.

(2) summarize map output records with the same key.

(3)* reduce the volume of data transfer between map and reduce.

Note: usually map output volume is high.

How combiner works?

(1) no predefined interface. implement the reducer interface's reduce() method.

(2) produce sumary info. from a large dataset because it replaces the original Map out.

<u>MapReduce Tutorial</u>

VII. Combiner (con't).

How combiner works?

(3) Usually the code and operation for a combiner is similar to that of a reducer.

*(4) The combiner phase takes each key-value pair from the Map phase, processes it, and produces the output as key-value collection pairs.

e.g. wordcount.java.