BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **16**, No 3
Sofia • 2016 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2016-0033

A Survey on Job Scheduling in Big Data

M. Senthilkumar[1], *P. Ilango*[2]

[1]*School of Information Technology and Engineering, VIT University, Vellore, India*
[2]*School of Computer Science and Engineering, VIT University, Vellore, India*
Emails: mosenkum@gmail.com, dr.p.ilango@gmail.com

Abstract: *Big Data Applications with Scheduling becomes an active research area in last three years. The Hadoop framework becomes very popular and most used frameworks in a distributed data processing. Hadoop is also open source software that allows the user to effectively utilize the hardware. Various scheduling algorithms of the MapReduce model using Hadoop vary with design and behavior, and are used for handling many issues like data locality, awareness with resource, energy and time. This paper gives the outline of job scheduling, classification of the scheduler, and comparison of different existing algorithms with advantages, drawbacks, limitations. In this paper, we discussed various tools and frameworks used for monitoring and the ways to improve the performance in MapReduce. This paper helps the beginners and researchers in understanding the scheduling mechanisms used in Big Data.*

Keywords: *Big Data, Hadoop, MapReduce, classification, HDFS* (*Hadoop Distributed File System*), *scheduler.*

1. Introduction

Big Data plays very important role in many industries like healthcare, automobiles, IT etc. Effective utilization of energy, resource, time becomes challenging task nowadays. Big Data has become more popular in IT sector, banking, finance, healthcare online purchasing, engineering and many other areas. Big Data refers wide range of datasets and hence it's difficult to manage by existing applications. The data sets are very complex and growing day by day in humongous volume. Raw data are continuously generated from social media, online transactions, etc. Due to continuous increase in volume, velocity and variety complexity increases; it induces lots of difficulties and challenges in data processing. Big Data becomes a complex process in terms of correctness, transform, match, relates, etc.

The main aim of scheduling in Big Data processing is to plan the processing and completion of as many tasks as possible by handling and altering data in a proficient way with a minimum number of changes. Different methods are preferred for the allocation of resources, which have special architectural characteristics. Finding the best scheduling method for a particular data processing and request leftovers is an

important challenge. The Big Data processing as a large "batch" process that runs on an HPC cluster by dividing a job into smaller tasks and distributing the work to the cluster nodes. The Big Data processing models must be aware of data locality when deciding to shift data to the computing nodes or to create new computing nodes near data locations. Job scheduling is to allocate resources to jobs fairly. Large and small jobs will be practically assigned to each node by analyzing the realistic situation of resource utilization through the static and dynamic priority scheduling in MapReduce clusters. Moreover, the job scheduling can predict the resource utilization of the jobs which have not been performed by analyzing the performed jobs.

Big Data poses many research challenges like analyzing of Big Data, handling the data volume, data privacy and security, storage, visualization, job scheduling, fault tolerance and energy optimization. Analysis of Big Data becomes very difficult due to the incomplete and heterogeneous nature of the data produced. The collected data are available in different formats, structure and variety. Dynamic scheduling of jobs with distributed computing demands even scheduling of resources across various geographical areas [1, 7].

In grid base architecture, workflow generation, resource management and scheduling are the major concerns. The architecture does not require any human intervention for task execution. It provides flexibility and complexity gets reduced in workflow generation of tasks, mainly it saves time and cost [9]. Grid scheduling becomes very essential part to aggregate the power of distributed resources and it provides a non-trivial solution to the user [12]. Resource scheduling and application scheduling play major role in grid computing, scheduling, evaluation done by simulation approaches and real time environment [27].

Real time scheduling with multiprocessor is proposed with different scheduling methods and performance metrics used for comparing the scheduling methods. Global, hybrid scheduling algorithm and various approaches for sharing and usage in real time scheduling are discussed in [10]. To make decisions better Big Data need to convert the data to an interactive format. In data visualization different forms are used to represent the same data, which resolves the problems like perception based issues and limited screen based issues [2]. Choosing the Big Data tools based on understanding the requirements and data analytics is a very complex process, selecting the tools and result analysis plays an important role. Tools are designed to help the people to perform various tasks [3]. YARN is not following push based scheduling and hence it reduces the major problem of lowering job latency. Facebook uses Corona, in Corona job tracker and client runs at the same time, Facebook invested large sum of money in Hadoop and made many changes in Hadoop distribution. Corona performs time optimization for Facebook and applications [4]. MapReduce model is predicting the completion time of the map tasks based on demand from CPU and disk by calculating MVA (Mean Value Analysis). MVA introduced an analytical model with multiple classes of jobs and performs comparison of a single node and multi node in Hadoop environments [5].

Hadoop^{++} is used to improve the performance of the Hadoop framework without changing the environment. Hadoop injects the technology in the right place through user defined functions. Hadoop^{++} is very useful in tasks like indexing and join

processing [11]. MapReduce model with iterative processing is introduced in HaLoop. HaLoop improves their efficiency by adding a loop aware scheduling and cache mechanisms in Big Data Applications with iterative processing. HaLoop extends the new novel programming model with many optimization methods like loop aware scheduler, loop invariant in data caching and caching for efficient verification with fix point [6]. iMapReduce is a user friendly environment and allows the user to specify the iterative computation with MapReduce function. Performance is getting improved in iterative implementations due to the reduction of overhead, removal of static data shuffled and asynchronous tasks allowed in MapReduce. iMapReduce delivers five times faster performance when compared to Hadoop [38]. The hybrid data base system Llama is introduced, columns are defined into correlation groups and it will provide support for the tables to vertically partition. MapReduce with the query engine supported by Llama also provides a new join algorithm for fast join processing. The experiment is conducted with EC2 and shows that excellent load and query performance is achieved, and MapReduce follows row wise storage [26]. Manimal automatically analyzes and applies suitable data aware optimizations in MapReduce program, so that no additional work is required from the developer. It also detects optimization opportunities, and hence the speed is increased [19]. SkewTune is introduced for user defined MapReduce programs. The advantage of SkewTune is that it just extends Hadoop and reduces job run time. Many applications become very effective by using SkewTune [22]. The Starfish is used for the self-tuning system, however cost based optimizing for MapReduce programs are complex in Starfish. From Starfish the user must be able to find out MapReduce program's behavior during execution of tasks; the ways of program's behavior change when changing parameters like resources and input data; it also optimizes the program effectively [18]. MapReduce workflow needs comprehensive plan space for generating the workflows. For that purpose Stubby is introduced as a cost based optimizer, which is capable of searching the selective subspace through full plan space. Stubby enumerates the plans and transfer the plan to efficient search algorithm [27]. Twister supports MapReduce iterative operations and performs computations effectively and efficiently [14]. Data analytics need key requirement scalability, and MapReduce becomes popular because of its salient features like fault tolerance, scalability, flexibility and ease of programming. Improving the performance of MapReduce with classification is based on the specific problem [13].

FIFO algorithm is used for default scheduling in MapReduce and jobs are which enters first into the ready queue grabs first priority [28]. This scheduling gives disadvantages for some jobs which enter the queue later, and thus leads to the starvation of jobs. Fair scheduler is designed to run small jobs quickly and thus overcomes the drawback of FIFO scheduler. Fair scheduling assures the service for all jobs [15]. Delay algorithm is introduced in Facebook, by applying some changes in MapReduce with existing data locality; the performance is improved and the lowest response time for map tasks is achieved [16]. Cost base optimizer for complex MapReduce programs is introduced for optimization of large space configuration parameters of the program. The profiler is used for collecting statistical details from unmodified MapReduce programs; it also performs fine grained cost estimation [17].

Hadoop uses round-robin scheduling algorithm; when the smaller weight jobs are more and the larger weight jobs are less, the round-robin scheduling is used to overcome the problem. The scheduler puts forward weight and updates rules based on the situations. Task allocation becomes very effective by using this idea and only job tracker is used in the Hadoop platform [34]. Hadoop scheduler causes performance degradation in heterogeneous environments; to overcome the problem LATE (Longest Approximate Time to End) is introduced and it is highly robust to the heterogeneous environment. Hadoop response time is improved by using LATE scheduler [40]. Delay scheduling is used in Hadoop to improve data locality in an optimal way and throughput increased for different kinds of tasks. Fairness in job is achieved by using this delay scheduling due to simplicity and policies for fair sharing between the resources [39]. Hadoop and Dryad are introduced for efficient scheduling and enhanced utilization in a shared environment. Scheduling with MapReduce holds the issues, synchronization overhead and locality. Scheduling multiple jobs with fairness, locality improvement and delay scheduling are achieved in Hadoop and Quincy scheduler [38, 41]. Deadlines are very important requirement considered by cloud based data processing platform such as Hadoop. By default Hadoop uses FIFO scheduler with priority based option. Scheduling jobs with deadline constraints are given by the user and ensure that deadlines are met properly, so that estimated schedule is executed correctly [20]. COSHH (Classification and Optimization based Scheduler for Heterogeneous Hadoop) scheduler is a combined FIFO and fair sharing algorithm which supports hybrid solution [29]. ARIA is proposed for Service Level Objective (SLO) with three components. SLO based scheduler determines the job ordering and amount of resources used for meeting the job deadlines in Hadoop. SLO's job objectives are effectively implemented by the scheduler until job demands exceeds the Hadoop cluster resources [33].

Spark streaming is one of the open source frameworks for more reliable, high-throughput and low latency stream processing. It is a near real time stream processing framework processing commodity hardware, so real time event processing is not assured in its scheduling system. Profiling outcome indicates that the total delay time of events with unstable inputs is more unstable and presents big fluctuations. Effective scheduling approach reduces the worst case event processing time by dynamically adjusting the time window of batch intervals. It is a real time enhancement to Spark Streaming based on Spark's framework [24]. Spark is a category of efficient Big Data processing platform based on memory and alike to Hadoop MapReduce. But the Spark default task scheduling plan does not take the different capacity of node into account for heterogeneous Spark cluster, thus leading to lower the system performance. An adaptive task scheduling strategy for the heterogeneous Spark cluster, which analyzes parameters from surveillance to dynamically adjust the task allocation, weights nodes through monitoring the load and resource utilization of nodes. Experimental results validate that this strategy for heterogeneous nodes is superior to the default task scheduling strategy in aspects like task completion time, nodes working state and resource utilization [36].

The structure of this paper is as follows. Section 2 presents Hadoop MapReduce architecture, features, working principles and issues in Hadoop. Section 3 is about

the Hadoop job schedulers. Section 4 discusses MapReduce optimization techniques. Finally, Section 5 gives the conclusion.

2. Hadoop MapReduce: Architecture, features, working principles and issues

Hadoop Distributed File System (HDFS) is storing huge files in a distributed manner in various HDFS nodes. The data are divided into 64MB /128 MB blocks. HDFS maintains three copies of every block in unique machines. HDFS maintain the metadata in name node and data are kept in separated nodes are called as data nodes. In this section we discuss the architecture of MapReduce, scheduling of job, Hadoop features and issues.

2.1. Hadoop MapReduce architecture

Hadoop includes MapReduce, a distributed data processing model that runs on large clusters of machines. A Hadoop MapReduce job mainly has two user-defined functions, map and reduce function. The Hadoop job takes the input of key-value pairs (k, v) and map function is called for each of these pairs. Job tracker and task tracker have two types of nodes that control the execution process. The job tracker coordinates and schedules all tasks to run on task trackers. Task trackers in turn send progress reports to the job tracker. If a task fails, the job tracker can reschedule it on a different task tracker. The essence of Hadoop MapReduce is that the users just define map and reduce functions. The Hadoop framework takes care of everything and Hadoop MapReduce uses the Hadoop Distributed file system to perform I/O performance.

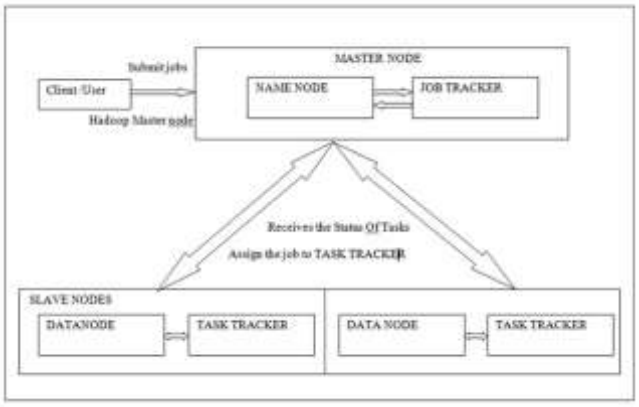

Fig. 1. Hadoop MapReduce architecture

2.2. Hadoop features

i. Hadoop has the power to add new nodes without need to change the clusters (data). Hadoop cluster can accommodate more than one node without any difficulty.

ii. Since Hadoop is used to store large TB's of data it's more affordable when compared to any other servers.

iii. Hadoop doesn't follow the norms for structuring data, hence it is flexible to work. Another important feature is that multiple data can be combined in Hadoop. No matter is the data structured or not, Hadoop can perform map and reduce jobs very effectively.

iv. Hadoop can redirect the data into another location when there is a fault in the given primary node. When one of the nodes in a cluster fails, the job can be redirected to other node and hence Hadoop is highly a fault tolerant system.

v. Since Hadoop does parallel computing, system is more effective and efficient in terms of deriving the results.

vi. Hadoop offers a large cluster of local servers to store large amount of data.

2.3. Issues

Managing Hadoop cluster with multiple MapReduce tasks on multiple nodes needs effective and efficient scheduling policies to achieve the performance and resource utilization. Performance is affected by some issues like energy, fairness, data locality and synchronization.

2.3.1. Energy

In Big Data applications large scale of data operations are held out by data centers with MapReduce. Hadoop requires large amounts of energy in processing the data within data center and energy becomes problematic. Overall cost of energy increases in the data center, so minimizing the energy consumption becomes a big challenge in data centers.

2.3.2. Fairness

The resources are shared among the users and fair measures are required in scheduling the all jobs without starvation. MapReduce with heavy workload uses the entire resource in terms of the cluster. Jobs with short computation may not have desired response time. Workload must be fairly shared or distributed among the jobs in the cluster. Fairness deals with locality and MapReduce phases. Performance degradation of throughput and response time are reduced if the MapReduce jobs are equally shared and the input files are distributed in clusters.

2.3.3. Data locality

The distance between the data node which holds the input and the task node is called locality. The data transfer rate depends on the distance between the input data and computation node, if the input data node is very near then data transfer rate becomes low. Many times Hadoop cluster is not able to achieve locality; in such attempts rack locality is achieved.

2.3.4. Synchronization

The process of transferring intermediate outputs of the mapping process to the input of reduce process is called synchronization. This is an important issue in MapReduce

of scheduling jobs. After the completion of map phase only reduce phase will take place and reduce tasks are fully dependent on the map task. If any one of the map nodes becomes slow, then the entire process performance becomes slow. In a heterogeneous environment synchronization becomes a problem; each node in the Hadoop cluster has unique computation facility, hardware and bandwidth and this leads to decreased performance of Hadoop cluster.

3. Hadoop job schedulers

3.1. Classification of schedulers used in Big Data

Classification of scheduler is based on various parameters like priority, time, strategy, environment, energy, resource awareness (CPU, IO, disk, memory, and free slots). Classification of scheduling in Hadoop is achieved based on using available resources effective and efficient, scheduling strategy and time. This classification of different scheduler is shown in Fig. 2.

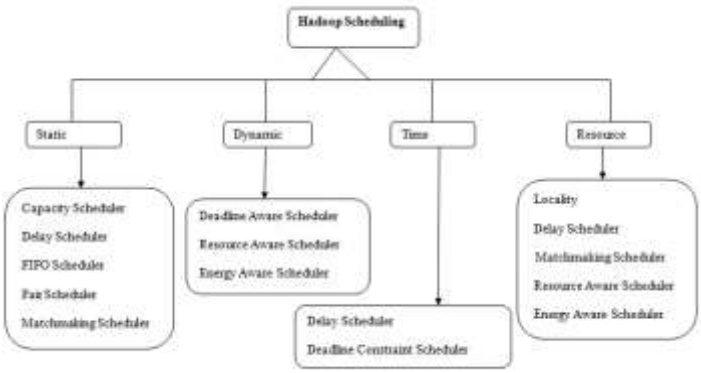

Fig. 2. Classification Hadoop scheduler

3.1.1. Static scheduling strategy

In static scheduling strategy to job allocation in processors, it is achieved before the program to begin the job execution time, in completion time. Processing resources are known only at the compile time. The objective of static strategy is to minimize execution time of processors. In static scheduling Capacity scheduler, Delay scheduler, FIFO scheduler, Fair scheduler, and Matchmaking scheduler are available.

3.1.2. Dynamic scheduling strategy

In dynamic scheduling strategy job allocation is performed during execution time. The resources have little knowledge about jobs. The environment is unknown for the jobs, but jobs will execute during their lifespan. Whenever a job is executed, a decision is made and dynamic environment is applied to the processors. Deadline Aware scheduler, Resource aware scheduler and Energy aware scheduler are used in dynamic scheduling strategy.

3.1.3. Based on time

In this scheme scheduling of the job is based on time and job scheduling is achieved on the base of deadline provided by the cluster. Every job must complete the job within the specified time and deadline is given by the user to check whether the executed job is within the deadline or not. Delay scheduler and Deadline constraint scheduler are used in time based scheduling.

3.1.4. Based on available resource

Scheduling is based on the requirements of the resources. In this scheme, job performance and resource utilization is improved, various resources and their usage are also compared. Resources are memory, disk storage, CPU time, IO, etc. Delay scheduler, Matchmaking scheduler, Resource aware scheduler and Energy aware scheduler are used in resource based scheduling.

3.2. Description of schedulers used in Big Data

Hadoop jobs are sharing the resources with scheduling policy based on the scheduling mechanisms when and where jobs have to be executed. The main objective of scheduling is to maximize the throughput, minimize the completion time; overhead and available resources must be balanced parallel by allocating jobs to processors. Schedulers used in Hadoop are:

3.2.1. Default FIFO scheduler

Hadoop uses default scheduling policy FIFO; this scheduling prefers the earlier submitted jobs over later submitted jobs. Whenever the new job arrives, job tracker is initialized. It places the job in the queue. When running multiple jobs with priority, not supported jobs are strict with data locality and completion of previous job, only next jobs are allowed and the job must wait for completion of previous task. FIFO scheduler is mostly used when the user feels that the execution order of jobs has no importance.

3.2.2. Fair scheduler

Fair scheduler is introduced by Facebook and the core idea of this scheduler is to assign the resources to every job, so that every job will get equally shared resources. By using pools in Fair Scheduler the jobs are grouped and perform fairness in sharing between pools. Job configuration properties are controlled by pools. Every user has his/her own pool and minimum resources shared with assigned pool. By default, all pools must contain equal sharing of resources with MapReduce task.

3.2.3. Capacity scheduler

Capacity scheduler is produced by Yahoo; this scheduler is used when multiple organizations want to share the large cluster with minimum capacity and sharing excess capacity among the users. Instead of pools several queues are created; each queue is configurable with MapReduce slots. The queue has priority with FIFO and

is achieved in capacity scheduler. Jobs high priority must access resources, compared to the less priority jobs. Scheduled tasks are understood by memory consumption of each task; the scheduler has ability to control and allocate memory based on the physical available resources. Based on the organization needs resources are shared, available resources are partitioned with MapReduce cluster among many organizations.

3.2.4. Delay scheduler

This approach is used by Facebook and locality is achieved by waiting approach in the Hadoop cluster. In this scheme, if the data for the task is not present, then a task tracker waits for a particular amount of time. If there is any request from the local node for task assigning, then the scheduler will check the size of the job; if the job is too short, then the scheduler will skip that job and look for any subsequent jobs available to run. If this job, skipping persists for long time duration, then it will launch nonlocal task to avoid starvation. This scheduler resolves the locality problem like sticky slot and head of line scheduling. Fairness with locality is achieved by this scheduler and strict jobs order relaxes with task assignment.

3.2.5. Matchmaking scheduler

Data localities of map tasks are enhanced by the matchmaking scheduling. Before assigning non local tasks, scheduler ensures that every local task grabs a slave node to assign job first. Scheduler keeps on trying to find matches with a slave node with some input data for unassigned tasks. A locality marker will mark the node and guarantees that each node gets chance to grab the tasks. This scheduler leads to highest data locality and less response time for tasks (map). Unlike FIFO, this scheduler relaxes strict job order for task assignment.

3.2.6. LATE (Longest Approximate Time to End) scheduler

Tentative tasks will progress very slowly due to various issues like CPU load, background process running slowly, resource contention and large number of background process. It will detect slow running task in the cluster and launch equivalent task as a backup; this process is called speculative execution of tasks. The main goal is to minimize a job response time as much as possible. When the user is running short jobs, answers must be very quick and response time is important. By default, this scheduler supports homogeneous clusters and is highly robust to heterogeneity. It is mainly used for optimizing the performance of jobs, but it doesn't provide the guarantees of reliability.

3.2.7. Deadline constraint scheduler

The user specifies deadline constraints while scheduling the jobs, ensures that jobs met deadline and are scheduled for execution. This scheduler improves system utilization dealing with deadline constraint and data processing. It is achieved by cost model for job execution and Hadoop scheduler with constraint. Cost model with job execution considers various parameters like MapReduce task with runtime, the input

size of data and data distribution. A Hadoop scheduler with user constraints has deadline as part of the input; when the job is submitted for testing, it checks whether the job can be finished within the time specified by the deadline or not.

3.2.8. Resource aware scheduler

In Hadoop, resource utilization is minimized by this scheduler. The schedulers like Fair scheduler, FIFO, Capacity scheduler the admin must assign job to a queue and manually makes sure that specific resources are shared. These schemes focus on how effectively resource utilization has been done with various types of utilization like IO utilization, memory utilization, disk utilization, CPU utilization, network utilization. This scheduler makes use of the two metrics: "dynamic free slot advertisement" and "free slot filtering/priorities". Instead of having a fixed number of computation slots configured with each task tracker node, the free slots are dynamically used according to the demand by Dynamic Free Slot Advertisement. In "free slot filtering/priorities" maximum number of slots is fixed per node. Free slots are identified by task tracker by adverting based on the resource availability.

Table 1. Comparison various schedulers used in Big Data with Hadoop

Scheduler	Job allocation	Environment		Priority in job queue	Resources sharing	Features	Drawbacks
		Homogeneous	Heterogeneous				
FIFO	Static	✓	✗	No	No	Easy to implement efficient	Data locality job starvation
Fair	Static	✓	✗	Yes	Yes	Fast response time mixing small with large job possible	Configuration problems unbalanced performance
Capacity	Static	✓	✗	No	Yes	Potential to reuse unused jobs in the queue	Complexity choosing scheduler
Delay	Static	✓	✗	Yes	No	Simple No overhead	No effective. Slots are limited
Match-making	Static	✓	✗	Yes	Yes	High data locality. Utilization level is high	--
LATE	Static	✓	✓	Yes	Yes	Heterogeneity more robust	Reliability
Deadline constraint	Dynamic	✓	✓	Yes	Yes	Supports optimization	Node must uniform
Resource aware	Dynamic	✓	✓	Yes	Yes	Performance resource utilization	Monitoring bottlenecks
Energy aware	Dynamic	✓	✓	Yes	Yes	Energy optimized	Multiple MapReduce jobs

From Table 1 is visible that Capacity scheduler and Fairness scheduler are used for resolving fairness issues in short jobs and production jobs. The

data locality issue is solved by Matchmaking Scheduler and high utilization of resources is achieved. Any real time work is designed with deadline constraint scheduler. Job deadline is provided by the user and job completion is achieved in real time cluster. Hadoop uses static job allocation policy and the number of Map/Reduce slots is fixed. Hadoop cluster with resource aware cluster provides cluster utilization, minimizing resource consumption. Job scheduling algorithm is an important research direction in Big Data processing.

4. MapReduce optimization techniques

4.1. Iterative processing

MapReduce framework is not supporting iterative processing. HaLoop is designed for users who want to perform iterative operations like caching, looping, incremental iterations and recursive queries in MapReduce model. HDFS is used for storing each input and output data. HaLoop made changes to existing MapReduce environment: i) Hadoop has interface to the user, master node contains the new loop control module; ii) Hadoop works with data locality for new task scheduler and caches; Hadoop indices application data on slave nodes and invariant data is cached, so doesn't need not be reloaded. Twister is another technique used to achieve the iterative processing. After reduce stage is completed, Twister adds an extra phase called "combine stage". A daemon process is initiated on each node for managing locally running MapReduce tasks with status, they communicate with other nodes. In this model the data are read from the local disks and intermediate data is handled by worker node with distributed memory. Repeated instantiation of workers are avoided by a twister. It also supports static / dynamic variable with configurable MapReduce tasks. A worker node with intermediate data improves the performance of the cluster. Twister is not supporting the drawback loop control.

iMapReduce supports iterative processing. iMapReduce has achieved: i) to avoid repeated task scheduling, persistent map/reduce tasks is used; ii) only one time the input data are loaded into local file system, synchronization barrier is controlled using iMapReduce by providing asynchronous execution. During iterative processing the MapReduce tasks stay alive until the process gets completed. iMapReduce is implemented ot the base of Hadoop and Online prototype. Jobs can be terminated during processing by one of the two ways: defining a fixed number of iterations or bounding the distance between consecutive iteration. MapReduce online has problems like shuffling with intermediate result limit, pipeline processing and frequent check pointing. To overcome the problems some modifications are done in MapReduce Online: the mappers push data temporally to reducers in the same MapReduce job, there is pipelining support for continuous queries. This was not possible earlier in MapReduce and is achieved now.

4.2. Join operations

The join condition is used after map and reduce tasks are completed. MapReduce is processing single input, but joins need two inputs with MapReduce, which leads to problems. The limitation of MapReduce joining multiple dataset into one task needs additional processing to perform join operations. This model enables heterogeneous datasets with processing data multiple times. Map function is transformed with a key/value pair of the input to intermediate key/value pair. The difference between the MapReduce model with map join reduce model is the production of key/value lists of the reduce function. Merge function needs the input datasets to be organized by keys and keys are passed to function to be merged. Map-Join-Reduce model is used to perform aggregation in Big Data. After standard Map and Reduce one new operation "Join" is performed and multiple data sets are joined with aggregation in the framework. The user specifies join *O* function to perform joins between the data sets. During runtime the system automatically perform joins with multiple datasets based on the join order. Too many shuffling is achieved in this model by shuffling each intermediate result of key/value pair to many joiners in the same time. In this model two kinds of processes follow: map/reduce tasks and joiners to invoke reduce tasks.

4.3. Data access

Hadoop^{++} improves the performance of query processing by User Defined Functions (UDF) added to Hadoop which becomes Hadoop^{++}. It implements indexing functionality called "trojan indexes". The trojan indexes are added with HDFS input and splits at load time. Trojan join new technique is introduced for data partitioning for map tasks with join operations. Llama scheme uses column-wise format called CFile and suppors multiway join in a single MapReduce job. Llama is implemented with column wise storage for MapReduce in column wise format known as CFile. Data is split into vertical group and stored in HDFS with particular column. Grouping and partitioning of data enables selective access of the column.

4.4. Load balancing

Load imbalance takes place during map or reduce phase. The imbalance in MapReduce is known as skew. Skew arises when there is a huge difference in size and cost of the task. Two types of skew introduce input data with uneven distribution, some data parts take longer time to process. Skew dynamically divides large partition into smaller partitions during the runtime. Skew contains three phases: detect, scan and plan.

4.5. Short job optimization

During data analysis, processing of data takes much time to complete. MANIMAL is used for automatic optimization of single MapReduce job. The analyzer tests the MapReduce program code before execution without using any runtime details. In data analysis some rules lead to creating of pre-computed B^+ tree index.

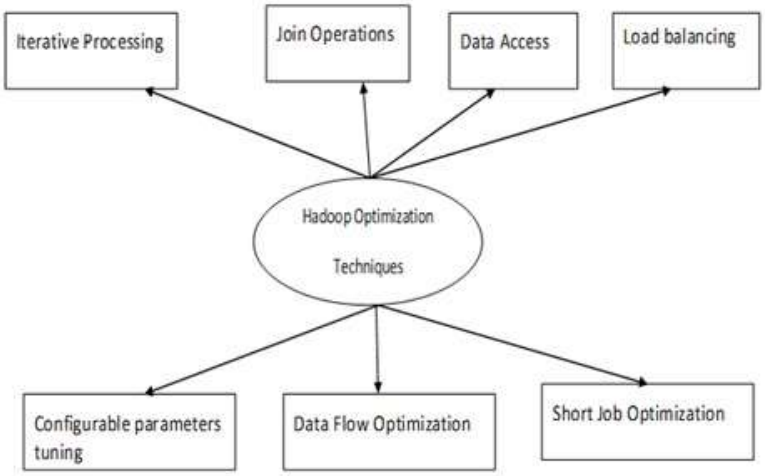

Fig. 3. Hadoop optimization techniques

4.6. Data flow optimization

Stubby is introduced in MapReduce for job workflow to perform cost based optimizing. In large dataset complexity of dataset gets multiplied with the increase in data in MapReduce jobs. Stubby searches for optimization in the subspace with a full plan of workflow and identifies the opportunity to improve the performance. An annotated MapReduce workflow is given as an input for stubby and produces optimized plan as an output.

4.7. Configurable parameters tuning

Starfish is introduced for cost based optimization and determining values for configurable parameters of MapReduce jobs. Starfish improves the performance of MapReduce with configurable parameters by tuning, but this job becomes tedious. Profiler is introduced to estimate the size of the processed data, the amount of resources usage and execution of each job. What-if engine is introduced to estimate the advantages from tuning the configuration parameter using simulation. The cost based optimizer is used to find the potential settings to be configured and makes sure that good performance is achieved.

Table 2. Comparison of MapReduce optimization techniques

Techniques	Application	Features	Drawbacks
EMRSA I & II	TeraSort, Page rank, and K-means clustering	Energy optimized nearly 40%	Multiple MapReduce jobs
Hadoop++	Relational query like projections and joins	MapReduce interface remains same. Runtime improved without using a local dbms	Fault tolerance is very less
HaLoop	Data mining, web ranking	Loop are of task scheduler is available. Checking termination supports the extra devoted jobs	Not providing support abstractions
iMapReduce	Data mining, web ranking, online social network analysis, graph analysis	Static graph shuffling. Avoid shuffling of static data between tasks	Optimization limited
Llama	High level workload management, Data warehousing	Good load performance Fair data locality	Overheads in CFile
Manimal	Static code analysis data centric MapReduce programs	Optimizations without code change, semantic compression for reducing IO	Rule based optimization. Not performing cost based optimization with profiling
Map-Join-Reduce	Query processing, processing *N* way operations	Join multiple datasets	Join is not optimal
MapReduce online	Event monitoring stream processing	Pipelining of intermediate data	Lacking in cache data
SkewTune	Query optimization in web search, page ranking	No need to input minimizes the side effects	Very slow and performance degrade
Starfish	Query optimization with job profiling and optimization	Finds automatically good configuration	No support for logical decisions
Stubby	Log analysis, reporting, business analytics, information processing with retrieval	Cost based optimization automated	Transforma-tion not supported
Twister	Graph search, matrix multiplication, page ranking, dimension reduction	Long running, avoiding unnecessary data to be read	Need large dataset with multiple files

5. Conclusion

Hadoop with job scheduling plays a vital role to achieve the performance in Big Data. In this paper, we discussed various issues in scheduling related to locality, fairness, performance, throughput, load balancing, etc. To overcome the scheduling issues many job scheduling algorithms are presented: FIFO scheduler, Fair scheduler, Delay scheduler, Capacity scheduler. The advantages and disadvantages of respective algorithms are discussed. Various tools are preferred in node allocation, load balancing and optimization of jobs. Comparative study of different tools along with their merits and demerits are discussed. Various optimization techniques are used to efficiently utilize the resources within the constraints of time, energy and memory. Comparison of all those optimization techniques was discussed.

References

1. Abraham, A., R. Buyya, B. Nath. Nature's Heuristics for Scheduling Jobs on Computational Grids. – In: Proc. of IEEE International Conference on Advanced Computing and Communications, 2000, pp. 1-8.
2. Ada, R., R. Kaur. – International Journal of Advanced Research in ComputerScience and SoftwareEngineering. – Ijarcsse, Vol. **3**, 2013, No 3, pp. 665-668.
3. Assunção, M. D., R. N. Calheiros, S. Bianchi, M. A. S. Netto, R. Buyya. Big Data Computing and Clouds: Trends and Future Directions. – Journal of Parallel and Distributed Computing, 2015, 79-80, pp. 3-15.
4. Bardhan, S., D. Menasc. The Anatomy of Mapreduce Jobs, Scheduling, and Performance Challenges. – In: Proc. of Computer Measurement Group, 2013.
5. Bardhan, S., D. Menascé. Queuing Network Models to Predict the Completion Time of the Map Phase of MapReduce Jobs, 2012.
6. Bu, Y., B. Howe, M. D. Ernst. HaLoop : Efficient Iterative Data Processing on Large Clusters. – Proceedings of the VLDB Endowment, Vol. **3**, 2010, No 1-2, pp. 285-296.
7. Casavant, T. L., J. G. Kuhl. A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. – IEEE Transactions on Software Engineering, Vol. **14**, 1988, No 2, pp. 141-154.
8. Coulouris, G., J. Dollimore, T. Kindberg. Distributed Systems: Concepts and Design. – Computer, Vol. **4**, 2012.
9. Dabhi, V. K., H. B. Prajapati. Soft Computing Based Intelligent Grid Architecture. – In: Proc. of International Conference on Computer and Communication Engineering (ICCCE'08), Global Links for Human Development, 13-15 May 2008, pp. 574-577.
10. Davis, R. I., A. Burns. A Survey of Hard Real-Time Scheduling for Multiprocessor Systems. – ACM Computing Surveys, Vol. **43**, 2011, No 4, pp. 1-44.
11. Dittrich, J., J.-A. Quiané-Ruiz, A. Jindal, Y. Kargin, V. Setty, J. Schad. Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without it Even Noticing). – Proceedings of the VLDB Endowment, Vol. **3**, 2010, No 1-2, pp. 515-529.
12. Dong, F., S. G. Akl. Scheduling Algorithms for Grid Computing : State of the Art and Open Problems. – Components, Vol. **202**, 2006, No 4, pp. 1-55.
13. Doulkeridis, C., K. Nørvåg. A Survey of Large-Scale Analytical Query Processing in MapReduce. – VLDB Journal, Vol. **23**, 2014, No 3, pp. 355-380.
14. Ekanayake, J., H. Li, B. Zhang, T. Gunarathne, S. Bae, J. Qiu, G. Fox. Twister : A Runtime for Iterative MapReduce. – In: Proc. of 19th ACM International Symposium on High Performance Distributed Computing, HPDC'10, 2010, pp. 810-818.
15. Hadoop Fair Scheduler Design Document, 2010, pp. 1-11.

16. He, C., Y. Lu, D. Swanson. Matchmaking: A New MapReduce Scheduling Technique. – In: Proc. of 3rd IEEE International Conference on Cloud Computing Technology and Science, CloudCom'2011, 2011, pp. 40-47.
17. Herodotou, H., S. Babu. Profiling, What-if Analysis, and Cost-Based Optimization of MapReduce Programs. – PVLDB: Proceedings of the VLDB Endowment, Vol. **4**, 2011, No 11, pp. 1111-1122.
18. Herodotou, H., F. Dong, S. Babu. Mapreduce Programming and Costbased Optimization? Crossing this Chasm with Starfish. – Proceedings of the VLDB Endowment, Vol. **4**, 2011, No 12, pp. 1446-1449.
19. Jahani, E., M. J. Cafarella, C. Ré. Automatic Optimization for MapReduce Programs. – Proceedings of the VLDB Endowment, Vol. **4**, 2011, No 6, pp. 385-396.
20. Gautam, J. V., H. B. Prajapati, V. K. Dabhi, S. Chaudhary. A Survey on Job Scheduling Algorithms in Big Data Processing. – In: Proc. of IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT'15), Coimbatore, 2015, pp. 1-11.
21. Kc, K., K. Anyanwu. Scheduling Hadoop Jobs to Meet Deadlines. – In: Proc. of 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom'10), 2010, pp. 388-392.
22. Kwon, Y., M. Balazinska, B. Howe, J. Rolia. SkewTune: Mitigating Skew in Mapreduce Applications. – In: Proc. of 2012 ACM SIGMOD International Conference on Management of Data, 2012, pp. 25-36.
23. Lee, K. H., Y. J. Lee, H. Choi, Y. D. Chung, B. Moon. Parallel Data Processing with MapReduce: A Survey. – SIGMOD Record, Vol. **40**, 2011, No 4, pp. 11-20.
24. Liao, Xinyi, et al. An Enforcement of Real Time Scheduling in Spark Streaming. – In: Proc. of Green Computing Conference and Sustainable Computing Conference IEEE, 2015, pp. 1-6.
25. Lim, H., H. Herodotou, S. Babu. Stubby: A Transformation-Based Optimizer for MapReduce Workflows. – Proceedings of the VLDB Endowment, Vol. **5**, 2012, No 11, pp. 1196-1207.
26. Lin, Y., D. Agrawal, C. Chen, B. C. Ooi, S. Wu. Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. – In: Proc. of 2011 International Conference on Management of Data (SIGMOD'11), 2011, pp. 961-972.
27. Prajapati, H. B., V. A. Shah. Scheduling in Grid Computing Environment. – In: Proc. of 4th International Conference on Advanced Computing & Communication Technologies, 2014, pp. 315-324.
28. Rao, B. T., L. S. S. Reddy. Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments. – International Journal of Computer Applications, Vol. **34**, 2012, No 9, pp. 29-33.
29. Rasooli, A., D. G. Down. A Hybrid Scheduling Approach for Scalable Heterogeneous Hadoop Systems. – In: Proc. of 2012 SC Companion: High Performance Computing, Networking Storage and Analysis (SCC'2012), 2012, pp. 1284-1291.
30. Sakr, S., A. Liu, A. G. Fayoumi. The Family of MapReduce and Large-Scale Data Processing Systems. – ACM Computing Surveys, Vol. **46**, 2013, No 1, pp. 1-44.
31. International Journal of Advanced Research, Vol. **3**, 2013, No 5, pp. 875-878.
32. Suthaharan, S. Big Data Classification: Problems and Challenges in Network Intrusion Prediction with Machine Learning. – Performance Evaluation Review, Vol. **41**, 2014, No 4, pp. 70-73.
33. Verma, A., L. Cherkasova, R. Campbell. ARIA: Automatic Resource Inference and Allocation for MapReduce Environments. – In: Proc. of 8th ACM International Conference on Autonomic Computing, 2011, pp. 235-244.
34. Wang, C. Journal of Computers, Vol. **8**, 2013, No 3.
35. Wolf, J., A. Balmin, D. Rajan, K. Hildrum, R. Khandekar, S. Parekh, R. Vernica. CIRCUMFLEX: A Scheduling Optimizer for MapReduce Workloads with Shared Scans. – ACM SIGOPS Operating Systems Review, Vol. **46**, 2012, No 1.
36. Yang, Zhiw Ei, et al. Adaptive Task Scheduling Strategy for Heterogeneous Spark Cluster. – Computer Engineering, 2016.
37. Yong, M., N. Garegrat, S. Mohan. Towards a Resource Aware Scheduler in hadoop. – In: Proc. of ICWS, 2009, pp. 1-10.

38. Y o o, D., K. M. S i m. A Comparative Review of Job Scheduling for MapReduce. – In: IEEE International Conference on Cloud Computing and Intelligence Systems, 2011, pp. 353-358.
39. Z a h a r i a, M., D. B o r t h a k u r, J. S e n S a r m a, K. E l m e l e e g y, S. S h e n k e r, I. S t o i c a. Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling. – In: Proc. of 5th European Conference on Computer Systems, 2010, pp. 265-278.
40. Z a h a r i a, M., A. K o n w i n s k i, A. J o s e p h, R. K a t z, I. S t o i c a. Improving MapReduce Performance in Heterogeneous Environments. – Osdi, 2008, pp. 29-42.
41. Z h a n g, Y., Q. G a o, L. G a o, C. W a n g. iMapReduce: A Distributed Computing Framework for Iterative Computation. – Journal of Grid Computing, Vol. **10**, 2012, No 1, pp. 47-68.