**Abstract**

In this paper we explore the problem and solution of MapReduce: a common phenomenon where a small number of mappers or reducers take significantly longer time than the others to complete. The effects of these stragglers include unnecessarily long wall-clock running times and sub-optimal cluster utilization. Straggler avoids executing job in a poorly running node. Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

**Keywords**: MapReduce, Stragglers, Hadoop, LATE, Apache, Big Data

**Introduction**

MapReduce is emerging as an important programming model for large-scale data-parallel applications such as web indexing, data mining, and scientific simulation. Hadoop is an open-source implementation of MapReduce enjoying wide adoption and is often used for short jobs where low response time is critical. Hadoop’s performance is closely tied to its task scheduler, which implicitly assumes that cluster nodes are homogeneous and tasks make progress linearly, and uses these assumptions to decide when to speculative ly-re-execute tasks that appear to be stragglers of “stragglers”[1].

Distributed computing accomplished broad appropriation because of consequently parallelizing and transparently executing tasks in distributed environments. Stragglers tasks is an essential test confronted by all big data processing frameworks for example MapReduce, Dryad, and Spark. Stragglers are the assignments that run much slower than different tasks and since a job completes just when it's last undertaking completions, stragglers postponement work completion. [2] Stragglers extraordinarily impact little occupations such that employments comprising of a couple of undertakings? Stragglers are another common performance issue in MapReduce-like systems. We define a straggler as a task which is disproportionately slow when run on a particular node, as opposed to a task that is slow because it must
perform more computation than its peers. Stragglers may be caused by hardware failures, transient software issues, or clusters composed of a heterogeneous mix of hardware.

Stragglers can significantly impact job performance because time is determined by the completion time of its final task. A job consists of a graph of phases (e.g., map, reduce, and join), with each phase executing the same type of tasks in parallel. We identify stragglers by comparing the progress rates of tasks within All a phase. The progress rate of a task is defined as the size of its input data divided by its duration. In absence of stragglers, progress rates of tasks of a phase are expected to be similar as they perform similar IO and compute operations.

Several techniques have been developed to address stragglers. Recent work has attempted to avoid stragglers by predicting that machines may run tasks slowly and using this information to choose better task assignments [3]. Stragglers may be caused by unpredictable, transient failures, so dynamic approaches must be used to mitigate stragglers at runtime. There is a massive improvement in the computer technology which leads to the development of large new devices. Furthermore, the data itself may be too big to store on a single machine. So to reduce the time it takes fast processing of the data, and to have the minimum storage space to store the data, it is necessary to write programs that can execute on two or more computers and distribute the workload among them. While abstractly the computation to perform may be simple, historically the implementation has been difficult. [4]

Hadoop investigates the load balancing mechanism and straggler problem in MapReduce framework. This is an important area of research because load balancing and straggler problem are the key issues which degrade the performance of MapReduce job. Moreover, this report provides a method to reduce the required memory footprint. Using the proposed methods there will be an improved computation time for MapReduce when these methods are executed on small or medium sized cluster of computers.
Apache’s Hadoop is an implementation of the Google’s Map/reduce. World Wide Web has proved to be the efficient platform for developing applications which are data intensive in nature. As huge volumes of the data are generated day by day, more number of popular applications becomes data-intensive in nature.

**HADOOP**

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. The Apache Hadoop software library is a framework that uses simple programming models for the distributed processing of large data sets across clusters of computers. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures. Apache Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. Hadoop allows running applications on systems with thousands of nodes with thousands of terabytes of data. [3]

**MAPREDUCE**

Map-Reduce was introduced by Google in order to process and store large datasets on commodity hardware. Map Reduce is a model for processing large scale data records in clusters.[5]

MapReduce have two stages which are- [6]

1. **Map ()**:- The master node takes the input, divide into smaller subparts and distribute into worker nodes. A worker node further do as this again that leads to the multi-level tree structure. The worker node process as the m=smaller problem and passes the answer back to the master Node.

2. **Reduce ()**:- The, Master node collects the answers from all the sub problems and combines them together to form the output. It permits programmers to extract from the issues of booking, parallelization, parceling, replication and concentrates on creating their applications. As demonstrated in Figure below.

**FIG 2: Map and Reduce in action**
WHY STRAGGLERS EXISTS

Stragglers are the undertakings that take longer time to execute than other comparative tasks. There are numerous purposes behind the task to take longer time, for example, flawed machines, heterogeneity among hardware, measure of data to process, system blockage and contention for the existing assets. Be that as it may if one task runs slower on a given machine it is not important for the entire present and future task to run slower on that specific machine. Likewise it is not important for a task to be slower all around its execution.

Techniques

Stragglers detection and mitigation algorithms are: Hadoop native scheduler, LATE

The Hadoop native scheduler is designed to run Hadoop applications as a shared, multi-tenant cluster in an operator-friendly manner while maximizing the throughput and the utilization of the cluster.

Traditionally each organization has its own private set of compute resources that have sufficient capacity to meet the organization’s requirements under peak or near peak conditions. This generally leads to poor average utilization and overhead of managing multiple independent clusters, one per each organization. Sharing clusters between organizations is a cost-effective manner of running large Hadoop installations since this allows them to reap benefits of economies of scale without creating private clusters. However, organizations are concerned about sharing a cluster because they are worried about others using the resources that are critical.

LATE:- It addresses the errors brought about in the Hadoop scheduler in a heterogeneous environment. LATE comes up with an alternate system that approximates the completion time for tasks in same class to predict foresee potential stragglers.

CONCLUSION

In this paper we analyzed that Hadoop is the distributed processing environment which is used for processing huge volume of data. It is the partitioning of the data which determines the workload of the reducers. In order to overcome the straggler problem, the output from mappers are divided into smaller tasks larger than the number of reducers and are assigned to reducers in “just-in-time” fashion.

REFERENCES


