© 2014 IJEDR | Volume 3, Issue 1 | ISSN: 2321-9939

# A Survey on Scheduling Techniques in Hadoop

Harshitha R[1], Rekha G S[2], Dr. H S Guruprasad[3]
[1]Asst. Professor, Dept. of CSE, BMSCE, Bangalore
[2]Asst. Professor, Dept. of CSE, BMSCE, Bangalore
[3]Professor and Head, Dept. of CSE, BMSCE, Bangalore

***Abstract* - Cloud computing has major role in parallel and distributed data processing. Hadoop environment is mainly used for storage and processing of such data. In Hadoop, MapReduce framework is a programming model which processes terabytes of data in very less time. MapReduce framework uses a task scheduling method to schedule task. There are various methods available for scheduling task in MapReduce framework. Scheduling of jobs parallely across nodes is a major concern in distrusted file system. Discussion of these research papers emphasizes the scheduling vulnerabilities of existent as well as new techniques in the field of distributed file system.**

***Keywords* - Hadoop, MapReduce, Scheduling, Distributed data processing**

## I. Introduction

In the present day scenario, cloud has become an inevitable requirement for majority of IT operational organizations. Applications such as data storage, data retrieval and data portability have become significant requirements for cloud computing. Cloud computing application handles BigData to overcome these hindrances. A BigData is extremely large data sets that has unstructured, semi-structured and structured data where the data has the potential to be mined for information. Numerous applications are being developed to achieve higher performance. Efficient load balancing, load distribution, optimum resource utilization, minimum overheads and least possible delay have been the vital issues for cloud infrastructure.

Hadoop is an open source framework for distributed data storage and processing big data on cluster of commodity hardware. Users submit jobs to a queue, the cluster process them in the order of jobs submitted. As the input data increases enormously, computation to those jobs will also increase. The increase in computation will provide us the feature like sharing Hadoop cluster among multiple users. Benefits of sharing Hadoop cluster increases the data locality and also cost efficient. Hadoop originally developed by Google and later made an open source implementation by Apache. The Hadoop framework allows developers to focus on computation with parallel processing programs for very large data. Hadoop includes two levels of processing i) Hadoop Distributed File System (HDFS), which stores large amount of data with decreased makespan ii) Hadoop MapReduce: a framework which processes distributed data on clusters. Hadoop is a reliable, scalable, distributed computing platform developed by Apache. It offers local computation and storage which is designed to level up from single datacenter to thousands of machines. Hadoop has a distributed storage system called Hadoop Distributed File System (HDFS). Hadoop provides a MapReduce engine that runs on top of HDFS to process the data. HDFS framework divides large data sets into data chunks. This framework is built on master-slave architecture. The master nodes are called the NameNode and JobTracker. The slave nodes are called DataNode and TaskTrackers.

NameNode, HDFS has a master node called NameNode which controls slave nodes called DataNode. NameNode has all the information of where the data is stored, how data is divided into blocks, on which node the data is placed and the health of the distributed file system. The NameNode is a single point of failure, where each cluster has NameNode. DataNode is a slave node in the Hadoop cluster. It is responsible for data management and reads its data from HDFS. DataNode reports back to the NameNode with data management status and manages data on each physical node. The client application submits jobs to the JobTracker which in turn talks to the NameNode to determine the location of the data. The JobTracker locates available slots in the TaskTracker at or near the data. The job is submitted to the JobTracker to the chosen TaskTracker and monitored. The JobTracker and TaskTrackers communicates using heartbeat signal. If the heartbeat signal is not received in a time interval, they are assumed to have failed and the work is scheduled on a different TaskTracker. The JobTracker updates its status once the work is completed and is submitted to the client application.

In this survey paper, we focus on different types of latest scheduling techniques to provide us an efficient result with less makespan. Scheduler in Hadoop introduced two schedulers for multi user environment. Fair scheduler developed at Facebook and Capacity scheduler developed at Yahoo.

## II. Literature Survey

### 1. Data Locality-Aware Task Scheduling

Engin Arslan et. al. [1] aims to work on reduce task scheduling to improve both data locality and network traffic of data intensive applications. A new scheduling algorithm is been proposed, LONARS, which helps in choosing the optimal task tracker for all the pending reduce tasks. A heartbeat is received from all the reduce task trackers when a job is ready to schedule the reduce task. A cost function is been defined to identify the cost of giving a reduce task to each task tracker. The task trackers are partitioned into groups for each reduce tasks based on the cost function using K mean clustering. The task tracker with lowest cost function is the best candidate for reduce task. They have conducted micro benchmarking to evaluate the performance of LONARS which outperforms the default Hadoop reduce task scheduling algorithm by 25%.

Xiao Ling et. al. [2] proposed Pregather to exploit the spatial locality of virtual machine and disk I/O efficiency in virtual environment while sustaining virtualization transparency. The proposed and implemented algorithm named pregather, which is an adaptive non-work conserving disk scheduling framework with spatial locality - aware heuristic algorithm, SPLA. The SPLA algorithm helps pregather to make decision waiting for new requests. The pregather reduces I/O interfaces by allocating each VM dynamic I/O time slices and schedules VM in a round robin fashion. Pregather schedules the requests a deadline and queues both the sort queue and EDF queue. It also calculates the efficiency for exploiting both regional and sub-regional spatial locality. Pregather holds an intelligent prediction model named VNavigator, which helps in prediction of distribution among sub region under each region and also the arrival time of new requests which accesses these sub regions. The experiment results show a significant improvement of disk throughput with high spatial locality.

Jongse Park et. al.[3] discussed dynamic VM reconfiguration technique, called Dynamic Resource Reconfiguration, DRR, for virtual cluster using Hadoop is been proposed and evaluated. The configuration of virtual computing nodes can be changed using DRR technique which maximize the data locality and also increases the computing resource of a VM. To achieve this two schemes has been implemented, synchronous DRR and queue- based DRR in which both are based on naïve Hadoop scheduler for fairness while data locality is also supported. The requirement of additional VM cores i.e., physical CPUs, to run local tasks is a drawback with respect to DRR scheduling.

Aprigo Bezerra et. al. [4] explains about the increasing volume of experimental information which is to be stored and processed, generates problems related to the management, maintenance. Then, it is necessary to apply techniques to reduce the use of disk and network resources to improve data locality. In classical FIFO scheduling, Fair-share and Capacity scheduler, the problem of locality is not well resolved. Delay scheduling algorithm is proposed to increase data locality in MapReduce applications. Delay scheduling also had some problems to achieve data locality. Job scheduling policy is proposed which is based on prefetching technique for clusters with racks, there increasing data locality when accessing racks does not guarantee data locality when evaluating individual computing nodes. Scheduling technique which recommends the need to make global decisions related to the block allocation data files when a set of applications need to share same input file set, here, they try to solve globally the problems of allocation using integer linear programming techniques. In the proposed policy tasks are scheduled individually trying to place new tasks that handles the same data block used by the task just completed. The search of tasks which is to be assigned to a node occurs between different jobs in a queue. It was developed for example bioinformatics applications like read-mapping applications.

XiangPing Bu et. al. [5] proposed the ILA scheduler which works in a Hadoop virtual cluster. The cluster consists of a number of physical servers each of which are in the same virtualized environment. On each server, Multiple VMs are allocated hosting running applications. The core of ILA-based task management is located in the master which consists of four components.

1. Interference-Aware Scheduling Module (IASM) to initialize the interference between tasks running on co-hosted VMs with the help of an interference prediction model.
2. Locality-Aware Scheduling Module (LASM) maintains good data-locality for map tasks by using Adaptive Delay Scheduling algorithm.
3. The task profiler measures the task demand of each job and gives the information of task to IASM and LASM modules.
4. The ILA scheduler instructs IASM and LASM modules to carry on interference-free-high-locality task management.

Ruiqi Sun et. al. [6] discuss on Virtualized Hadoop cluster which not only simplifies cluster management, but also facilitates cost-effective workload consolidation for resource utilization. In Hadoop System the data locality is very important which impacts the performance of MapReduce applications. Existing task scheduling approach is not so effective in improving data locality, because of two levels distribution of data virtual machine and physical server. Here virtualized Hadoop cluster is deployed in which computing node and storage node are placed in respective virtual module to improve flexibility, a novel task scheduling approach called DSFvH which aims to improve data locality in virtualized Hadoop system. The process migrates the computing node through VM to the physical server running VM.

Saba Sehrish et. al. [7] have proposed MapReduce framework which understands data semantics and performance improvement by reducing MapReduce phases. The framework makes use of default scalability and fault tolerant nature of MapReduce and combines them with access patterns. The proposed access pattern is named as MapReduce Access Pattern (MRAP). MRAP is a unique combination of programming framework and data access semantics used in HPC application based on analytics. MRAP framework is designed based on 3 components, MRAP API, MRAP data restructuring and MRAP data centric scheduling. These overall framework and its design improves data layout and data locality for HPC application on MRAP. Data restructuring framework result shows 70% of performance gain and scheduling result shows an improvement of 18% on independent application and 49% on depending application.

## 2. Task-Aware Scheduling

Xite Wang et. al. [8] proposed an algorithm which outperforms previous scheduling algorithm which rarely considered the job intensive environment and system with higher throughout never achieved. In this paper the proposed algorithm is a throughput driven task scheduler. The throughput driven scheduler satisfies 4 major factors, high ratio of local processing, non-local assignment of computation intensive task, avoiding read data of non-local task, full use of some resources. Throughput driven Scheduler adopt multi job co scheduling model for scheduling jobs. Hot queue data structure is been introduced to avoid hot stop and also conception job status is been introduced to solve idle resource problem for the jobs in processing. Hence the throughput driven scheduler to improve some throughput for MapReduce processing.

Weina Wang et. al. [9] have presented a new queuing Architecture and proposed a map task scheduling algorithm organized by join the shortest queue policy together with the MaxWeight policy. This algorithm helps in achieving data locality for heavy- traffic network. The jobs are scheduled to the shortest queue among the three local machine queues and a queue in the remote machine. The outer bound and the lower bound are been derived based on the capacity region and expected number of backlogged tasks respectively. The full capacity region and minimizes the expected number of backlogged tasks in the considered heavy-traffic is attained by the proposed scheduling algorithm. The technique proposed achieves both heavy- traffic optimality and throughput-optimality.

Yi Yao et. al. [10] they have implemented an efficient resource HaSTE Scheduling policy for scheduling map and reduce tasks using Hadoop YARN. This scheduling policy improved the makespan of MapReduce tasks. HaSTE scheduling is categorized based on fairness and Urgency. Fairness refers to the availability of resources in the system and demand of resources from each task. Whereas urgency refers to the dependency between tasks and impact of each task's progress. Thus the HaSTE scheduling policy helps in better utilization of resources and reduces the total makespan of a job under different workloads.

Fahad R, Dogar et. al. [11] explains that many data center applications perform rich and complex tasks. The tasks comprises of multiple flows that travels different parts of the network at different times. The task-aware network scheduling, groups flows of a task and schedule them together, which reduces both the average as well as tail completion time for typical data center applications. To achieve the benefits, Baraat a decentralized task-aware scheduling is designed. It aims at achieving FIFO-LM scheduling in a decentralized fashion, which do not have any coordination between network switches. In Baraat, each task is assigned a globally unique identifier(TASK) based on its arrival time, task with lower ideas will have higher priority and one with higher ideas will have lower priority based on the identifier, switches can make consist decisions with any coordination. If two switches observe flows of two different tasks, both make the same decision based on flow priority, which achieves consistency over space. If a switch observes flows of two tasks at different times, it makes the same decision, which achieves consistency over time.

## 3. Fairness-aware scheduling

Kenzo Van Craey Nest et. al. [12] proposed Fairness-aware Scheduling to overcome the problem of Current multi-core Schedulers which pin thread to cores, which leads to unfair performance on heterogeneous multicores. For barrier-synchronized multi-threaded workloads, unfair performance leads to threads that are running on a big core to wait at barrier for the other threads running on the small cores which does not lead to compromise quality variability in performance across simultaneously running programs. Fairness can be increased to an extent by dynamically scheduling threads across core types. Yet, fairness is still poor. Fairness-aware scheduling is proposed which optimize for fairness by considering it as the primary objective. Two techniques are proposed to make sure all threads get to run on either core types for equal shares. Equal-time Scheduling schedules threads such that they all spend equal amounts of time on either core type. Equal progress scheduling aims at getting all threads to make equal progress, and three methods are defined for dynamically estimating a thread's progress and also scheduling mechanism is exposed that achieves fairness for throughput, and also describes ways for implementing fairness-aware scheduling at different time scales and both in software and Hardware.

Jia Rao et. al. [13] proposed a technique which overcomes the existing issues to enforce fairness among VMs with different vCPUs running on multiple machines. To improve fairness and efficiency among VM Scheduling, they have proposed a new vCPU scheduling scheme, named FLEX, which improves fairness and efficiency among parallel applications. Flex is designed in two sectors, FlexW and FlexS. FlexW is for vCPU weighted adjustments among CPUs to achieve VM-level fairness. FlexS is for flexibility scheduling vCPUs to decreases and to eliminate busy waiting time which is wasted. FlexW calculates individual VMs fair share based on their weights. FlexW adjusts VM weights to compensate the differences, if there are any differences between actual allocated CPU time and fair share. FlexS Schedule vCPUs with high priority work and busy waiting vCPU s are de-scheduled. Hence the synchronization efficiency is maximized by eliminating busy waiting vCPUs. The proposed system 'Flex' outperforms Xen by 10 times for parallel application using busy waiting time or blocking synchronization.

Jiayin Wang et. al. [14] has implemented an approach called "FRESH". When processing the batch jobs the default Hadoop has predefined slots per node for map tasks and reduce tasks. This fixed number of slots leads to underutilization of resources and extensions of execution time. To eliminate this, they have proposed a new approach named "FRESH" to achieve fair and efficient slot configuration and scheduling of Hadoop clusters. FRESH is designed as (i) static and dynamic slot configuration (ii) map/reduce tasks assigned to available tasks. Static slot configuration is done before launching the Hadoop cluster. And dynamic slot configuration is done after each slot finishes its task. This will improve the makespan and also the fairness without impacting fairness. The experimental results shows the improvement of FRESH is about 31.32% of makespan compared to Fair scheduler and fairness value is >0.95 in all test cases.

Jisha S Manjaly et. al. [15] implemented a new scheduler named which is Task Tracker aware scheduler. The Task Tracker aware scheduler implementation is based on the default FAIR scheduler with some modification in which it provides efficient scheduling. In the correspondence to the current status of the task tracker the jobs are been scheduled. The Task Tracker sends heartbeat periodically to the Job Tracker with its status. The status includes list of running tasks, the number of slots on the nodes and few other information. The job tracker sends the status information to the task tracker aware scheduler. According to the Task Tracker availability and job in the queue to the job scheduler the jobs are scheduled, only if the task present in the task object matches with the task tracker configuration. This Modified algorithm helps in efficient scheduling with user flexibility.

ZujieRen et. al. [16] has provided a huge observation on comprehensive workload analysis with Hadoop cluster at Taobao. This observation gives researchers to understand the production environment. Using these observation they have designed a workload generator ANKUS, which synthesizes e-commerce style MapReduce workload at low cost using job modeling, job clustering and job mixture. ANKUS takes lot of information about the job. The information consists of job submission time, allocated memory for each task, bytes and records read/written by each task and map/reduce tasks count. Also a new scheduler algorithm, FAIR4S, has been implemented, which is designed to be subjective towards small jobs. FAIR4S algorithm prioritizes the job based on priority.

The techniques used to address the weakness are, setting slots quota for pools, setting slot quota for individual users, Assigning slots based on pool weight and extending job priorities.

Bogdan Ghit et. al. [17] explains that running multiple instances of the MapReduce framework concurrently in a multicluster system or datacenter enables data failure and version isolation. It achieves some form of performance isolation, but achieving this in the time-varying workloads which are submitted to the MapReduce instances, a dynamic resource allocation is required for those instances. It is a challenging target for many organizations with large-scale data processing infrastructure. Here, FAWKES is presented, which is a mechanism to balance the allocations of multiple MapReduce instances so that they experience similar service levels. FAWKES is based on the MR-cluster, a new abstraction used for positioning the MapReduce instances on physical resources. Here, it is assumed that the data locality is achieved. In this paper, experimental approach is taken to give a provision for multiple MR-clusters in a datacenter or multicluster system.

## 4. DeadLine-Aware Scheduling

Shen li et. al. [18] discuss on improving the workflow deadline satisfaction ratios over Hadoop clusters. In the Hadoop, single master node monitors the entire clusters, and also it is responsible for scheduling all tasks. The master node receives lots of task assignment request per second. With these constraints, master node may not be able to carry out complex runtime workflow analysis. It leaves little storage space for workflow related profiles because of its monitoring and scheduling responsibilities. WOHA is designed, which is a scalable framework which concentrates on meeting workflow deadlines. WOHA gives complex workflow analysis to clients during the submission of workflow and it also shifts costly job processing to slave nodes after the initialization of job. The client node with the workflow configuration files queries for minimal cluster status information from the master node. Then, the client generates a scheduling plan locally, which the master node can use to prioritize jobs among multiple concurrent workflows. WOHA adds negligible overhead to the master node.

Jorda polo et. al. [19] proposes a scheduling technique for multi-job map reduce workloads that are able to dynamically build performance models of the executing work load. Here, the task scheduling approach enable map reduce runtimes to dynamically allocate resources in a cluster of machines which is based on the observation of progress achieved by the various jobs and the time of completion with each job. Here, a different approach which estimates the completion time of a job during its run time dynamically. The main goal of the schedules is to minimize the distance to the dead line of the jobs. By dynamically calculating the priority of each job, we can estimate the completion time and performance goals.

## 5. Distributed Scheduling

Wenzhi Cui et. al. [20] proposes DiFS (Distributed Flow Scheduling) which is based adaptive routing for hierarchical data center networks is a switch-only routing protocol that is executed independently on the control unit of each switch DIFS aims at balancing flows among different links and it is also improves bandwidth utilization for data center networks DIFS does not require centralized control and DIFS does not allow flow splitting and hence it limits packet reordering. DIFS achieved land balancing by working into directions. First, each switch uses the path allocation algorithm that assigns flows evenly to all out going links which avoid local flow collisions. Second, each switch also monitors its incoming links by running the imbalance defection algorithm. If a collision is detected, the switch will send an Explicit Adoption Request (EAR) message which suggests the switch that is sending to change its path. After receiving the EAR, the sending switch will run explicit adaption algorithm to avoid remote flow collisions. Here, we can also see that the load balance among core switches is not enough to achieve load balance among different links. DIFS solves this problem using the control messages called Explicit Adoption Requests.

## 6. Job Scheduling based on shared input policy

In Hadoop application the input data from large volume are well handled by Hadoop. Input data are divided into small splits and through streaming data are accessed. However, data placement when assigning tasks to reduce phases is not considered well. The intermediate data generated by Hadoop is stored in temporary buffer. Data is spilled to disk when these buffers are full. This increases the waiting time and decreases the application performance. Aprigio Bezerra et. al. [21] have proposed and implemented on extension of shared input policy by adding a RAMDISK for temporary intermediate data storage between hard disk and buffer created by Hadoop. Hadoop spills intermediate data from buffers to RAMDISK when buffer reaches their threshold limit. He proposed policy which uses RAMDISK for handling spilled data improved average makespan time by 48.4%.

Yintian Wang et. al. [22] has proposed an enhancement of Round robin scheduler. The MapReduce job scheduling algorithm is one of the core technologies of Hadoop. The default job scheduling algorithm in Hadoop is FIFO; this scheduler will affect the jobs which are submitted later. The paper uses the round robin with multiple feedbacks scheduling algorithm to eliminate the problem. For scheduling, job with higher priority is been chosen by the Job tracker and the jobs are processed in round robin fashion. The jobs with lower priority are moved to the next queue, so that these jobs get more slots for processing the data. Thus helps in making scheduler fairer. The average response of the scheduler is reduced by 10% - 50% when compared to FIFO and fair scheduler. The throughput of the round robin with multiple feedbacks is also increased.

## 7. Rule based scheduling

Chun-Wei Tsai et. al. [23] implements a novel heuristics scheduling algorithm named hyper heuristics scheduling algorithm (HHSA). Rule based scheduling algorithm has been extensively used in many cloud computing system, because they are simple and easy to implement. Single heuristic algorithm has combined two or more heuristics algorithm for better scheduling. The proposed HHSA algorithm for scheduling cloud computing systems to reduce makespan has two levels of detection operations, one for diversity detection and another for improvement detection to eliminate the timing to consider the low level heuristic algorithm. The low level heuristic (LLH) algorithm is used to select one of the heuristic algorithms from the pool. The LLH algorithm does the selection and acceptance of heuristic algorithm. It algorithm decides whether the selected algorithm will be continuously used or not. This LLH phase is an important phase in hyper heuristic algorithm. When the selected heuristic algorithm gets stuck with a solution for certain iterations, LLH determines the selection timing which selects a new heuristic algorithm for later iterations. At the processing phase of hyper heuristic algorithm, it does transition, evaluation and determination. This characteristic of hyper heuristic algorithm determines the high search diversity which increases the chance to find the better solutions at further iterations without affecting computation time.

**8. Size based Scheduling**

Mario Pastorelli et. al. [24] have proposed size based scheduling with job aging. This Scheduling schema has an efficient approach to ensure shorter mean response time, which provides better response time. The aging module receipts an input job size according to the estimates, and outputs a priority for every active job, which is used by the job scheduler. The job aging helps in reduces starvation, attain fairness and it needs minimal computational load which will improve the total job completion time i.e., make span. They proved that a size-based discipline such as HFSP performs very well, and that particular job size information is not crucial for the scheduler to function properly. The experimental observation compared HFSP to the widely used default FAIR scheduler. The observations indicated that both interactivity and efficiency requirements were largely met and both small and large jobs do not starve. The job break time distribution is consistently in favor of HFSP. The paper has met the significant needs i.e., HFSP is simple to configure, and allows resource pools to be collective because workload diversity is essentially accounted by the size-based scheduling.

**9. Budget driven Scheduling**

Alex D. Breslow et. al. [25] proposes Persistent Online Precise Pricing Agent (POPPA) framework. The objective is to provide accurate performance interference estimates for parallel applications with fewer overloads. POPPA consists of a controller and a series of Execution managers. Execution manager is responsible for the execution of parallel application. The execution Manager launches the selected job and attaches a Performance Monitoring Context (PMC) to the job. PMC monitors the job performance b reading and evaluating appropriate Hardware performance counter. During execution, the Execution Manager updates and reports the current status and performance data of the job to the controller. Controller responsibility is to conduct shuttering, a mechanism to measure and quantify the performance interference among the running applications. After the execution of the job has completed, the pricer thread analyses the raw performance data logged by the controller and quantifies the performance interference and degradation.

Yang Wang et. al. [26] addresses the problem of scheduling a batch of MapReduce jobs as a workflow within budget and deadline is considered. A task-level Scheduling is addressed where the scheduled unit is a task and to solve the fixed-budget problem, an efficient in-stage greedy algorithm is designed which computes the minimum execution time with a given budget for each stage. Dynamic programming (DP) algorithm is developed to achieve a global optimal solution. To improve efficiency, greedy algorithms, Global Greedy Budget (GGB) and Global Refinement (GR) are developed. In GGB in-stage greedy algorithm is extended and in GR DP-based algorithm to distribute exponentially reduced budget in the workflow is run iteratively.

**10. Profile-based scheduling**

Zhenning Wang et. al. [27] discuss on CAP Scheduling. Nowadays, Heterogeneous systems with CPU and GPU are becoming popular. It is important to take advantage of using all the systems to perform a single task using data parallelism. Data Parallelism scheduling strategies does not use GPU's performance characteristics. Thus, it is too much overhead or it does not produce accurate results. Here CAP, a profile-based scheduling strategy to solve the problem by advancing for the performance characteristics of the GPU. In CAP scheduling, first the threads are generated using pthread to handle all the devices and assign devices these threads. In the scheduling part, only one thread can execute, while other threads waiting for the scheduling thread to finish its work. Here, the main thread of the program is used as the scheduling thread. The scheduler distributes the workloads to each thread then threads can do their work. If GPU-assigned thread is received a work which needs partial data. When doing partial computation, it will just transfer the needed data. Otherwise it loads all data into GPU memory before the first execution. By using this technique, we can reduce the data transfer time between GPU and CPU. Results show that comparing with existing strategies, CAP can achieve up to 45.1% performance improvement by estimating the performance of the GPU accurately.

**11. Dynamic Scheduling**

He ma et. al. [28] explains about DASH which is designed to provide high quality of media content over the internet which is delivered from HTTP web Servers. The visual content is divided into number of a number of different bitrates, which helps the MPEG DASH client to automatically select the next segment which is to be downloaded and to play back depending on the current network conditions. In order to support the live distribution of media events and to make the user satisfied the overall processing delay should be kept minimum. Here, a dynamic scheduling algorithm for video transcoding jobs is designed to support DASH over HTTP in a cloud environment. First video transcoding time (VTT) is estimated based on the statistics, from duration of the video segment and the targeted bit rate. Then, the scheduler dynamically distributes video transcoding jobs VTJ according to the VTT and system workload. The dynamic scheduler algorithm, dynamically serve requests and adjust the video transcoding mode VTM accordingly. It also distributes urgent jobs to the fastest processor and lessens the transcoding completion time.

**12. Other Scheduling**

B. Thirumala Rao et. al. [29] discuss on Apache Hadoop which was designed mainly for running large batch of jobs such as Web indexing and Log mining. Users submitted jobs to a queue, and the cluster process them in the order of jobs submitted. As and when data becomes larger, they developed more computations they wanted to run; another use case was sharing a MapReduce cluster between multiple users. The benefits of sharing is high, costs becomes less because sharing a cluster is of less cost than building a Separate Hadoop cluster for each group. The Scheduler in Hadoop introduced two Schedulers for multi-user workloads: The Fair Scheduler developed at Facebook, and the Capacity Scheduler, developed at Yahoo. The Fair Scheduler worked with Facebook's need to share its data warehouse between multiple users. Fair Scheduler gives a fair share of the cluster capacity over time. Users assign jobs to pools, with each pool allotted a minimum number of Map and Reduce slots. If free slots are available in idle pools, it may be allocated to other pools, and if there is an excess capacity in pool, it is shared among jobs. Capacity Scheduler developed at Yahoo address a scenario where the number of users is large, and there is a need to ensure a fair allocation of computation resources among users. The capacity scheduler allocates jobs based on the submitting users to queues with number of Map and Reduce slots. Queues that contain jobs are given their configured capacity, while free capacity in queue is shared among other queues. Within a queue, scheduling operates on a modified priority setting allocated to that user and class of job. When a Task tracker slot becomes free, the queue with the lowest load is chosen, from which the oldest remaining job is chosen. A task is then scheduled from that job. Overall, this has the effect of enforcing cluster capacity sharing among users; whereas fair scheduler was sharing among jobs.

Linh.T.X.Phan et. al.[30] discuss about MapReduce Scheduling in which there are two tasks, Map Task and Reduce Task, each of which is a single unit that is performed in a parallel at the two phases i.e., Map phase and Reduce phase. Job Scheduling in Hadoop is performed by a master node, which in turn distributes the work to a number of slave nodes, each slave corresponds to one physical machine, and has a number of predefined map/reduce slots for executing Map and Reduce tasks. Scheduling of MapReduce jobs in Hadoop proceeds as follows. First, slave nodes inform the master about the availability of free slots. The master will assign the pending task to a free slot based on a scheduling policy. The Hadoop implementation includes two typical scheduling policies. The first is a FIFO Policy, in which the scheduling maintains a FIFO waiting queue sorted by arrival time. Whenever a slot becomes available, the master simply selects a task from the next job on the waiting queue for execution, while FIFO is easy to implement, it does not prioritize the MapReduce jobs, or ensure that all the jobs get equal opportunities for execution. As an alternative to FIFO, Fair scheduling policy ensures that all submitted MapReduce jobs will get an equal opportunity for executing their tasks.

## III. CONCLUSION

The MapReduce framework along with the overview of different scheduling techniques with their applications is discussed in this paper. Different scheduling techniques to enhance the data locality, makespan, efficiency, fairness and performance are discussed with special emphasis on its real time applications. However, the scheduling technique is an open area for research.

## IV. ACKNOWLEDGEMENT

The work reported in this paper is supported by the college through the TECHNICAL EDUCATION QUALITY IMPROVEMENT PROGRAMME [TEQIP-II] of the MHRD, Government of India.

## REFERENCES

[1] EnginArslan, MrigankShekhar, TevfikKosar, "Locality and network-aware reduce task scheduling for data-Intensive applications", 5th ACM International Workshop on Data-Intensive Computing in the Clouds, pp 17-24, Nov 2014, DOI: 10.1109/DataCloud.2014.10.

[2] Xiao Ling, Shadi Ibrahim, Song Wu, Hai Jin, "Spatial Locality Aware Disk Scheduling in Virtualized Environment", IEEE Transactions on Parallel and Distributed Systems, Volume 1, Issue 99, September 2014, pp 1, DOI : 10.1109/TPDS.2014.2355210.

[3] Jongse Park, Daewoo Lee, Bokyeong Kim, Jaehyuk Huh, Seungryoul Maeng, "Locality-aware dynamic VM reconfiguration on MapReduce clouds", 21st International ACM Symposium on High-Performance Parallel and Distributed Computing, pp 27-36, June 2012, DOI: 10.1145/2287076.2287082.

[4] Aprigio Bezerra, Porfidio Hernandez, Antonio Espinosa, Juan Carlos Moure, "Job Scheduling for Optimizing Data Locality in Hadoop Clusters", 20th European MPI Users' Group Meeting, Sept 2013, pp 271-276, ISBN: 978-1-4503-1903-4, DOI: 10.1145/2488551.2488591.

[5] Xiangping Bu, Jia Rao, Cheng-Zhong Xu, "Interference and locality-aware task scheduling for MapReduce applications in virtual clusters", 22nd international symposium on High-performance parallel and distributed computing, June 2013, pp 227-238, ISBN: 978-1-4503-1910-2, DOI:10.1145/2462902.2462904.

[6] Ruiqi Sun, Jie Yang, Zhan Gao, Zhiqiang He, "A virtual machine based task scheduling approach to improving data locality for virtualized Hadoop", IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS), 2014, pp 297-302, DOI: 10.1109/ICIS.2014.6912150.

[7] Saba Sehrish, Grant Mackey, Pengju Shang, Jun Wang, John Bent, "Supporting HPC Analytics Applications with Access Patterns Using Data Restricting and Data Centric Scheduling Techniques in MapReduce", IEEE Transactions on Parallel and Distributed Systems, Vol. 24, No. 1, pp 158-169, January 2013, DOI: 10.1109/TPDS.2012.88.

[8] Xite Wang, Derong Shen, Ge yu, Tiezhng Nie, Yue Kou, "A Throughput Driven Task Scheduler for Improving MapReduce Performance in Job Intensive Environments", IEEE International Congress on Big Data, pp 211-218, July 2013, DOI: 10.1109/BigData.Congress.2013.36.

[9] Weina Wang, Kai Zhu, Lei Ying, Jian Tan, Li Zhang, “Map task scheduling in MapReduce with data locality: Throughput and heavy-traffic optimality”, IEEE/ACM Transactions on Networking, Volume 1, Issue: 99, pp 1-14, November 2014, DOI: 10.1109/TNET.2014.2362745.
[10] Yi Yao, Jiayin Wang, Bo Sheng, Jason Lin, Ningfang Mi, "HaSTE: Hadoop YARN Scheduling Based on Task-Dependency and Resource-Demand", IEEE 7th International Conference on Cloud Computing, pp 184-191, June 27-July 2 2014, DOI: 10.1109/CLOUD.2014.34.
[11] Fahad R. Dogar, Thomas Karagiannis, Hitesh Ballani, Antony Rowstron, “Decentralized task-aware acheduling for data center networks”, ACM conference on SIGCOMM, 2014, pp 431-442, ISBN: 978-1-4503-2836-4, DOI:10.1145/2619239.2626322.
[12] Kenzo Van Craeynest, Shoaib Akram, Wim Heirman, Aamer Jaleel, Lieven Eeckhout, “Fairness-aware scheduling on single-ISA heterogeneous multi-cores”, 22nd International Conference on Parallel Architectures and Compilation Techniques, Edinburgh, Sept 2013, pp 177-187, DOI: 10.1109/PACT.2013.6618815.
[13] Jia Rao, XiaoboZhuo, “Towards fair and efficient SMP virtual machine scheduling”, 19th ACM SIGPLAN symposium on Principles and practice of parallel programming, Volume 49, Issue 8, pp 273-286, August 2014, DOI: 10.1145/2692916.2555246.
[14] Jiayin Wang, Yi Yao, Ying Mao, Bo Sheng, Ningfang Mi, “FRESH: Fair and Efficient Slot Configuration and Scheduling for Hadoop Clusters”, IEEE 7th International Conference on Cloud Computing, Anchorage, AK, June 27 - July 2 2014, pp 761-768, DOI: 10.1109/CLOUD.2014.106.
[15] Jisha S Manjaly, Varghese S Chooralil, “Task Tracker Aware Scheduling for Hadoop MapReduce”, 3rd International Conference on Advances in Computing and Communications, Cochin, 29-31 Aug 2013, pp 278-281, DOI: 10.1109/ICACC.2013.103.
[16] Zujie Ren, Jian Wan, Weisong Shi, Xianghua Xu, Min Zhou, “Workload Analysis, Implications, and Optimization on a Production Hadoop Cluster: A Case Study on Taobao”, IEEE Transactions on Services Computing, Volume 7, Issue 2, pp 307-321, June 2014, DOI: 10.1109/TSC.2013.40.
[17] Bogdan Ghit, Nezih Yigitbasi, Alexandru Iosup, Dick Epema, ”Balanced Resource Allocations Across Multiple Dynamic MapReduce Clusters”, ACM international conference on Measurement and modeling of computer systems, pp 329-341, ISBN: 978-1-4503-2789-3, DOI:10.1145/2591971.2591998.
[18] Shen Li, Shaohan Hu, Shiguang Wang, Lu Su, Abdelzaher T, Gupta I, Pace R, ”WOHA: Deadline-Aware Map-Reduce Workflow Scheduling Framework over Hadoop Clusters”, IEEE 34th International Conference on Distributed Computing Systems, Madrid, June 30-July 3 2014, pp 93-103, DOI:10.1109/ICDCS.2014.18.
[19] Jorda Polo, Yolanda Becerra, David Carrera, Malgorzata Steinder, Ian Whalley, Jordi Torres, Eduard Ayguade, “Deadline-Based MapReduce Workload Management”, IEEE Transactions on Network and Service Management, Volume 10, Issue 2, Jan 2013, pp 231-244, DOI: 10.1109/TNSM. 2012.122112.110163.
[20] Wenzhi Cui, Chen Qian, ”DiFS: Distributed Flow Scheduling for Adaptive Routing in Hierarchical Data Center Networks”, ACM/IEEE symposium on Architectures for networking and communication systems, Oct 2014, pp 53-64, ISBN: 978-1-4503-2839-5, DOI: 10.1145/2658260.2658266.
[21]Aprigio Bezerra, Porfidio Hernandez, Antonio Espinosa, Juan Carlos Moure, ”Job Scheduling in Hadoop with shared Input policy and RAMDISK”, IEEE International Conference on Cluster Computing, Madrid, pp 355-363, Sept 2014, DOI:10./1109/CLUSTER.2014.6968788.
[22] Yintian Wang, Ruonan Rao, Yinglin Wang, “A Round Robin with multiple feedback job Scheduler in Hadoop”, IEEE International Conference on Progress in Informatics and Computing, Shanghai, pp 471-475, 16-18 May 2014, DOI: 10.1109/PIC.2014.6972380.
[23] Chun-Wei Tsai, Wei-Cheng Huang, Meng-Hsiu Chiang, Ming-Chao Chiang, Chu-Sing Yang, “A Hyper-Heuristic Scheduling Algorithm for Cloud”, IEEE Transactions on Cloud Computing, Volume 2, Number 2, pp 236-250, June 2014, DOI: 10.1109/TCC.2014.2315797.
[24] Mario Pastorelli, Antonio Barbuzzi ,Damiano Carra, Matteo Dell Amico, Pietro Michiardi, “HFSP: Size-based Scheduling for Hadoop”, IEEE International Conference on Big Data, Silicon Valley, CA, 6-9 Oct 2013, pp 51-59, DOI:10.1109/BigData.2013.6691554.
[25] Alex D. Breslow, Ananta Tiwari, Martin Schulz, Laura Carrington, Lingjia Tang, Jason Mars, “Enabling Fair Pricing on HPC Systems with Node Sharing”, ACM International Conference on High Performance Computing, Networking, Storage and Analysis, 2013, Article No. 37, ISBN: 978-1-4503-2378-9, DOI: 10.1145/2503210.2503256.
[26] Yang Wang, Wei Shi, “Budget-Driven Scheduling Algorithms for Batches of MapReduce Jobs in Heterogeneous Clouds”, IEEE Transactions on Cloud Computing, Volume 2, Issue 3, 2014, pp 306-319, DOI: 10.1109/TCC.2014.2316812.
[27] Zhenning Wang, Long Zheng, Quan Chen, Minyi Guo, “CAP: Co-Scheduling Based on Asymptotic Profiling in CPU+GPU Hybrid Systems”, International Workshop on Programming Models and Applications for Multicores and Manycores, pp 107-114 ISBN: 978-1-4503-1908-9, DOI: 10.1145/2442992.2443004.
[28] He Ma, Beomjoo Seo, Roger Zimmermann, “Dynamic Scheduling on Video Transcoding for MPEG DASH in the Cloud Environment”, 5th ACM Multimedia Systems Conference, 2014, pp 283-294, ISBN: 978-1-4503-2705-3, DOI: 10.1145/2557642.2557656.
[29] B Thirumala Rao, L S S Reddy, “Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments”, International Journal of Computer Applications, Volume 34, No. 9, Nov 2011.
[30] Linh T X Phan, Zhuoyao Zhang, Boon Thau Loo, Insup Lee, ”Real-time MapReduce Scheduling”, University of Pennsylvania Department of Computer and Information Science Technical, pp 1-6, Report No. MS-CIS-10-32, 2010.