Map Reduce

Simplified Data Processing on Large Clusters

1. Definition

MapReduce is a programming model and an associated implementation for processing and generating large data sets.

Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs

And a *reduce* function that merges all intermediate values associated with the same intermediate key.

Why MR? Because we need fast computing.

2. Programming Model

Inputs/Outputs: key/value pairs

Map: written by user, takes an input pair and produces a set of intermediate key/value pairs

Reduce: written by user, accepts an intermediate key I and a set of values for that key

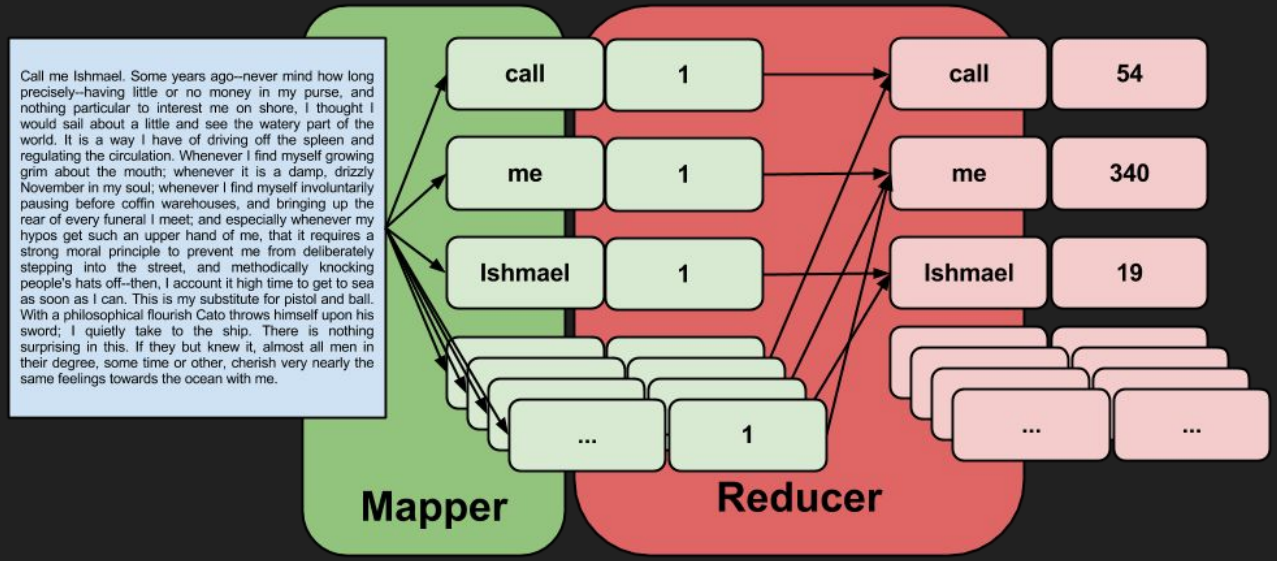
Map (k1, v1) => list(k2, v2)

Reduce (k2, list(v2)) => list(v2)

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

2. Programming Model, Word Count Example



1. split (system)
2. map (user define)
3. shuffle (system)
4. reduce (user define)
5. combine & output

Shuffle: the process of partitioning by reducer, sorting, and copying data partitions from mappers to reducers(No randomness)

2. Programming Model, More Example

- Distributed Grep
- Count of URL Access Frequency
- Reverse Web-link Graph
- Term-Vector per Host
- Inverted Index
- Distributed Sort

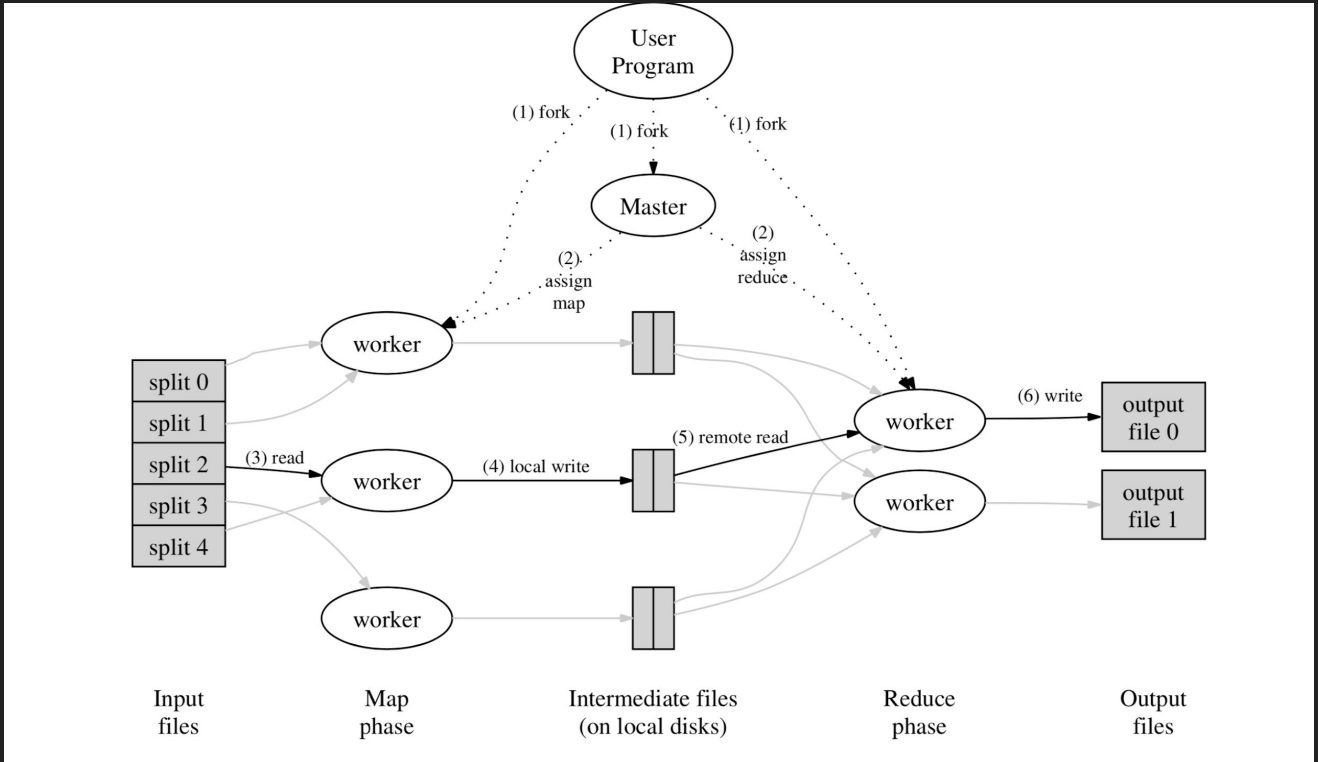
3. Implementation

Structure: Large clusters of commodity PCs connected together with switched Ethernet.

Execution Overview:

master & workers

reducer: sort by intermediate key (sorting is performed in stages, SSTables&LSM Trees)



3. Implementation

Master Data Structure: for each map/reduce task, store state(idle/in-progress/completed)

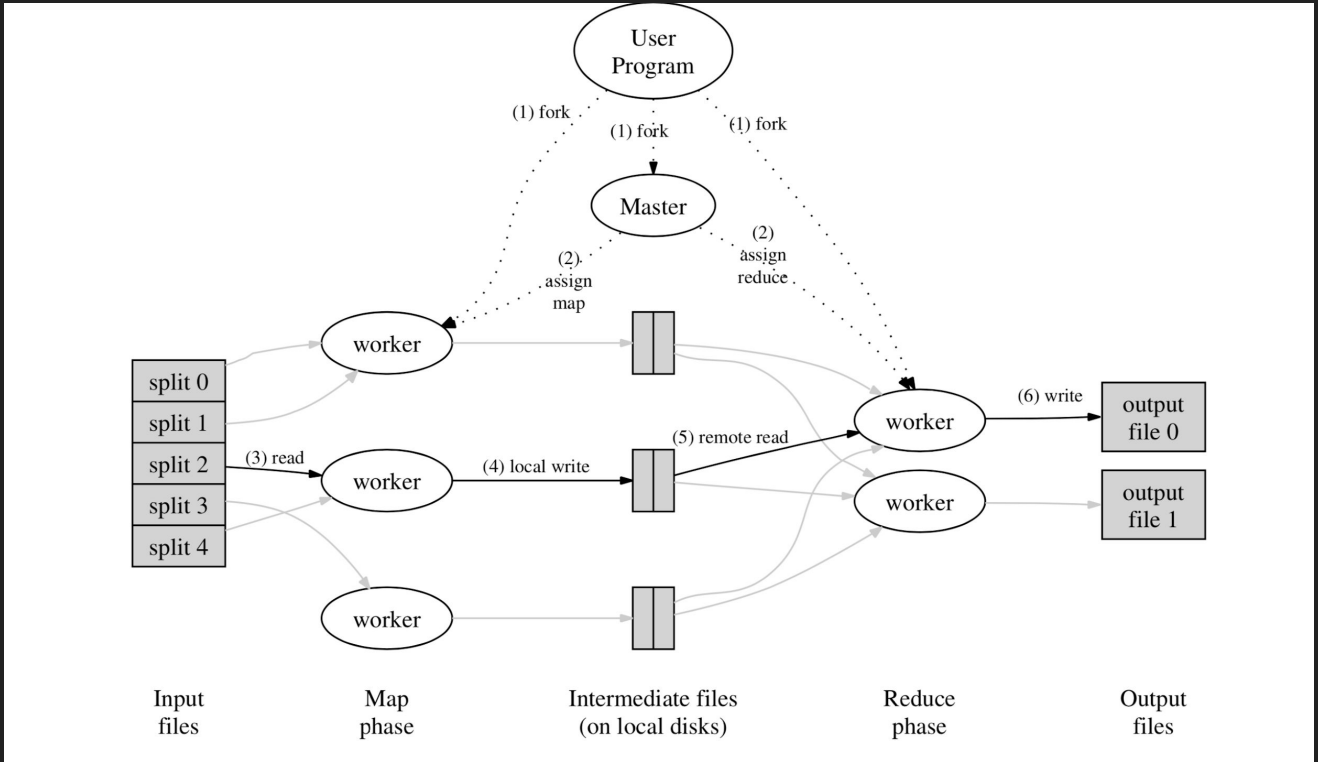
Also stores location&size of intermediate files

Fault Tolerance:

worker failure(ping)

master failure(checkpoints)

semantics in the presence of failures(nondeterministic)



3. Implementation

Locality: GFS stores n copies on different machines. Most input data is read locally, consuming no bandwidth.

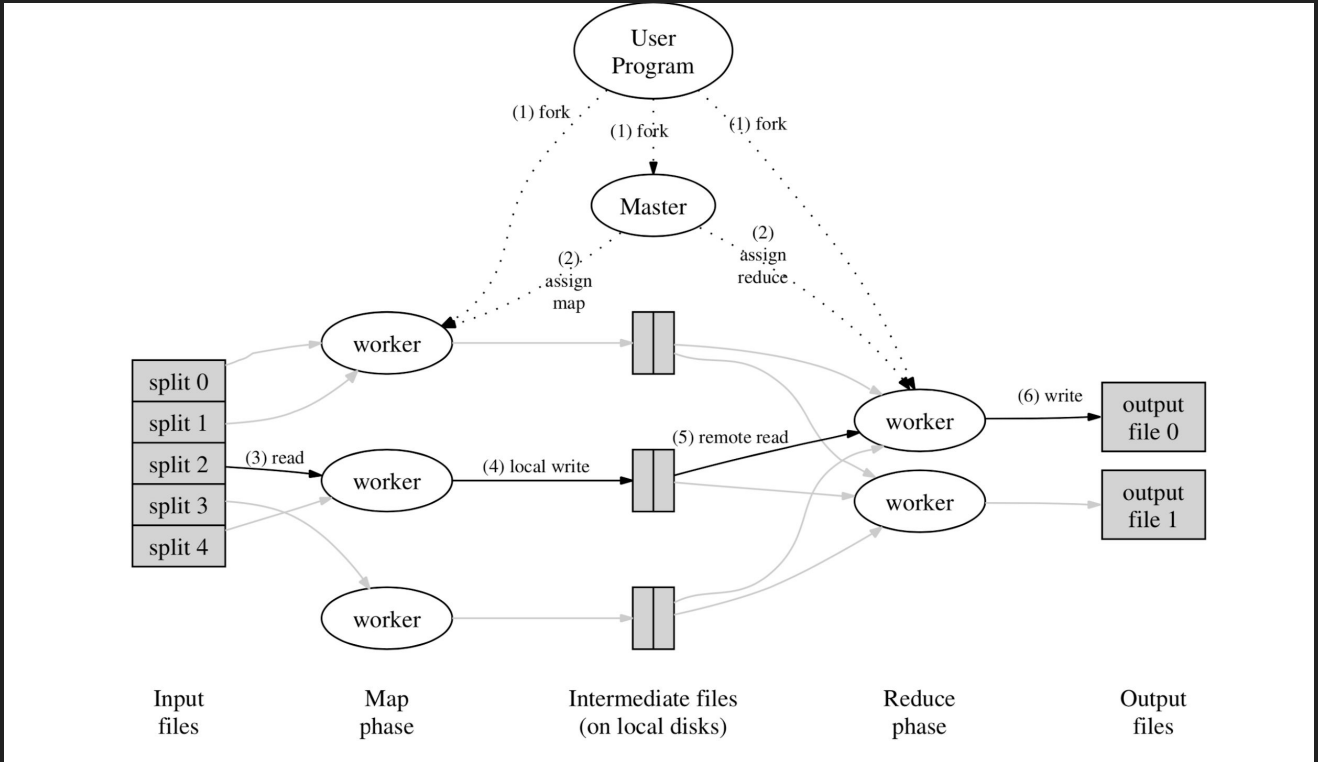
Task Granularity:

M, R > workers

O(M+R) / O(M*R)

Backup Task:

Straggler is bad.



4. Refinement

Partition Function: for example, hashing(key) mod R, same key same reducer

Ordering Guarantees: inc-key order in one partition

Combiner Function: partital merging during map task

Input/Output Type: reader interface

Side-effects: produce auxiliary files. but atomic??

Skipping Bad Records: Signal handler. Master would skip repeated failure on some bug

Local Execution: for debugging and testing

Status Info: Master runs an internal HTTP server + status page

Counter: count occurance of events for sanity check

Skip 5 & 6, performance & experience