**Map Reduce: Simplified Data processing on large clusters**

**Introduction:**

A Programming model to process large datasets collected from a varied set of sources such as crawled documents, web request logs and so on at a quick pace, is Map Reduce. This involves two major operations namely Map, and Reduce steps where data is split into intermediate Key /Value pair and manipulated and regrouped to write into a single final output file.

Advantages include fault tolerance, load balancing and data distribution that helps process data at rapid pace.

**Programming Model:**

**Input file:** A file/document is input as a value to the map function and it generates intermediate key/value pairs. (Object in terms of any OOPs language).

**Map:** Involves split the input file into sets of key/value pairs. (Typical map function works on more sets than a reduce and vice-versa). (Function in terms of any OOPs language).

**Reduce:** Receives the intermediate key values generated by the map function and reduces the values by manipulating the values in the pair. (Function in terms of any OOPs language).

This can be written as below: map (k1,v1) → list(k2,v2) reduce (k2,list(v2)) → list(v2)

**Examples:** Word count in a document, Count URL visits, Inverted index (tracks position of words in a documents and helps in auto-correct) etc.

**Implementation:** Different implementation types are small shared memory machine, NUMA multi-processor, that depends on the required environment.

Distributed systems at Google are commodity hardware, interconnected to perform parallel operation on huge datasets like queries ran per day etc.

**Execution Overview:** The Map function is invoked on the distributed machines by automatically splitting the file into M pieces. These are parallel processed by the machines and the intermediate keys are generated.

The intermediate keys are passed over to a partitioning function and that gives the R splits to be parallel processed by Reduce function.

1. **Process:**  The input file is split into M pieces of size ranging from 16MB to 64 MB. This is copied to other clusters to create a replica.
2. One of the copies of program present in the cluster, is special- the master and the rest are workers. The master picks up idle workers (other machines in the cluster) and assigns each a map tasks and a reduce task.
3. The worker picks up its corresponding split file and passes the file into map function. The intermediate Key/Value pairs are buffered om memory.
4. At regular intervals, these values are written into a local disk with R partitions created by the partitioning function. These locations are passed to master, who shares it with workers to perform reduce tasks.
5. Through remote procedure calls, the workers fetch data from local disks, once instructed by master. Workers passes the data into reduce function, generates intermediate keys that are later sorted and for new intermediate key, the value is passed to reduce function.
6. This output from reduce function is appended to final output file in the worker to mitigate disk partitioning.
7. After completion of MapReduce program, master returns to user code. Now the workers together have R part files that are written to a single file and passed on to another mapreduce function.

**Master Data Structures:** Master keeps track of state (idle, in-progress, completed) and identity of each workers performing a map and reduce tasks separately. For each completed map tasks, the location of files is shared with master, so that workers with in-progress reduce tasks are updated on the next files.

**Worker Failure:** Master pings workers at regular intervals to keep track of the state. If a worker failure happens, the state of worker is changed to idle and the work is re-executed by a different worker. All worker that perform reduce tasks are notified on the status as re-execution, thus resilient to large scale failovers.

**Master Failure:** Master writes to the checkpoints of the operation using master data structures. Even if a master fails, the operation can be started from the last completed checkpoint, however, since there is only one master in this model, the programmer need to abort and restart the operation from the latest checkpoint.

**Semantics in the presence of failures:**

Atomic commits are used to derive deterministic values from the mapreduce execution.

If same reduce task is executed on multiple workers, multiple rename calls will be posted on final output file

**Locality:**  Mapreduce master contains the location information of all the files bits present across all nodes. By assigning the worker the location of reducer file, I/O bandwidth is save and so improves performance.

**Task Granularity:**  More maps than reduces, helps achieve task granularity.

**Backup Tasks:** Straggler is the delay that is caused due to I/O, CPU performance, multiple tasks etc. on a machine that slows the map reduce operation. Bugs in machine initialization code that has disabled cache, caused a difference time about a factor of 100. Running back up tasks marks a task as complete if the primary or back up task completes. This method saves about 44% of total run time.

**Refinements:** Some improvements to the code such as including a partitioning function, combing function, order guarantees help speed up the process significantly. partitioning function involves Hashing the input to reduce tasks, combining function is about doing reduce operations even before passing the values over the reducer. Local execution is nothing but creating a sequential execution of all mapreduce operations on a single local machine. This helps in debugging, small scale testing and profiling of data. The master generates the status of all the operations and shares it via a HTTPS server, which is helpful in understanding the failure tasks and workers affected.

**Experience:** Applications of MapReduce at Google involved computation of large scale machine learning problems, largescale graph computations.

**Related Work:** Sorting facility, part of MapReduce library is similar to NOW-sort. Re-execution system is similar to TACC.