Unit-5 - Map Reduce

Map Reduce:

- programming model associated implementation for processing & Generating large data sets
- Developed - Google - for large scale data processing in distributed computing environments
- Dividing large data set into smaller chunks, Processing them in a parallel Process across multiple nodes in a distributed system & combining into final output

Two stages:-

- Map
- Reduce

Map:- Each node in a system process a partition of input data & generates a set of intermediate key value pairs

- Typically specified by used in the form of map fn, which take a key - value pair & produces a set of intermediate key - value
- The intermediate key value shuffled & sorted based on their keys & send to reduce stage

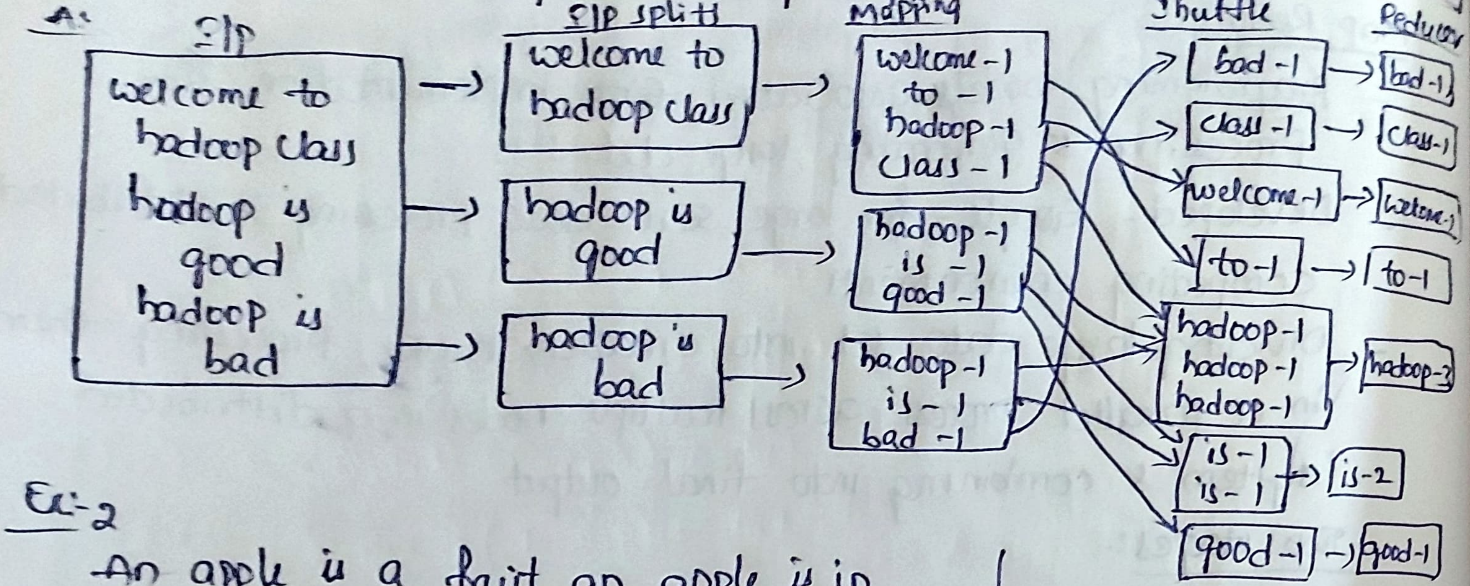
Reduce:- Process inter key value produced by map stage & generate final output.

Algorithms using MapReduce:-

- Matrix vector multiplication
- word count Map reduce Architecture
- MR has following phases
  - I/p splits
  - Mapping
  - shuffling
  - sorting
  - Reducing

Ex:-1 welcome to hadoop class hadoop is good hadoop is bad (word count)

A:



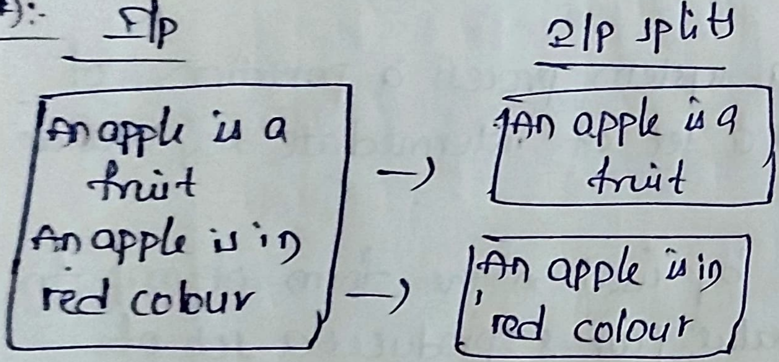
Ex:-2

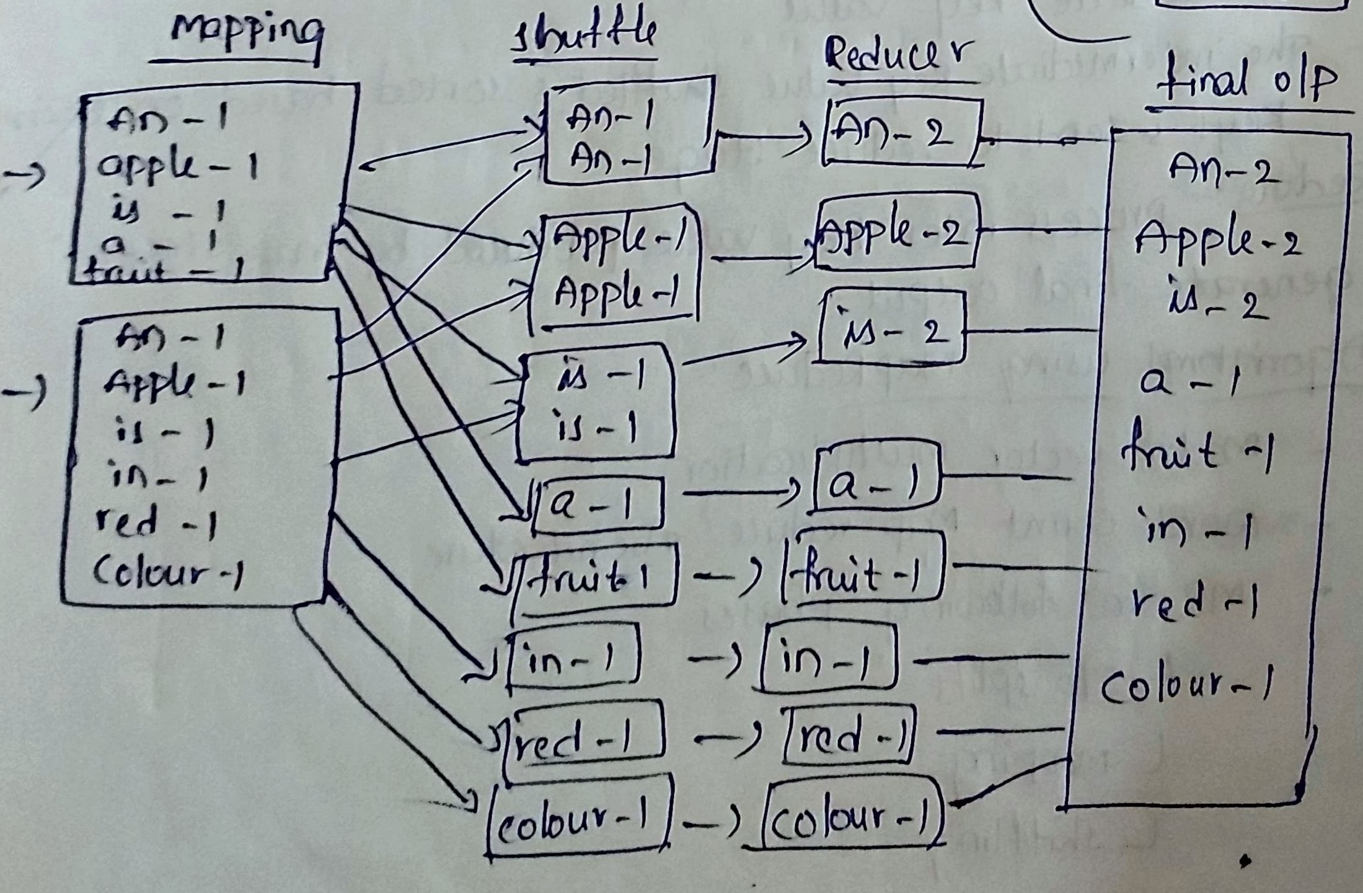
An apple is a fruit an apple is in red colour.

final o/p

bad-1
class-1
welcome-1
to-1
is-2
good-1
hadoop-3

A:





YARN:-

- Yet Another Resource Negotiation
- Cluster management component
- Large scale distributed os for big data appl
- yarn- resource manager created by separating the process engine & mgt far of MR

└ monitors & manages workload

YARN Architecture:

└ splitting job tracker responsibility mgt (management) & job scheduling | monitoring into separate doemons

Concept:

↳ Application: job submitted to the system

eg. MR job

↳ container: Basic unit of allocation

- Resource allocation across multiple resource type eg:- 6cpu, container - 0: 2GB

Components

↳ client: submitting MR jobs

↳ Resource manager: manage use of resource across cluster

↳ 2 components < schedular / Application manager

*schedular: schedular of RM decides allocation of resource to running applications

Application manager: → Accepting job submission

→ Negotiating - in appli master

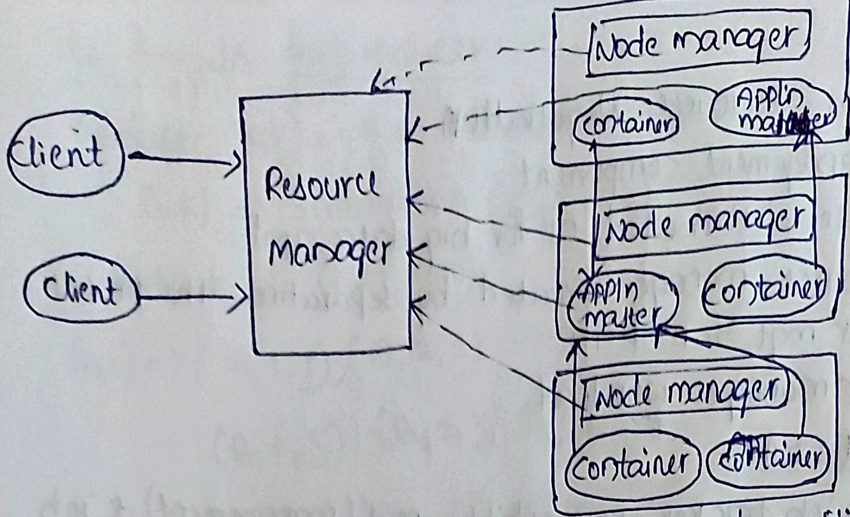
→ Restarting

Node manager: launching | monitoring component containers on machines in cluster

- Monitors resource usage

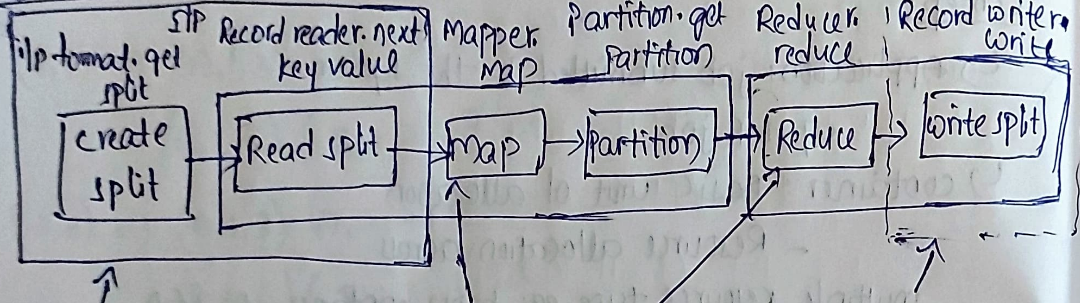
└ MR application cluster: checks task running the MR job

- Application master ~~of~~ & MR task run in container that are scheduled by resource manager & managed by node manager



Understanding i/p's and o/p's in MR:-

The partitioner job is typically format map o/ps to reducers

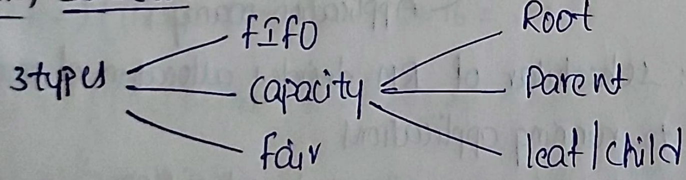


The i/p format & Record reader are responsible for determining what data to feed into map fun

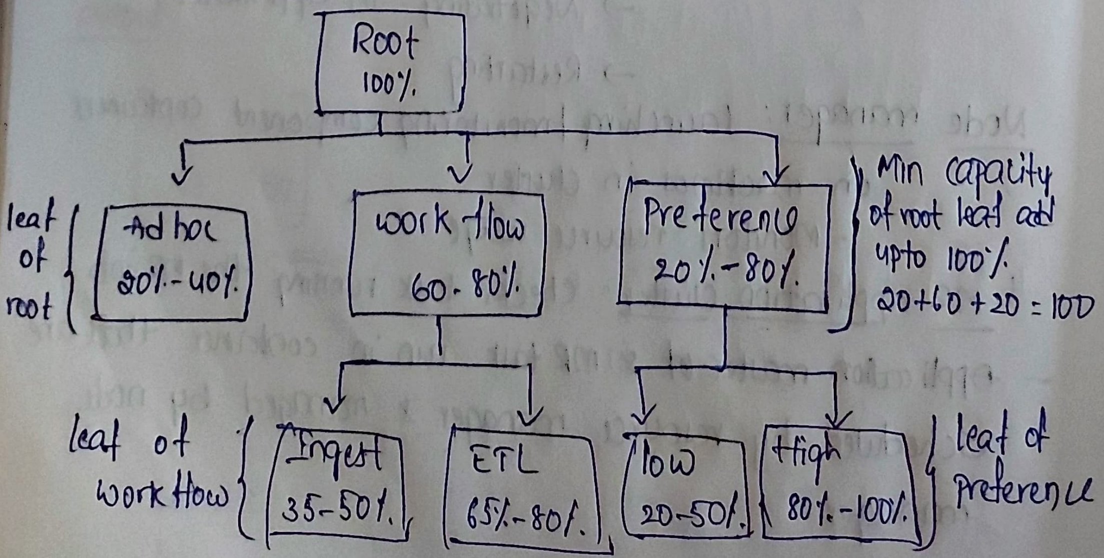
The map & reduce funs are typically written by the user to address the specific use case

The record writer writes the reduce o/p to the destination data sink, which is the final resting place of the MR data flow

Yarn schedular:



capacity schedular:


Root 100%
leaf of root
Ad hoc 20%-40%
work flow 60-80%
Preference 20%-80%
Min capacity of root leaf add upto 100%
20+60+20 = 100
leaf of work flow
Ingest 35-50%
ETL 65%-80%
low 20-50%
High 80%-100%
leaf of preference