EXC Notes
S0936300

December 3, 2013

1 MapReduce

Map reduce is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster. It is not simply Map and Reduce functions(like in functional languages), but the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine once [1].

Properties:

- Assumes large numbers of cheap, commodity machines
- Failure is part of life
- Tailored for dealing with Big Data
- Simple
- Scales well

Who uses it?

1. Google, Facebook, Twitter
2. IBM
3. Amazong Web Services
4. Edinburgh
5. Many small start-ups

Map reduce is composed of the Map() function which performs filtering and sorting as well as the Reduce() function which performs a summary operation(such as counting the number of students in each queue).

1. Map(): Master node takes input, divides into smaller sub-problems and distributes these to the workers. Workers can sub-divide again. Once a worker has it's answer, it passes this to the master node.

2. Reduce(): Master node then collects answers to all the sub-problems and combines to form the output.

1.1 Critique

Considerations as to whether MR can replace parallel databases. Parallel databases have been in development for over 20 years, they are robust, fast, scalable and based upon declarative data models.

MR is not really suited for low-latency problems as it's batch nature and lack of real-time guarentees means you shouldn't use it for front-end tasks.

MR is not a good fit for problems which need global state information. Many Machine Learning algorithms require maintenance of centralised information and this implies a single task.

Which application classes might MR be a better choice than a P-DB?

- Extract-transform-load problems
- Complex analytics
- Semi-structured data(no single scheme for the data, I.e logs from multiple sources)
- Quick and dirty analyses

Results indicated

- For a range of core tasks, P-DB was faster than Hadoop. P-DBS are flexible enough to deal with semi-structured data (unclear whether this is implementation specific)
- Hadoop was criticised as being too low-level
- Hadoop was easier for quick-and-dirty tasks.

  - Writing MR jobs can be easier than complex SQL queries.
  - Non-specialists can quickly write MR jobs

- Hadoop is a lot cheaper

2 MapReduce2

2.1 Programming model

MR offers one restricted version of parallel programming.

- Coarse-grained
- No inter-process communication
- Communication is (generally) through files

Input data is divided into **shards**. The map operation then works over the shards and emits key-value pairs(can be anything which can be represented as a string).

Key-value pairs are then hashed. All those with the same hash-value are sent to the reducers. The input into the reducer is then sorted on it's key so that key-value pairs are locally grouped together.

Each mapper and reducer runs in parallel[5]. There is no state sharing between tasks and task communication is achieved using either external resources or at start-time. There need not be the same number of mappers as reducers. It is possible to have no reducers.

Tasks read their input sequentially as sequential disk reading is far more efficient than random access.

Reducing starts once mapping ends. Sorting and merging can be interleaved.

Example :Count the number of words in a collection of documents.
Mapper: for each sentence, it emits the word and the value '1'
Reducer: takes input and gives a partial count for it's input (see slide 15)

2.2 Map Reduce Efficiency

MR algorithms involve a lot of disk and network traffic:

- Typically start with big data

- Mappers can produce intermediate results that are bigger than the input data
- Input may not be on the same machine as that task which implies network traffic
- Per-reducer input needs to be sorted.

Sharding might not produce a balanced set of inputs for each reducer, they are often skewed. Having an imbalanced set of inputs turns a parallel algorithm into a sequential one(why?)[5].

Selecting the right number of mappers and reducers can greatly improve speed. More tasks mean that each task might fit in memory/require less network access and that failures are quicker to recover from. Fewer tasks mean having less of an over-head. But this is all a matter of guess-work.

Algorithmically, we can:

- Emit fewer key-value pairs: tasks can locally aggregate results and periodically emit them(combining)
- Change the key: Key selection implies we partition the output. Some other selection might partition it more evenly.

3 Hadoop

Apache hadoop framework is composed of the following:

1. Hadoop Common: Contains libraries and utilities needed by other hadoop modules
2. HDFS: A distributed file-system that stores data on the commodity machines, providing very high aggregate bandwidth across the cluster.
3. Hadoop YARN: A resource-management platform responsible for managing compute resources in clusters and using them for scheduling of user's applications.
4. MapReduce: a programming model for large scale data processing.

All modules in hadoop are designed with the fundamental assumption that hardware failures are common and thus should be automatically handled in the software by the framework. Hadoops HDFS and Mapreduce were originally derived from google's HDFS and map-reduce.

HDFS

Files are stored in HDFS (Hadoop File Distributed System). Files are stored as chunks or blocks which are by default 64mb. A file that is 150mb for example will be split up into blocks of 64:64:22. Each block is given a unique name and number and stored on a different *datanode* in the cluster. The *namenode* contains meta data as to where each block is stored and which blocks make up a file.

Is there a problem?

- Network failure
- Disk failure on DN
- Disk failure on NN
- It is not a problem that block sizes may differ (i.e file split into 64, 64 where 22 differs)
- It is not a problem if not all DN(datanodes) are used.

In order to deal with failure on DN, Hadoop replicates each block 3 times when stored in HDFS. When a datanode fails it isn't a problem as there are two other copies. Hadoop realises this is under-replicated and replicates the file again to ensure there are always 3 copies of the file available.

What if there are problems on the NameNode?

- If there is a network failure, data inaccessible
- If there is a problem on the disc drive, the data is lost forever(how do we know where blocks are stored?)

One method was to store data on the NN and also NFS. Now there are commonly two name nodes an 'Active' name node and a 'Standby' name node which is used if the active NN fails.

3.1 Lab 1

Number of maps	Number of samples	Time	Result
2	10	32.49	3.8
5	10	32.29	3.28
10	10	35.2	3.20
2	100	47.367	3.12

4 Other concepts

4.1 BigTable

BigTable is a form of Database.

5 Definitions

Definition: Granularity

Granularity is the extent in which a system is broken down into small parts. It is the extent to which a larger entity is sub-divided. For example a yard broken down into inches has finer granularity than a yard broken into feet

Definition: Coarse-grained

Consist of few, larger components

Definition: Fine-grained

Smaller components of which larger ones are composed

Definition: Cluster

Large number of nodes(computers) that are on the same local network and use the same hardware

Definition: Moore's law

Computing power doubles every 18 months.

Definition: Kryder's law

Storage is growing even faster than Moore's law

Definition: Cell

A grouping of servers, admins, users and clients

Definition: HDFS: Namenode

The namenode contains information as to which chunks/blocks of files are stored where

Definition: HDFS: Datanode

The datanode is where files/blocks in HDFS are stored

Definition: NFS

Network File System. A method of mounting a remote disk.

6 Questions

Question: Do we use Hadoop with MapReduce?

MapReduce libraries have been written in many programming languages with different levels of optimization. A popular open-source implementation is Apache Hadoop [1]

Question: Maps and Reducers run in parallel? Does it mean that they run together constantly or run one after the other? But this could still happen in parallel as long as each batch is waited on

none

References

[1] Wikipedia

[2] Udacity

[3] Coursera

[4] Re-write and remove detailed garble when know more on the topic.

[5] There's a question about this