A Task Scheduling Approach in Hadoop Environment with Dynamic Load

Hemant Chilhate, Dr. Nishchol Mishra, Priyamwada Sharma
School of Information technology RGPV, Bhopal, India

Abstract **-** With the advent of digital technology and smart devices, a large amount of digital data is being generated every day. Advances in digital sensors and communication technology have enormously added to this huge amount of data, capturing valuable information for enterprises, businesses. This Big data is hard to process using conventional technologies and calls for massive parallel processing. Technologies that are able to store and process exabytes, terabytes, petabytes of data without tremendously raising the data warehousing cost is a need of time. Ability to derive insights from this massive data has the potential to transform how live, think and work.Benefits from Big data analysis range from healthcare domain to government to finance to marketing and many more.

Keyword **-** *Big Data*

I. INTRODUCTION

Big data open source technologies have gained quite a bit of traction due to the demonstrated ability to parallely process large amounts of data. Both parallel processing and technique of bringing computation to data has made it possible to process large datasets at high speed. These key features and ability to process vast data has been a great motivation to take a look into the architecture of the industry leading big data processing framework by Apache, Hadoop. Understand how this big data storage and analysis is achieved and experimenting with RDBMS vsHadoop environment has proven to provide a great insight into much talked about technology [2].

Hadoop has drawn the inspiration from Google's file system (GFS). Hadoop was spun from in 2006 to become a sub-project and was renamed to hadoop. Hadoop does not rely on expensive, high efficiency hardware. Instead it leverages on benefits from distributed parallel processing of huge amounts of data across commodity, low-cost servers. This infrastructure stores as well as processes the data, and can easily scale to changing needs. Hadoop is hypothetical to have boundless scale up capability and tentatively no data is too large to handle by distributed architecture [8].Hadoop is designed to run on commodity hardware and can scale up or down without system interruption. It consists of three main functions: storage, processing and resource management.

It is hard to omit hadoop while talking about big data. Hadoop is the open source software platform managed by the apache software foundation. It's the most widely recognized platform to efficiently and cost-effectively store and manage enormous amount of data.

II. BIGDATA

The amount of data generated every day in the world is exploding. The increasing volume of digital and social media and internet of things, is fueling it even further. The rate of data growth is astonishing and this data comes at a speed, with variety (not necessarily structured) and contains wealth of information that can be a key for gaining an edge in competing businesses. Ability to analyze this enormous amount of data is bringing a new era of productivity growth, innovation and consumer surplus. "Big data is the term for a collection of data sets so large and complex that it becomes difficult to process it using traditional database management tools or data processing applications. The challenges include the areas of capture, duration, storage, search, sharing, transfer, analysis, and visualization of this data".

III. PROPOSED WORK

In this Work the dynamics involved in big data technologies mainly Hadoop, distributed data storage and analysis architecture of hadoop, setup and explore hadoop Cluster on Amazon Elastic Cloud. As well, conduct performance benchmarking on RDBMS and hadoop cluster.

3.1. Apache Spark

A. Apache Spark is a lightning debauchedcluster-computingintended for fast computation. It was constructed on topmost of hadoopmapreduce and it spreads the mapreduce model to resourcefully use additional types of computations thatcontains interactive queries plus stream processing.
B. Industries or Trades are exhausting hadoop widely to analyze their data groups. The motive is that hadoop framework is centered on a modest programming model (mapreduce) and it allows a computing resolution that is accessible, flexible, fault-tolerant and price effective. Here, the key concern is to uphold speed in processing big datasets in terms of waiting time among queries and waiting time to outing the program.
C. Spark was presented by apache software foundation for rapid up the hadoop computational computing software method.
D. As against a mutual belief, spark is not analtered version of hadoop and is not, actually, reliant on hadoop as it has its personal cluster management. Hadoop is impartial one of the methods to implement spark.

E. The chief feature of spark is the situation in-memory cluster computing which upsurges the processing speed of a software

3.2. Evolution of Apache Spark

Spark is a sub-project of Hadoop, developed by MateiZaharia at AMPLab, at UC Berkeley in 2009. BSD was opened in 2010. Apache released Apache software in 2013 became the best apache project in 2014.

3.3. Features of apache spark

These are some features of spark as follows:

- **Speed**
  Spark supports to run a program in the Hadoop cluster 100 times faster and 10 times faster on the disk. This is possible by downloading the number of read / write operations. Deletes intermediate processing data.
- **Supports multiple languages**
  Spark delivers built-in APIs in Java, Python, or Scala. Thus, you can write program in different languages. Spark comes with 80 top operators for interactive surveys.
- **Advanced Analytics**
  Sparks are not just 'Map' and 'Reduce'. It also supports Streaming data, SQL queries, Machine learning (ML) and Graph algorithms.
- **Spark in mapreduce (SIMR)**
  Spark in mapreduce is employed to launch spark task in count to standalone deployment. With SIMR, user can start spark and can use shell without administrative access.

3.4. Resilient Distributed Datasets

The variable distributed database (RDD) is the main information structure of the spark. It is a widespread collection of objects. Each dataset in RDD is divided into logical sections that can be calculated in the different nodes. RDDs can contain python, java, or scaled objects, including custom classes.

An RDD is a collection of read-only shared records. RDDs can be determined by data stored on a stable layer or other RDDs. RDD is a fault-tolerant set of resistance to parallel controls. There are two ways to generate an RDD - Parallel to an existing set of disk software or by referring to a database in an external storage system, such as. A distributed file system, HDFS, HBase or any data source that provides a hiero input format. Spark uses the RDD concept to achieve faster and more efficient mapreduce operations.

3.5. Data Sharing

It shows the way of data shared between the datanodes and namenode and the data sharing is as follows:

- **Data Sharing is Slow in mapreduce**
  MapReduce is generally accepted for processing and production with a parallel distributed algorithm on a large database. This allows users to write parallel calculations with high-end operators without having to worry about workflow and fault tolerance. Unfortunately in the most current context.
- **Data sharing using spark RDD**
  Duplication, serialization and disk IO ensure slower data exchange. Most Hadoop applications use more than 90% of the time in HDFS read and write operations. In exploring this problem, researchers have developed a special frame called apache spark. The main idea in the spark is Resilient Distributed Datasets (RDD), which supports memory development calculations. That is, it keeps the memory status as an object of the task and the object is sharp between these things. Sharing memory information is 10 to 100 times faster than network and disk.

3.6. Iterative operations on mapreduce

Reuse interval results for multiple calculations in multistage applications. The illustration below illustrates how the current structure works and when routine operations are performed on mapreduce. This significantly reduces the amount of data, reducing the I / O and serialization of the disk, making the system less responsive.

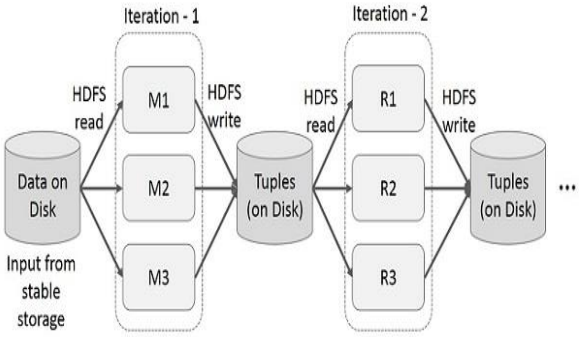

Fig.1 Iterative operation on mapreduce

3.7. Iterative operations on spark RDD

The design given below demonstrations the iterative maneuvers on spark RDD. This will stock intermediate results in a dispersed memory in its place of Stable storage (Disk) and create the system faster.

Note − If the Distributed memory (RAM) is not adequate to store intermediate results (State of the JOB), now it will store those results on the disk.

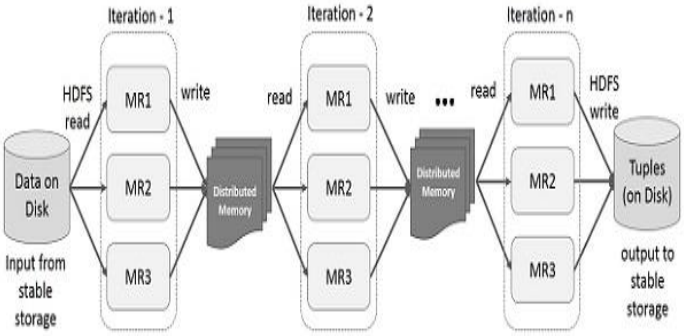

Fig.2 Iterative operation on spark

3.8. Interactive operations on mapreduce

User runs ad-hoc enquiries on the similar subset of data. Every query will do the disketteinput output on the constant storage, which can dominates application execution time.

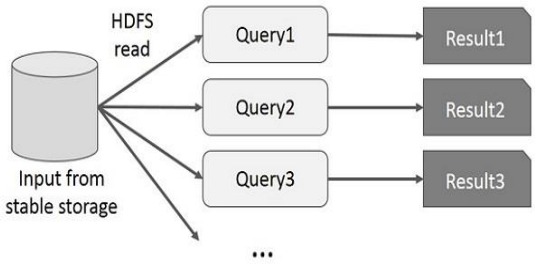

Fig.3 Interactive operations on mapreduce

3.9. Interactive operations on spark RDD

These design demonstrations interactive operations on spark RDD. Uncertainty different queries are perform on the equal set of data repetitively; this specific data can be kept in memory for better execution times.

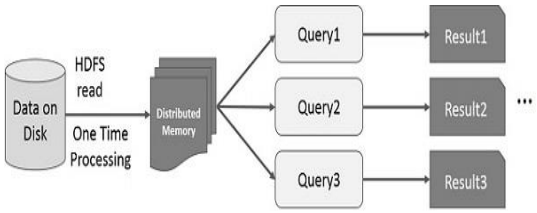

Fig.4 Interactive operations on spark

3.10. *Difference between Spark and Hadoop*
MapReduce

It shows how spark computation is differ from mapreduce and this is as follows-

- **Performance**
  Apache spark processes data in-memory while hadoopmapreduce persists back to the disk after a map or reduce action, so spark should outperform hadoopmapreduce.
- **Speed**
  Apache spark is a lightning-fast clustering computing tool. Spark offers 100x faster memory and up to 10x faster haseoop clusters. It is possible to reduce the number of read / write times and to store the intermediate data in the memory. Mapreduce reads and writes to disk and slows the process speed.
- **Difficulty**
  Spark, with RDD, has a high-level high-end operator, so it's easy to program - Resilient Distributed Dataset In the assignment, developers must register every transaction code that is very difficult to work on
- **Easy to Manage**
  Spark, Batch, Interactive and Machine Learning can form a complete data analysis engine, learn and learn all streaming in the same set. It is therefore not necessary to control different components for each request. Sparks in a set are sufficient to meet all requirements.
- **Real time analysis**
  Real-time event flows; millions of events per second, for example, data can be activated. Share or send Twitter data or Facebook for example. The power of sparks, the ability to operate live streams efficiently Mapreduce does not fail when it comes to processing real-time data, because it is intended to perform aggregation of volume scale data [28]
- **Latency**
  Spark deliverssmall latency computing
  Map Reduce is a great latency computing structure
- **Interactive mode**
  Spark can development data interactively
  MapReduce does not have cooperative mode
- **Streaming**
  Spark can methodactual time data via spark streaming
  With Mapreduce only process data in batch mode
- **Ease of use**
  Spark is cooler to use, its concept (RDD) allows user to process data exhaustinghigh-level operators. It delivers rich APIs in Java, Python Scala.
  Map Reduce is multifaceted; we prerequisite to handle small level APIs to process data thatneeds lots of hand coding.
- **Failure recovery**
  Hadoop is obviously resilient to organism faults or failures since data are written to disk after every operation, but spark has similar built-in resiliency by virtue of the fact that its data objects are stored in something called resilient distributed datasets distributed across the data cluster. "These data objects can be stored in memory or on disks, and RDD provides full recovery from faults or failures," Borne pointed out[28]

3.11. Proposed Methodology

The proposed method develops a task scheduling algorithm on hetrogeneoushadoop clusters in dynamic workload to be reliable and time efficient.The original task scheduling algorithm of hadoop can meet the performance requirements of hadoop clusters but fails to achieve the reliability in task scheduling in hetrogeneoushadoop clusters. In the hetrogeneoushadoop clusters with dynamic change of load at run time the performance is increased by using monitoring module but still the tasktracker cannot make the system reliable. So in this work reliability and time efficiency is considered as the main issue in the hetrogeneoushadoop clusters with task scheduling in dynamic workload.

3.12. A Framework for Proposed Methodology

As discussed earlier the main objective is to make the system reliable and reduce the overall execution time of system by using the spark as a base platform for the purpose of execution of program because spark is a highly reliable and time efficient system based on Resilient distributed dataset (RDD).

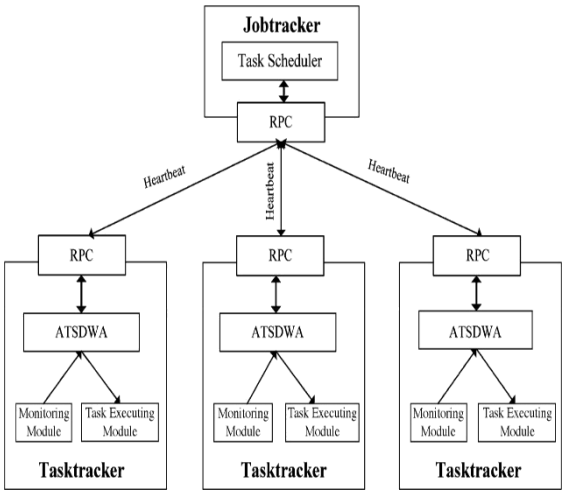

Fig.4: Runtime environment of DWAA

According to preceding analysis, the existed scheduling strategy of Hadoop cannot meet better performance. This concept proposes a novel load balancing algorithm, i.e. DWAA, for the heterogeneous Hadoop. The whole algorithm is working on running clusters. Each tasktracker periodically checks its load, which is based on collected parameters and dynamically adjusts the maximum number of slots for tasks. (MaxTaskCapacity), which replaces the approach of allocating the fixed number of slots for tasks (FixedTaskCapacity). Our strategy changes, each tasktracker measures its own workload in the next heartbeat, and then makes a decision about whether to allocate more tasks on same or not. In this we proposealgorithm for the heterogeneous hadoop clusters. The adaptability here is defined as the way of algorithm working, which means in a running cluster, according to the tasktrackers' own resources and the dynamic change of their load, they can make corresponding adjustment to achieve the optimal state and realize self-regulation[29]

Each tasktracker periodically weighs its load based on the collected load parameters and dynamically adjusts the maximum number ofslots for tasks, which is marked as MaxTasksCapacity, which replaces the approach of allocating the fixed number of slots for tasks (FixedTasksCapacity). Our strategy changes the traditional one heartbeat, full allocation strategy, making the tasktracker able to get part tasks in a heartbeat period only, not providing all capacity to accommodate more tasks,thus improving the dynamic controllability of each node in the cluster. Each tasktracker reassesses its own workload in the next heartbeat and then makes a decision about whether to accommodate more tasks or not[30].

3.13. Algorithm

DWAA: Dynamic workload adjustment algorithm

Input: Current tasktracker load information

Avgclusterload: Calculate the average clusters CPU and memoryload information.

K :Be the total number of node in cluster. Cth, Lth: threshold parameter.

MaxTaskCapacity: total capacity of cluster to run task in parallel.

Output : AskForNewLoad, MaxTaskA, MaxTaskB

```
1.  for(i=0; i<n; i++){
    /* Every cluster contains the n number of nodes, and collect the set of information,depositthem in a corresponding load array. */
2. perCpu[i] = getCpuPercentageUsage;
/* Get CPU percentage utilization of each node */
3. perMem[i] = getMemPercentageUsage;
/*Get memory percentage utilization of each node*/
4.}
5. for(j=0; j<n j++){
7. AvgperCpu = perCpu[j];
8. AvgperMem = perMem[j];
9. }
10 CpuInfo = AvgperCpu/K;
      /*K is the number of node in cluster*/
11. MemInfo = AvgperMem/K;
12. if(CpuInfo<Cth&&MemInfo<Mth){
13. if (CpuInfoA>CpuInfoB )
14. {
/*Compare the CPU load between the clusters */
15.  MaxTaskB = MaxTaskB + no_of_task;
/*Assign the next tasks (no_of_task) on cluster B*/
16.}
17.  else if (CpuInfoA<CpuInfoB ){
18. MaxTaskA = MaxTaskA + no_of_task
19.   }
20. else{
21.  if (MemInfoA>MemInfoB){
/*Comparethe Memory load between the clusters */
22.        MaxTaskB =  MaxTaskB + no_of_task
23.   }
24.  else if(MemInfoA<MemInfoB){
25.    MaxTaskA = MaxTaskA + no_of_task
26.    }
27.        else {
28.   if (MaxTaskCapacityA>MaxTaskCapacityB){
/*Compare the Maximum task running capacity of clusters */
29.        MaxTaskA =  MaxTaskA + no_of_task;
30.       }
31.        else {
MaxTaskB =  MaxTaskB + no_of_task;
}
32.        else
{
33.        MaxTaskA =  MaxTaskA – 1;

34.        MaxTaskB =  MaxTaskB - 1
}
```

IV. EXPERIMENT RESULTS

Provided experiment results are based on the comparison between execution times taken by ATSDWA algorithm and proposed approach DWAA withSPARK. Figure 5 shows the comparative analysis of both the algorithms on 10MB,100MB,200MBdata set in multinodecluster,From the graph it is clear that ATSDWAwithSPARK is performing better in single node cluster and 2 node cluster.

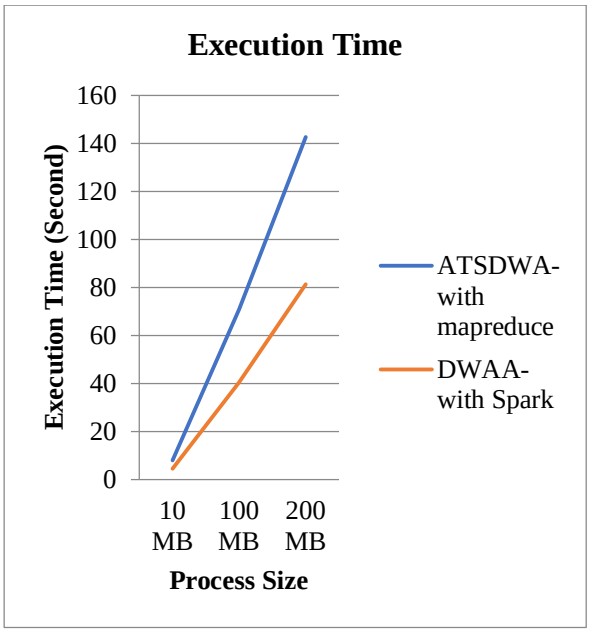

Fig.5 Graphical comparison of mapreducevs spark

Table 1 Comparative Analysis

Proposed Method	Previous Method
4.495	7.945
40.45	70.84
81.25	142.52

V. CONCLUSION

This Paper discussed about Task scheduling algorithm on heterogeneoushadoop cluster with MapReduce programming model by Hadoop.Proposed a new Task scheduling algorithm on heterogeneoushadoop cluster with spark analysis to provide better time than task scheduling with mapreduce.DWAA with Spark is highly efficient and reliable for heterogeneous hadoopclusters. Experiment results validate that ATSDWA significantly benefits both tasktrackers and jobtracker. On the taskertracker's side, task execution time is reduced, node performance is more stable, task failure rate is decreased, and both hunger and saturation are avoided at the same time.On the jobtracker's side, the failure of jobtracker due to overloading can be avoided. In fact, ATSDWA with Spark is applicable to both homogeneous and heterogeneous clusters and can improve the overall task throughput rate of cluster without bringing extra load to tasktrackers.

VI. REFERENCES

[1]. E. Dumbill.What is Big Data? An Introduction to the Big Data, 2012. http:// strata.oreilly.com/2012/01/what-is-big-data.html, accessed April 2017.

[2]. H. Hu, Y. Wen, T.-S. Chua, and X. Li, ''Towards scalable systems for big data analytics: A technology tutorial,'' IEEE Access, vol. 2, pp. 652–687, 2014.

[3]. A. Rabkin and R. H. Katz, "How Hadoop Clusters Break," IEEE Software, vol. 30, pp. 88-94, 2013.

[4]. SaeedShahrivari, "Beyond Batch Processing: Towards Real-Time and Streaming Big Data", Computers, Vol. 3, pp. 117.129, 2014.

[5]. Apache Hadoop.What Is Apache Hadoop?, 2014. http://hadoop.apache.org/, accessed April 2017.

[6]. Wikipedia Apache Hadoop, 2014 http://en.wikipedia.org/wiki/Apache_Hadoop, accessed April 2017.

[7]. Apache Hadoop. MapReduce Tutorial, 2013. https://hadoop.apache.org/docs/ r1.2.1/ mapred_tutorial.html, accessed April 2017.

[8]. Apache Hadoop. HDFS Architecture Guide, 2013. http://hadoop.apache.org/ docs/ r1.2.1/hdfs_design.html, accessed April 2017.

[9]. Rodrigo Agerri,etal.,"Big data for Natural Language Processing: A streaming approach" in Knowledge-Based SystemsVolume 79, May 2015, Pages 36–42.

[10]. YanfeiGuo, JiaRao, Dazhao Cheng, Changjun Jiang, Cheng-ZhongXu and XiaoboZhou, "StoreApp: A Shared Storage Appliance for Efficient and Scalable Virtualized Hadoop Clusters", 9 pages, Hong Kong, China, April 2015.

[11]. S.Johnston.Seminar on Collaboration as a Service Cloud Computing, 2012. http://www.psirc.sg/events/seminar-on-collaboration-as-a-service-cloud-computing, accessed May 2017.

[12]. XiaolongXu, Lingling Cao and Xinheng Wang., IEEE "Adaptive Task scheduling Strategy based on Dynamic workload adjustment for hetrogeneoushadoop clusters" cluster comput, vol.46, no. 19,pp. 162-219,June 2016.

[13]. Xiao Qin, Hong Jiang and ZongrefnHan,"A dynamic and reliability driven scheduling algorithm for parallel real time jobs executing on hetrogeneous cluster" J. Parallel Distrib.Computer . vol. 65, no. 8, pp. 885-900, Aug. 2012

[14]. Mark Yong, NitinGaregrat and ShiwaliMohan,"Towards a resource aware scheduler in Hadoop"in Proc. ICWS, pp. 102–109,2010

[15]. Zhuo Tang, Junqing Zhou, Kenli Li, Ruixuan Li "A MapReduce task scheduling algorithm for deadline constraints," Cluster Comput., vol. 16, no. 4, pp.651–662, Dec. 2013.

[16]. Xiaomin Zhu, Chuan He, Kenli Li and Xiao Qin"Adaptive energy-efficient scheduling for real-time tasks on DVS-enabled heterogeneous clusters,"J.Parallel Distrib.Comput.,vol.72, no. 6, pp.751–763, Jun.2012

[17]. SanthanamSrinivasanand Niraj K. Jha"safety and Reliability Driven Task Allocation in Distributed System" IEEE Trans. Parallel and Distributed System, 10(3), 2010, 238-251.

[18]. K. Kc, K. Anyanwu,"Scheduling Hadoop Jobs to Meet Deadlines," IEEE Second International Conference on

Cloud Computing Technology and Science, 2010, pp.388-392, doi: 10.1109/CloudCom.2010.97.

[19]. Jiang Xie et al., "Improving mapreduce performance through data placement in hetrogeneoushadoopclusters",IEEE in 2010,pp.1-9

[20]. J. Ekanayake, S. Pallickara, and G. Fox, "MapReduce for data intensive scientific analyses," In Proceedings of the 2008 IEEE Fourth International Conference on eEcience, 2008, pp.277 284,doi: 10.1109/eScience.2008.59.

[21]. G. Mackey, S. Sehrish, J. Bent, J. Lopez, S. Habib, and J. wang, "Introducing map-reduce to high end computing," In Proceedings of the 2008 3rd patascale data storage whokshop, 2008, pp.1-6, doi: 10.1109/PDSW.2008.4811889.