CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2015)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3578

Context-aware scheduling in MapReduce: a compact review

Muhammad Idris, Shujaat Hussain, Maqbool Ali, Arsen Abdulali, Muhammad Hameed Siddiqi, Byeong Ho Kang and Sungyoung Lee*,†

Ubiquitous Computing Laboratory, Department of Computer Engineering, Kyung Hee University, Seoul, Korea

SUMMARY

It is a fact that the attention of research community in computer science, business executives, and decision makers is drastically drawn by big data. As the volume of data becomes bigger, it needs performance-oriented data-intensive processing frameworks such as MapReduce, which can scale computation on large commodity clusters. Hadoop MapReduce processes data in Hadoop Distributed File System as jobs scheduled according to YARN fair scheduler and capacity scheduler. However, with advancement and dynamic changes in hardware and operating environments, the performance of clusters is greatly affected. Various efforts in literature have been made to address the issues of heterogeneity (i.e., clusters consisting of virtual machines and machines with different hardware), network communication, data locality, better resource utilization, and run-time scheduling. In this paper, we present a survey to discuss various research efforts made so far to improve Hadoop MapReduce scheduling. We classify scheduling algorithms and techniques proposed in the literature so far based on their addressing areas and present a taxonomy. Furthermore, we also discuss various aspects of open issues and challenges in the scheduling of MapReduce to improve its performance. Copyright © 2015 John Wiley & Sons, Ltd.

Received 31 July 2014; Revised 28 May 2015; Accepted 8 June 2015

KEY WORDS: scheduling; task scheduling; job scheduling; data-intensive computing; big data; cloud

1. INTRODUCTION

The cloud vision was first given by McCarthy at MITs centennial celebration in 1961 [1, 2]. According to this vision, 'Computing may someday be organized as a public utility just as the telephone system is a public utility'. It is a group of commodity hardware sharing computation resources to provide application/software, infrastructure, and platform as a service. In the recent few years, cloud computing [1] has transformed large part of information technology industry to make the services more attractive by offering on pay-as-you-go basis. In cloud computing, the end users are unaware of the infrastructure in the back end [1]. Unlike grid or utility computing, cloud computing users have no idea of where the servers are located. The term 'cloud' refers to infrastructure as cloud from where users can access applications on demand. The concept of cloud computing is supported by high data demands as the data are significantly growing on the Web [3]. Cloud resources can easily be scaled up and scaled down dynamically in response to meet the demands of users using virtualization and other distributed system techniques.

In this age of exponentially growing data in various aspects such as velocity, volume, variety, and veracity termed as big data [4, 5], cloud computing is a point of interest for industrial and academic research communities because of its scalable, distributed, and fault-tolerant storage, servers, and applications. Data processing in recent growing scale on cloud is one of the important problem and

*Correspondence to: Sungyoung Lee, Department of Computer Engineering, Kyung Hee University, Seocheon-dong, Giheung-gu Yongin-si, Gyeonggi-do, Korea.
†E-mail: sylee2014@oslab.khu.ac.kr

is therefore focused in research. Hadoop [6] has emerged as a distributed storage framework that supports computational model such as MapReduce on large-scale data on cloud [7]. MapReduce is mainly used for distributed fault-tolerant, scalable, and data-intensive application development. Scheduling of jobs and tasks in such distributed frameworks plays an important role in performance and efficiency of the framework [8]. Hadoop MapReduce's basic scheduling mechanism to efficiently schedule tasks is based on first-in first-out (FIFO) scheduling algorithm.

Hadoop default scheduler is based on FIFO scheduling mechanism. FIFO scheduler maintains a queue where the tasks at the start of the queue are scheduled based on their order of submission. In MapReduce, a job is divided into many tasks and assigned to TaskTrackers (TT) on DataNodes based on availability of free slots. By default, whole Hadoop cluster is used by a single job, and jobs are processed by their turns. Priorities can be assigned to jobs; however, it needs to be manually turned on. The ability to prioritize the tasks, to provide fair share of resources to every user [9], and support capacity scheduling [10] have been added in the newer releases of Hadoop by Facebook and Yahoo, respectively. Fair scheduler provides every user a fair share of cluster resources. It maintains pools with fair number of map and reduce tasks assigned to it. MapReduce jobs are assigned to pools for execution, and pools can also be assigned priorities. Tasks in overloaded pools are killed and assigned to those pools that are still waiting for their capacity. Idle resources of pools are assigned to other pools, and overloaded pool capacity is shared to maintain fairness and harmony. Fair scheduler tracks the actual time of computation and fairly allocates time to assign next tasks in a fair manner. Jobs that require shorter time are allocated sufficient resources to execute quickly while saving long-running jobs from starvation.

Capacity scheduler [10] enforces user-level cluster capacity sharing among users and enabling Hadoop cluster as multi-tenant cluster [11]. It provides security by safeguarding job from other users by allocation of computational resources using its elastic scheduler. The capacity scheduler prioritizes and allocates fair share of resources to users. It allocates jobs to queues with users ability to configure number of map and reduce slots. Queues are prioritized based on the time of job submission and constraints provided by the user. Allocated number of resources are provided based on user configuration, and excess resources of a queue are distributed among other queues. Dynamic resource allocation is performed based on run-time resource usage, state, and context capturing.

Scheduling in Hadoop is context-aware by capturing the context/state of the tasks and scheduling them according to the availability of machines and resources at run-time [12, 13]. Many efforts have been made so far to optimize the scheduling of jobs in Hadoop cluster consisting of homogeneous/heterogeneous resources with Hadoop basic assumptions. D. Yoo *et al.* in [14] presented a review on scheduling techniques and issues faced in MapReduce such as data locality, fairness, heterogeneity, and synchronization. Their work presents a comparative study of these issues and discusses solutions based on their weaknesses and strengths. Their study discusses only a limited number of techniques and does not cover the recent developments; however, much work has been performed in MapReduce scheduling, which will be discussed in detail in this paper. Recent surveys in [15–17] list some of the scheduling techniques such as Longest Approximate Time to End (LATE) [18], Self-adaptive MapReduce (SAMR) [19], and Delay schedulers. However, they lack the property to properly classify various techniques and methods based on their specific topics such as data locality, data replication, cluster environment, and situation awareness. Different scheduling optimization techniques have been proposed to address Hadoop basic assumptions such as data locality, data replication [20], network communication, and fault tolerance. In this paper, we discuss and list down most of the recent available scheduling techniques for MapReduce. We classify various available scheduling techniques based on their features and their focus and then make a taxonomical representation. A comparative chart is presented to identify the features, advantages, and targeting areas of specialty in MapReduce along with discussion. The objectives of the paper include

1. review of existing scheduling techniques in Hadoop MapReduce,
2. classification of MapReduce scheduling techniques, and
3. a comprehensive comparative analysis of the existing scheduling techniques, which will help the researchers to guide themselves towards the future trends of research in this area.

DOI: 10.1002/cpe

The rest of the paper is organized as follows: Section 2 briefly describes Hadoop and MapReduce background. In section 3, different recent schedulers are described. Section 4 discusses schedulers with comparative analysis, and Section 5 concludes the work with future directions.

2. HADOOP AND MAPREDUCE PRELIMINARIES

Hadoop Distributed File System (HDFS) is a system inspired by Google's Google File System [21] storing large files in multiple machines in a shared-nothing mechanism. It is developed to handle big data on clusters of commodity hardware. Apache Hadoop is an open-source implementation of Google's MapReduce [22] and acquired by many large organizations like Google [21], Facebook [23], Yahoo [24], Oracle [25], and Microsoft [26] for enabling their applications on cloud. It provides an abstract and easy programming model to write parallel programs instead of worrying about the data distribution, parallelization, fault tolerance, and computations. MapReduce is a parallel data-processing framework on top of HDFS (provides high throughput and access).

Hadoop has the best fault-tolerant, high-throughput, and server-failure survival mechanisms [27]. Like Google File System, Hadoop also maintain replicas of its data splits across different machines to provide data locality and reliability. Default chunk size of HDFS is 64 MB, and these chunks are once write-multiple-read chunks. Hadoop's master-slave mechanism is shown in Figure 1. It consists of NameNode and DataNodes. A NameNode also called MasterNode, is responsible for controlling the whole MapReduce job through a JobTracker and controlling tasks in a job through a TT while working as DataNode (e.g., single node cluster). A DataNode, also called SlaveNode, consists of DataNode and TT. A SecondaryNameNode in large clusters is used to generate snaps of NameNode to avoid loss and works as standby NameNode. NameNode stores meta-data about the files/data chunks stored in DataNodes, while DataNodes store the actual data chunks (64 MB). By default, each data chunk is replicated by a factor of 3.

MapReduce is a parallel processing framework designed originally by Google to process large-scale distributed data in the cloud [7]. MapReduce is an efficient approach for large-scale data processing. It hides the low-level details of parallel programming (data distribution and scheduling) and provides the user with a high-level abstract implementation. MapReduce works in two phases, that is, map and reduce as shown in Figure 2.

In map phase, the data are mapped as (key, value) pairs, while in reduce phases, map's output is aggregated based on same keys. In between map and reduce phases, aggregation of map output, sorting, and partitioning are performed. MapReduce works in master (JobTracker) and slave (TT) architecture as HDFS's NameNode and DataNode, respectively. The JobTracker is responsible to generate many map and reduce tasks for a submitted job. Each TT on DataNode has a number of

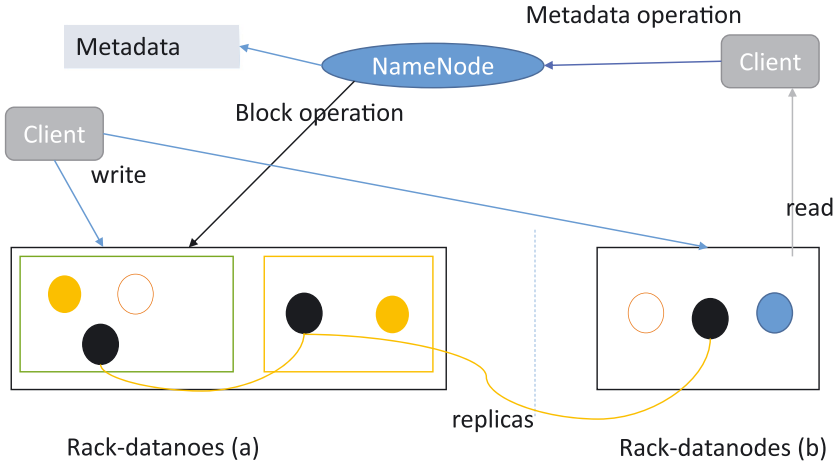


Figure 1. Hadoop architecture: meta-data contains information about the data chunks, replication, and DataNodes. Block operation is issued by the NameNode to write data blocks to the DataNodes. Client issues Metadata operation to specify the data to be read and processed.

DOI: 10.1002/cpe

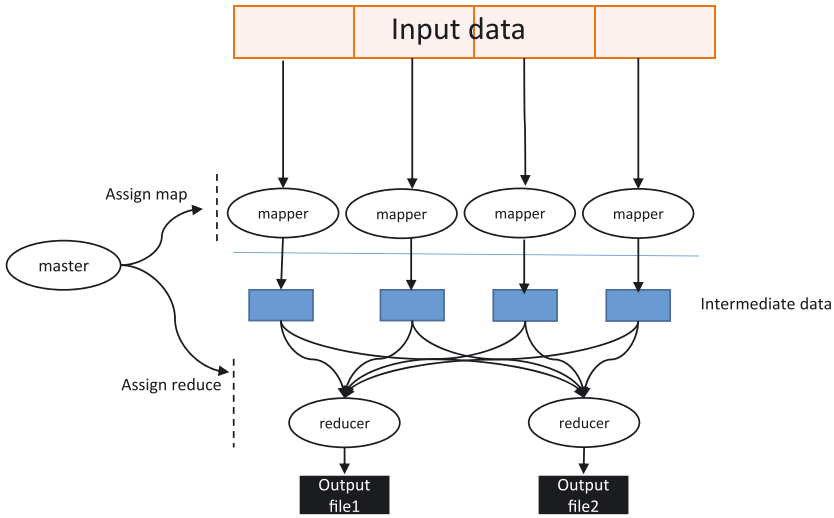


Figure 2. MapReduce architecture: this figure depicts functional work-flow of a typical MapReduce program. Input data are split and processed by mappers, shuffled, and passed to reducers, which collects and generates the final output.

tasks it can process for the reliability and survival from failure of cluster. The scheduling of tasks has gained an immense attention of research community. Many scheduling techniques have been proposed so far, to identify failing tasks, to retrieve the run-time status of tasks and jobs, and to schedule them in an efficient manner to prevent tasks failure. These techniques discuss scheduling in various contexts including data locality in Hadoop, resource sharing, replica awareness, environment awareness, and situation awareness. In the following section, we discuss various scheduling techniques and their targeting context and propose classification taxonomy.

3. CONTEXT-AWARE SCHEDULING IN MAPREDUCE: TECHNIQUES AND TAXONOMY

Context awareness is a property that is defined complementary to various aspects, including location awareness and situation awareness, and is responding to them correctly. In computing, context awareness is a process that involves adaptive interfaces, capturing relevant data in real time, discovery, implicit interactions, and smartness [28]. To provide run-time [29] scheduling and mitigate failure of tasks, context-aware scheduling mechanism is incorporated in MapReduce . In Hadoop MapReduce, scheduling of initial map/reduce tasks, failed map/reduce tasks, waiting tasks, and straggling tasks (Figure 3) is performed based on capturing the context. In traditional MapReduce, an application that needs to run is called a job, and it can be divided into two sets of tasks called map tasks and reduce tasks. Map tasks execute the map function, and reduce tasks execute the reduce function of the job. These tasks are executed on DataNodes and scheduled by TT to process HDFS data.

Figure 3 depicts working of a MapReduce job where a single submitted job is divided into multiple map and reduce tasks. These tasks are executed on distributed DataNodes, and a single straggling task can result in delay of whole job represented as dotted elliptical (computing) shape in the figure. This single task can result in delaying the whole job where other tasks are waiting for execution as shown using double line ellipses in Figure 3. Each MapReduce job has a single JobTracker on NameNode and a TT on each DataNode. Scheduling of map and reduce tasks is performed based on different metrics including data locality on DataNode, resource availability, input of map and reduce tasks, and other factors to avoid long waiting and straggling tasks delaying jobs. Several scheduling techniques, discussed in the following sections, are classified and reformulated into various categories based on their addressed problems as shown in Figure 4.

DOI: 10.1002/cpe

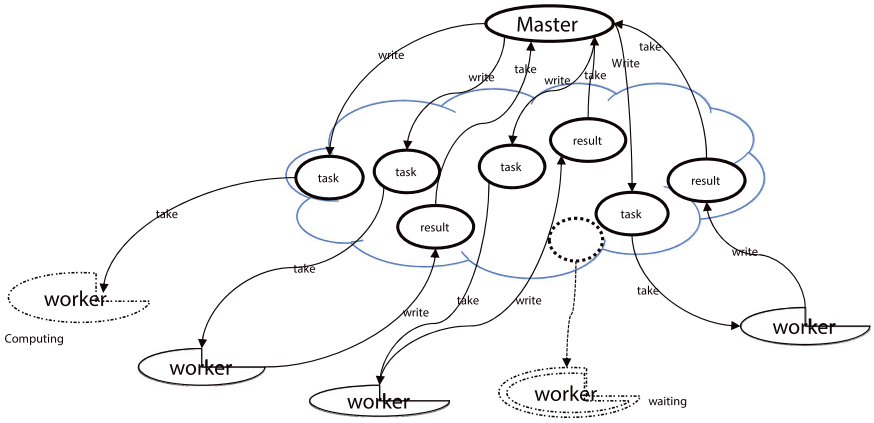


Figure 3. Tasks in a MapReduce Plus job: a computing task (double-dashed) is straggling and delaying the whole job as other workers have returned the result and a worker is waiting for task assignment.

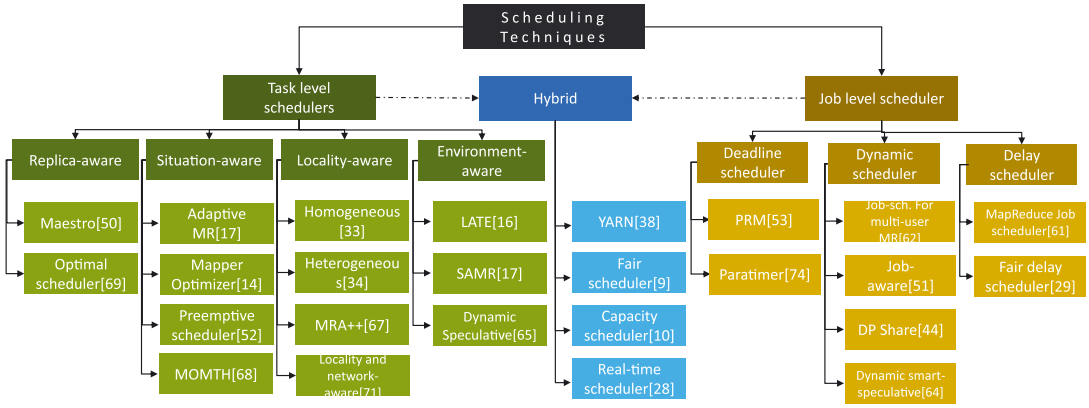


Figure 4. Taxonomy of various scheduler schemes in Hadoop MapReduce: top-level and middle-level nodes represent decisive research criteria, and leaf nodes represent actual scheduling techniques.

Different context-aware scheduling techniques in Hadoop include delay scheduling [30, 31], deadline-based scheduler [32], locality-aware [33, 34] scheduling, resource-aware [35] scheduling, MapReduce scheduling in heterogeneous environment [36], scheduling based on flow-shops [37], and situation-aware scheduling [38]. Hadoop's default scheduler is FIFO scheduler, which maintains jobs in a queue where every task in a job has to wait for its turn. It also supports prioritizing the tasks; however, it needs to be manually turned on.

Hadoop MapReduce version 2.0 or YARN [39] comes up with new architecture in hadoop-0.23, and it has separated the functions of JobTracker at MasterNode into cluster resource management life cycle of the job. In this new architecture, Fair scheduler [40] is used to assign fair share of resources to the applications in the cluster. This part of architecture deals with the resource management part of YARN, and the responsibility is detached from the job life cycle. In general, Fair scheduler only cares about the memory to schedule; however, in YARN, it decides based on both memory and CPU. In YARN, new apps do not have to wait; rather, they are assigned resources to continue processing. Unlike Hadoop default scheduler, when new jobs arrive, it assigns equal resources instead of queuing the jobs. It can also prioritize the jobs. It starts speculative tasks by comparing the progress of tasks to the average progress of the job.

Similarly, YARN uses capacity scheduler [10] to exploit Hadoop cluster as multi-tenant cluster with maximized throughput. Capacity scheduler provides security by safeguarding the job from other users, free resource allocation to other jobs as an elastic scheduler, and schedule based on resources. In a nutshell, capacity scheduler in Hadoop supports cluster sharing among various organizations, which they can support collectively, and it is cost-effective. Multi-tenancy is strongly

DOI: 10.1002/cpe

supported in YARN to fully provide the sharing between various organizations, and stringent limits are imposed in YARN capacity scheduler to avoid any resource occupation by a single or set of jobs. Some efforts have been made to improve and tune Linux schedulers and virtual environment techniques to tune the performance of Hadoop MapReduce.

In an effort to tune Xen scheduling mechanism for MapReduce, K. Hui *et al.* proposed a three-step MapReduce Group Scheduler [41] to guarantee fairness among jobs across cluster. The three-step mechanism includes facilitating MapReduce batch input/output (I/O) over virtual machines to reduce context switches, two-level scheduling policy to reduce blocking events for fairness, and operating it on symmetric multi-processor frameworks. This scheduler modifies the Xen hyper-visor to include these functions, and similarly an adaptive disk I/O scheduling mechanism to improve MapReduce in virtual environment [42]. It proposes a disk pairs' scheduling mechanism in hyper-visor and virtual environment. These proposals address the pure virtually managed environment of the cluster. To exploit MapReduce for high-performance computing (HPC), some techniques and Linux resource management systems have been used such as Simple Linux Utility for Resources Management (SLURM) and MR+.

SLURM is designed to support workload management on various sizes clusters in Linux as an open-source workload manager [43, 44]. It works as an HPC scheduler to support execution of open MPI applications and provides references to implementation of other resource management models. SLURM scheduler is favored over YARN for executing HPC applications specifically because of its key differences in resource utilization design. Being an HPC scheduler, SLURM is designed to share global status information unlike YARN, which uses heartbeats, and it has launching scalability of log *N*, whereas YARN launching scalability is *N* resulting SLURM to start much faster (*N* represents number of nodes in the cluster). Similarly, MapReduce-Plus (MR) project brings Hadoop close to general-purpose HPC clusters [45, 46]. It is an effort to run MapReduce applications on existing HPC MPI frameworks. MR utilizes MPI libraries and tries to overlay the MapReduce *JobClient* and communicates resource queries using sockets as SLURM extension. YARN communication design, that is, heartbeat design, being not suitable for large start-ups, however, has real-time failure and recovery without stopping the jobs. YARN also combines the processing and file-system (HDFS) nodes, and this distinction requires fair failure and recovery model. Thus, SLURM, MR, and other HPC cluster resource managers are more suitable for HPC applications, and the efforts made to combine them with MapReduce to support MR job executions can lead to exciting big data clusters. In this paper, we discuss improvements and optimizations to MapReduce scheduler; therefore, technical details of HPC schedulers are exempted. Some schedulers in distributed frameworks such as Apache Mesos address the distributed resource scheduling as discussed in the succeeding text; however, they do not fall in the category of MapReduce specific scheduling.

Apache Mesos [47] is a distributed system kernel that provides abstraction over storage, processing, and memory of the physical and virtual machines. Mesos is built on the principals of Linux kernel; however, it has a different level of abstraction. Unlike MapReduce 2.0 or YARN, Mesos supports various distributed systems and frameworks such as Spark, Hadoop, Kafka, and elastic search. Mesos is widely deploy-able on data centers for resource management and scheduling between these applications. Apache Mesos does not specifically address the MapReduce scheduling; rather; it addresses the scheduling of distributed systems in a wide data center. Like YARN, it also uses master–slave architecture to manage resources and its management. YARN is purely developed for MapReduce, and it takes care of both jobs in the cluster and tasks in a single job.

In Hadoop, scheduling of tasks in a job is decided based on metrics like input data, data placement, cluster environment(homogeneous and heterogeneous), network communication, intermediate data, and available cluster resources. In literature, a number of scheduling techniques can be found; however, there is no such study that discusses scheduling techniques proposed so far and clearly classifies them into their different categories. In the following subsections, different schedulers with their key features are described.

3.1. Scheduling techniques

In this section, we explain various scheduling techniques proposed for Hadoop optimization in the literature so far. All the techniques are reviewed and explained with respect to the following criteria.

DOI: 10.1002/cpe

3.1.1. Criteria for technical review. The review process in this paper is based on following parameters considered in a distributed environment:

a. Operating environment: Hadoop clusters consist of commodity machines, which may include (1) machines with similar configuration and specification known as homogeneous cluster or (2) those consisting of virtual machines and different configuration [48] (utility computing) such as Amazons Elastic Compute Cloud [49], known as heterogeneous cluster. We review scheduling techniques and classify them based on their targeting environment as homogeneous or heterogeneous.
b. Data locality: Local data execution by a map or reduce task plays an important role in the overall performance of MapReduce job. In large clusters, data locality is crucial because the network bisection bandwidth becomes a bottleneck [7]. Therefore, we identify scheduling techniques that address the problem of data locality in scheduling.
c. Data replication: Hadoop is a distributed environment where data are replicated by a factor of 2 by default and can be changed by the user manually. However, the rise of heterogeneous clusters may result in varied task execution time, task straggling, single node failure, and rack failure. Efficient replication scheme is helpful when a task has no local data, and it needs to access from a remote node. Scheduling techniques are also reviewed and classified based on consideration of data replication scheme.
d. Scheduling scope/context: In Hadoop, scheduling is addressed both at job and task levels. In job-level scheduling, the aim is to support and schedule multiple jobs by different or the same users in a single cluster for better resource utilization and service provisioning. In task scheduling, the aim is to schedule multiple map and reduce tasks in a single MapReduce job efficiently. We identify the scope and context of scheduling schemes and classify them as job-level or task-level scheme.
e. Communication: MapReduce works in two phases: (1) map phase and (2) reduce phase. Data generated by mappers in map phase and collected by reducers in reduce phase is known as intermediate data. The exchange of intermediate data and communication between mappers to mappers, reducers to mappers, and reducers to reducers play an important role in the overall job performance and execution time.

   Based on these criteria, we discuss the scheduling techniques in the succeeding text in detail. Tables at the end of each subsection shows the summary of the technique. Each column shows the addressed issue by the respective scheduler in each row of the table. 'Nil' value shows that the issue has not been addressed and 'Yes' shows that the issue has been addressed in the scheduling technique.

3.1.2. Mapper Optimization scheduler. In Hadoop, assignment of map task to the TT on the DataNode with local data is a key issue of research because of Hadoop's basic theme of moving computation to the data. In [50], Jin *et al.* have proposed a two-step mechanism to optimize the map task assignment strategy in Hadoop in homogeneous environments. Their mechanism includes (1) data locality scheduling and (2) replica selection strategies. In the first strategy, to overcome the limitation of FIFO scheduler, and to schedule a single task at a time, it introduces queues in MapReduce framework. JobTracker can take full advantage of the computation power available by scheduling many map tasks using queuing strategy. In the second strategy, it computes the load of the system for balancing. Load is computed based on some assumption, that is, the cluster is isomorphic (every node has the same specification in the cluster), and the features in TT load computation include (1) operating system overhead referred as A, (2) all map tasks overhead referred as B, and (3) all reduce tasks overhead as C with m number of map and r number of reduce tasks. The TT load is computed as load $= \mathrm{A} + mi * \mathrm{B} + ri * \mathrm{C}$. In this two-step mechanism, the authors show the performance improvement about 2.5 times as compared with Hadoop default FIFO scheduler.

DOI: 10.1002/cpe

Scheduling technique	Operating environment	Data locality	Data replication	Scope
Mapper optimization [50]	Homogeneous	Yes	Yes	Task level

3.1.3. Longest Approximate Time-to-End (LATE) scheduler. LATE scheduler [36] is originally developed for cluster composed of heterogeneous environments where virtualization [51] or different hardware is used. It states that the basic implicit assumptions of Hadoop could be broken in a heterogeneous cluster such as the following: the assumptions of nodes performing roughly at the same rate and constant throughout time could be broken by virtualization because of heterogeneity; and other assumptions also break down such as no cost of launching a task. All these factors lead Hadoop to perform poorly in a nonhomogeneous environment. Many researchers have focused on the need of improving MapReduce performance in heterogeneous environment; they include [52–54].

LATE's basic concept is speculating the tasks that will finish at the end in an environment where all the systems are of different configuration. LATE exploits speculative [55] execution of Hadoop for heterogeneity and proposes a scheduling algorithm on three principals: prioritize the speculated tasks, that is, assign high priority to failed tasks, and considering non-running tasks, prevent these tasks from failing, and run them on fast nodes. To identify a fast node, it uses a basic heuristic of calculating the 'ProgressRate [36]' as

$$\text{ProgressRate} = \text{ProgressScore}/(\text{task's running time}) \tag{1}$$

and then use it for estimating the time to completion (TT):

$$\text{TT} = (1 - \text{ProgressScore})/\text{Progressrate} \tag{2}$$

It maintains a threshold value for classifying a task as slow task called SlowTaskThreshold by comparing the task's progress rate. LATE has estimated much improvement over the default scheduling technique. LATE shows performance improvement by a factor of 2 over default Hadoop scheduler in a 200-node VM Amazons Elastic Compute Cloud cluster.

Scheduling technique	Operating environment	Data locality	Data replication	Scope
LATE [36]	Heterogeneous	Nil	Nil	Task level

3.1.4. Self-adaptive MapReduce (SAMR) scheduler. SAMR scheduling algorithm [19] is proposed as the successor of LATE [36]. It solves the issues of LATE scheduler such as LATE cannot compute the remaining time for a task correctly because it uses only static information and tries to classify the whole node as slow node. SAMR proposes a different heuristic approach to identify the slow tasks and resolve the bottleneck of computation time of under execution task. They maintain historical information of each task, that is, map and reduce, classifying the tasks into map slow tasks and reduce slow tasks, and then identifying map slow tasks and reduce slow tasks. SAMR uses a new heuristic to identify a single task as slow by comparing task's ProgressRate with Average Progress Rate (APR) as PRi $(1 - \text{STaC}) * \text{APR}$ (STaC is SlowTaskCap value ranging from 0 to 1). Similarly, it identifies slow TT as TrRmi $(1 - \text{STrC}) * \text{ATrRm}$ (STrC is SlowTrackerCap ranging from 0 to 1). Based on the slow tasks, it launches backup tasks on free nodes and then waits for the early completion of either backup or original tasks. Backup tasks are launched on different DataNodes to ensure that they will not be slow anymore. SAMR claims improvement over LATE by 14% decrease in execution time and 25% improvement over default Hadoop scheduler.

Scheduling technique	Operating environment	Data locality	Data replication	Scope
SAMR [19]	Heterogeneous	Nil	Nil	Task level

DOI: 10.1002/cpe

3.1.5. Delay scheduler. Delay scheduler [56] is based on giving a fair share of resources to new jobs. It analyzes a fair share (giving fair share of resources to new jobs) algorithm to observe the problems like data locality and head-of-line [57] problem. In Hadoop scheduling, when a task is waiting in queue for launching and reaches to the head and if it has no locality position, then it is launched on a node with no local data based on FIFO [9] algorithm . However, it violates the default assumption [58] of locality in MapReduce. Delay scheduler violates this head-of-line principal and lets the task in the head to wait for its locality by delaying it and launching at a node with data locality. It also prevents the task from starvation.

Delay scheduler uses long task balancing and hotspot replication [59] to minimize the chances of a node to struck in long tasks and concurrent access to a small data file by multiple jobs. In long task balancing, it treats the new jobs as long tasks and marks them as small tasks if they finish quickly. In hotspot replication, if a data file is too small to be replicated on many nodes and multiple jobs try to access this file, then it replicates the file at run-time. Delay scheduler has shown improvement in small jobs by five times faster in multiuser environment and double the throughput in I/O heavy jobs.

Scheduling technique	Operating environment	Data locality	Data replication	Scope
Delay scheduler [56]	Nil	Yes	Yes	Job/task level

3.1.6. Hadoop job to meet deadlines. Kc *et al.* [60] have proposed a deadline-based approach to cater the problem of long-running Hadoop jobs. They have proposed two models to solve this problem. Firstly, they propose a job execution model that considers the metrics like mapper and reducer running time, input data, and data distribution, and based on these metrics, it schedules the job. Secondly, they propose a deadline estimation approach to determine the deadline for a single job based on some assumptions like homogeneous cluster, uniform key distribution, reducers execute after mappers finish, and data already in HDFS. However, they allow these assumption to be broken at a later stage in the problem space.

In the second model, the user is asked to provide maximum execution time as deadline (also called 'Constraint based Hadoop Scheduler' [60]) and then schedules the job accordingly without considering other jobs running in the cluster. It shows that based on deadline, the number of task trackers are assigned to it. It schedules the jobs whose deadlines can be met only. The user submitting the job has to provide the deadline, and the system decides whether the job can be fulfilled in the specified deadline. It tries to provide the ability to execute many number of jobs at an instance of time and in the provided deadline.

Scheduling technique	Operating environment	Data locality	Data replication	Scope
Deadline scheduler [60]	Nil	Nil	Nil	Job level

3.1.7. Dynamic Proportional Share scheduler. Dynamic Proportional (DP) scheduling [61] extends Hadoop default schedulers and provides capacity or quality of service to users based on priorities. Capacity is allotted in map and reduce tasks on per time unit proportion. It allows users to choose which jobs to schedule and prioritize. It gives its users the capability to adjust their allocated resources to meet their job requirements fall and down. To avoid starvation and large job running grip of resources, it supports preemption [62] and guarantees to pay more for more and long resource holding. With no users or no resource allocation of resources, DP scheduler is an imitator enough to behave like fair scheduler.

DOI: 10.1002/cpe

DP scheduler addresses the problems of Hadoop default scheduler that is maintaining separate queues for separate pools, service guarantee over time, and manual priority setting. Sandholm *et al.* [63] propose Virtual Machine (VM)-hosted Hadoop cluster scheduling to solve the issues of dynamically scale-up and scale-down [64] VM instances at run-time based on job resource requirements efficiently. DP and Lottery scheduling [65] assign resources based on holding a lottery and winning the ticket. It is applied on VM resource management; however, in [61], it is applied for the first time in MapReduce scheduling.

Scheduling technique	Operating environment	Data locality	Data replication	Scope
DP scheduler [61]	Nil	Nil	Nil	Job level

3.1.8. Locality-aware scheduling. Data locality is a basic assumption of Hadoop MapReduce, and it critically affects its performance. Hadoop schedules map and reduce tasks to process data on their local systems where their TT reside. Local execution of data mitigates network communication in the cluster, which consequently improves the performance of Hadoop. Hadoop clusters may consist of commodity and virtual machines with the same or different hardware resources. To improve Hadoop data locality in homogeneous (the same virtual machines and hardware configuration) and heterogeneous (varies hardware resources) environments, various scheduling techniques have been proposed. These scheduling techniques are discussed in the following sections.

a. Homogeneous Environment Locality-aware scheduler: Scheduler in [36] also addresses the problem in Hadoop default scheduler assigning the task to a node with data locality. In default scheduler, if no task is found with local data on the requesting node, it assigns the task that is traversed first with no local data. However, the task selection method followed is a probabilistic approach to compute probabilities for each task and is known as next-K-node scheduling. This method sets some rules for probability calculation, and the task with local data available is assigned low probabilities. Tasks with high probability are scheduled to the requesting node. This scheduler also uses a method called node prediction to predict node that will request next for task assignment. It maintains a list of imaginary tasks having equal input sizes and maps their progress to the actual task on the DataNodes. After mapping the tasks, the node with imaginary task on the top in descending order is predicted as the next requesting node. It claims 78% reduction in map tasks with no locality and 77% reduction in network load.
b. Purlieus scheduler: In locality-aware scheduler (Purlieus), Palanisamy *et al.* [35] have proposed a resource allocation system that targets clusters of virtual machines. To reduce network traffic in MapReduce, it proposes improved data locality in map and reduce phases. It has proposed three placement techniques: (1) map-input heavy jobs (map tasks are placed local, while reduce can be placed anywhere), (2) map-and-reduce-input heavy jobs (both map and reduce tasks are placed in set of closely connected machines), and (3) reduce-input heavy jobs (reduce tasks are placed local, while map tasks can be placed anywhere). Purlieus introduces a technique of data placement in cloud for dynamically assigning VMs to the users and avoiding data redundancy and loading time. It also suggests VMs placement technique for a cloud service to provision data locality. Purlieus is purely meant for the cloud service providers who intend to provide Hadoop framework as service. Their results show improvement in execution time by 50% and up to 70% reduction in cross network traffic.
c. Heterogeneous Environment Locality-aware scheduler: Hadoop performance in heterogeneous environment degrades as compared with a homogeneous cluster. To address the problem of heterogeneity and locality-aware scheduler, Zhong *et al.*, in [66], propose a solution to mimic optimal task execution time. It maintains a trade-off between task waiting and handling or transmission time. To fulfill the request of a TT for a task assignment, it assigns

DOI: 10.1002/cpe

the task with data local to DataNode. If no such task exists, it chooses a task with data in a node closest to the requesting node. It reserves the task for node with local data if task's waiting time is less than its transmission time or else schedules the task to the requesting node. Waiting time for a task is computed as the shortest remaining time among the number of tasks executed on the DataNode with data locality.

Scheduling technique	Data locality	Environment	Scope
Homogeneous (a)	Yes	Homogeneous	Task level
Purlieus (b)	Yes	Heterogeneous	Task level
Heterogeneous (c)	Yes	Heterogeneous	Task level

3.1.9. Situation-aware scheduling. Adaptive MapReduce in [67] has proposed an idea of 'situation aware mappers', which violates the MapReduce assumption of independent mappers. In Hadoop default scheduler, DataNode is independent of other DataNodes, and it makes the mappers and reducers independent also. Adaptive MR implements mappers that communicate through distributed meta-data store. It provides efficient write and read operations as transactions and used as communication service. While mappers in adaptive MR communicate for situation awareness, adaptive mappers (take data chunks to decrease initial starting time), adaptive combiners (perform local aggregation and maintain local cache for frequent keys), and adaptive sampling and partitioning (samples map output and produce balanced input to reducer) help in dynamically handling the situation. Adaptive MapReduce claims three times improved performance over Hadoop default scheduler.

Scheduling technique	Operating environment	Data locality	Data replication	Scope
Situation aware [67]	Nil	Nil	Nil	Task level

3.1.10. Replica-aware scheduling. Maestro [68], a replica-aware scheduling algorithm, introduces map task scheduling mechanism to improve issue of huge amount of network traffic caused by map tasks execution on remote data in Hadoop. Their mechanism consists of two steps. First, based on data replication and number of map tasks on a node, it occupies free slots on a DataNode. It tries to identify the nodes having more capacity to process, less chunk size than the average, and low share rate. Based on these factors, it calculates NodeWeights (NodeW) in ascending order while prioritizing them on sharing more data chunks with other nodes. After the process of selecting node with less NodeW, it processes the chunk with maximal weight, and then all nodes will be hosting the same number of chunks. Secondly, based on input replicas, the execution runtime is computed on a given machine. When a DataNode requests for a task assignment, Maestro calculates chunk weight Cw (probability of processing chunk with no-locally) and assigns the task with high Cw. Therefore, in the first step, it fills the slots of each node based on NodeW, and secondly, at runtime, it assigns the tasks based on Cw. To resolve the issue of heterogeneity in the cluster, Maestro calculates the Cw based on number of slots in each node, and they claim that it reflects the degree of heterogeneity. Maestro shows 95% improvement in speculative execution of data local map tasks and 34% improvement in execution time.

Scheduling technique	Operating environment	Data locality	Data replication	Scope
Replica aware [68]	Nil	Nil	Yes	Task level

DOI: 10.1002/cpe

3.1.11. Job-aware scheduling. Nanduri *et al.* [69] propose a model to maintain harmony among jobs in a cluster by allocating tasks to a node without affecting other tasks. In this model, when a job is divided into map and reduce tasks based on task characteristics, that is, memory intensive, CPU intensive, disk intensive, and network intensive, it maintains separate vectors for map tasks and reduce tasks called a TaskVector (Tk). It is defined as

$$\mathrm{Tk} = \mathrm{Ecpue1} + \mathrm{Ememe2} + \mathrm{Ediske3} + \mathrm{Enwe4} \tag{3}$$

where E shows a particular resource usage and e1, e2, e3, and e4 show their basis vector for a particular job. It uses a task queue and a task selection algorithm to identify which task tracker to assign the waiting tasks, and selected task along with TT is handed-over to the task assignment algorithm for approval.

Task assignment uses two algorithms to know the task compatibility to the requesting node. First is machine learning incremental Naive Bayes classifier to know whether the task is compatible or not to the TT specifications (network, CPU, and memory). The classifier is trained on the features including network between requesting TT and data split of the task, hardware specification of TT, TaskVector of incoming task and TaskVectors of running tasks on the requesting node to obtain Tcompound. Based on this feature set, it decides the task compatibility. Second is heuristic-based algorithm to test the compatibility of the task with TT. In this approach, it computes a Tavailability vector as difference of Tr (total resources) and Tcompound and then computes unused resources as Tunused. Based on these heuristics, a combination value, that is, value combination, is computed and compared with a threshold to decide the compatibility of the task to TT. Job-aware model shows 21% run-time saving in heuristic-based approach and 27% in machine learning approach.

In another research, J. Polo *et al.* [70] propose performance-driven co-task scheduling in MapReduce run-time to provide better resource usage shared by multiple jobs. They exploit the MapReduce multiple small tasks capability to predict the completion time and properties for the remaining tasks. Methodology to adjust resource allocation among jobs is based on completion time estimate for each job on some given resources. The scheduler presented in [70] is termed as preemptive, and it can preempt other jobs to allocate resources to a higher priority job. Slots are TT worker processes that are used as the basic resource allocation units. Experimental results and evaluation is based on four criteria: a) job completion estimation, job with deadline min-scheduler, job with deadlines max-scheduler, and comparison with fair scheduler. They claim it to be the first approach for MapReduce run-time performance management.

Scheduling Technique	Operating Environment	Data Locality	Data Replication	Scope
Job-aware [69]	Nil	Nil	Nil	Job/task Level

3.1.12. Paused Rate Monotonic (PRM) scheduling. Paused Rate Monotonic (PRM) [71] is a scheduling algorithm that addresses 'supporting concurrent services sharing in a cloud' and 'guaranteed response time for deadline constrained services'. PRM proposes a real-time scheduling mechanism to provide general formulation and theoretical analysis of Hadoop scheduling. It addresses the real-time scheduling problem in several aspects. First, it models a service as series of map and reduce tasks and formulates the problem by specifying the characteristics of task, cluster, and algorithm. Hadoop cloud is abstracted as an exclusive cloud and supports preemption with little overhead. Second, it proposes PRM scheduling algorithm for priority-driven algorithms and to schedule the tasks based on their constraints. It pauses the MapReduce job between map and reduce stages to optimally use this time for partitioning, sorting, and combining the intermediate data. PRM aims at maximizing the number of task meeting their deadlines. The proposed solution is analyzed both theoretically and experimentally and claims better performance than other real-time scheduling mechanisms for MapReduce. While in [72], the authors have worked on dynamic acyclic graphs and estimated progress in MapReduce.

DOI: 10.1002/cpe

3.2. Taxonomy of scheduling techniques

In this section, we present a taxonomical representation of most of the discussed and referred scheduling techniques. Based on salient features of scheduling algorithms/techniques, their main themes and ideas, and targeted goals, we classify them initially in three different categories such as job level, task level, and hybrid. The techniques that target only MapReduce job scheduling are categorized as job-level schedulers, those targeting maps and reduces in a single job are termed as task-level schedulers, and the hybrid schedulers propose a solution for both jobs and tasks in MapReduce. The taxonomy is developed to provide an insight into various scheduling techniques developed so far to identify their targeting performance metrics improvements and recognize the future directions of scheduling in MapReduce. Performance metrics in Hadoop MapReduce include data locality, data replication, cluster environment (homogeneous and heterogeneous), network communication, intermediate data, data movement, fault tolerance, and resource utilization. Criteria for taxonomy development is described in the following subsection.

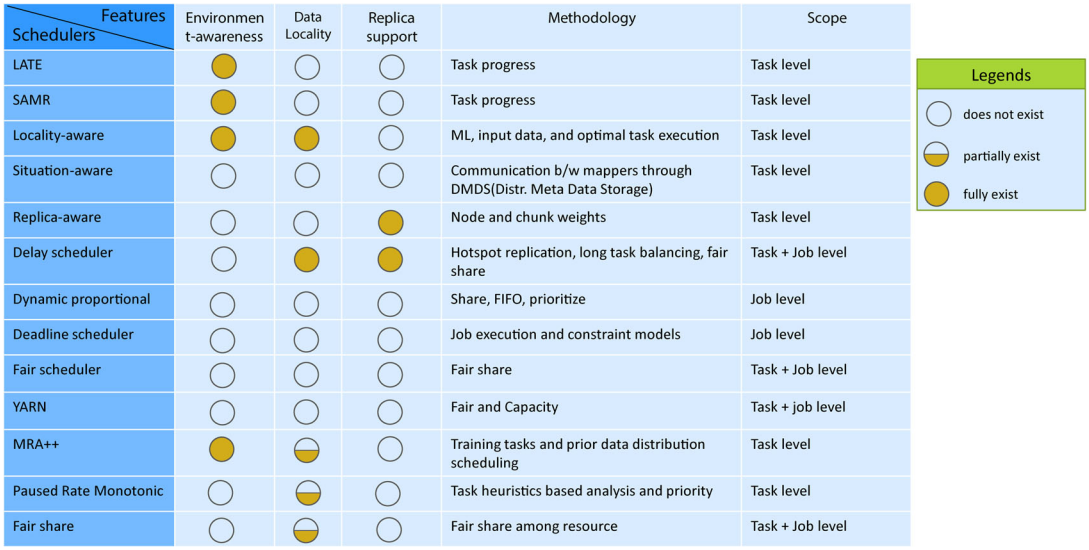
3.2.1. Taxonomical chart. The taxonomy in Figure 4 is developed keeping in view the Hadoop MapReduce execution, its operating environment, and performance metrics. Hadoop scheduling can be initially classified in two ways: job scheduling and task scheduling [73–77]. In job scheduling, most of the techniques incorporate scheduling multiple jobs in a single Hadoop cluster at an instance of time for better resource utilization and service provisioning. It also incorporate sharing of the cluster among multiple organizations in a fair share, capacity, and harmonious manner. On the other hand, task scheduling only cares about the resources allotted for a single job and schedules tasks in the available resource environment. Some techniques propose solutions for both jobs in the cluster and tasks in a single job, and they are referred as hybrid in this taxonomy. Referring to the job scheduling, various techniques have been proposed including delay scheduling, DP share scheduling, and deadline scheduler. Hybrid techniques such as Hadoop newer version of MapReduce, that is, YARN, uses fair scheduling mechanism to fairly share the cluster resources and capacity scheduling for its management among jobs. In task scheduling, the main purpose is to efficiently schedule multiple tasks in a single MapReduce job in heterogeneous environments with data locality, data replica, and situation of the individual tasks. These goals are achieved in the discussed techniques by avoiding task failures, recovery of failed tasks, communication among tasks, and data replication in the cluster. Based on these metrics, task scheduling is further divided into replica awareness (data replication), locality awareness (data locality), situation awareness (between tasks communication), and environment awareness (homogeneous and heterogeneous).

The top-level and middle-level nodes show the research areas and criteria, which have been addressed in techniques discussed in this paper. The leaf nodes represent the names of actual techniques. We have listed techniques that broadly fall into these categories, and it will help in identifying the basic requirements for design and development of a technique that can be used to address these issues in Hadoop MapReduce. In this taxonomical representation, it can easily be identified that we need some proposals to resolve issues in regard to the operating environment, resource usage and sharing, and the need to address the problems of batch processing and online processing in MapReduce distributed environment.

4. DISCUSSION

In this paper, we discuss various scheduling techniques that propose different approaches to improve Hadoop MapReduce job/task scheduling. There are already many scheduling techniques for distributed and HPC systems; however, Hadoop has its own processing mechanism with the aim of moving computation to the data and making each participating machine/node in the cluster an independent data-intensive compute node. MapReduce's low coupling between DataNodes, data local computation, independent data sets processing, and fault-tolerant infrastructure has led the research community to propose and develop new methods for improving MapReduce performance. In the previously discussed techniques and systems, some of them target to improve Hadoop performance to improve job scheduling (many MapReduce jobs) and some target to improve task scheduling in

DOI: 10.1002/cpe



Features / Schedulers	Environment-awareness	Data Locality	Replica support	Methodology	Scope
LATE	●	○	○	Task progress	Task level
SAMR	●	○	○	Task progress	Task level
Locality-aware	●	●	○	ML, input data, and optimal task execution	Task level
Situation-aware	○	○	○	Communication b/w mappers through DMDS(Distr. Meta Data Storage)	Task level
Replica-aware	○	○	●	Node and chunk weights	Task level
Delay scheduler	○	●	●	Hotspot replication, long task balancing, fair share	Task + Job level
Dynamic proportional	○	○	○	Share, FIFO, prioritize	Job level
Deadline scheduler	○	○	○	Job execution and constraint models	Job level
Fair scheduler	○	○	○	Fair share	Task + Job level
YARN	○	○	○	Fair and Capacity	Task + job level
MRA++	●	◒	○	Training tasks and prior data distribution scheduling	Task level
Paused Rate Monotonic	○	◒	○	Task heuristics based analysis and priority	Task level
Fair share	○	◒	○	Fair share among resource	Task + Job level

Figure 5. Comparative chart of Hadoop scheduling schemes. Figure depicts individual schemes with their addressing problems and scope.

a single MapReduce job. In Hadoop job scheduling, most of the systems provide better user experience and services by fair scheduling, DP sharing, job awareness, and user-provided deadline-based scheduling techniques. On the other hand, in task scheduling, most systems have targeted various aspects of Hadoop like data locality, data replication, optimization of map tasks assignment in a job, adaptive mappers/combiners/reducers, and different algorithms like machine learning and heuristics techniques to improve Hadoop MapReduce performance.

Hadoop clusters may consist of only homogeneous (nodes with the same hardware and specifications) or heterogeneous (virtual machines working as nodes, machines with different hardware specification) computing nodes. Heterogeneity of cluster may affect task running time and overall MapReduce job performance. To address this scenario, some scheduling schemes like LATE, SAMR, and heterogeneous environment Locality-aware schedulers discussed in Sections 3.1.3, 3.1.4, and 3.1.8-b, respectively, target the heterogeneous environment. However, these schemes address some parts of the problem like LATE addresses speculative execution of failed, straggling, and non-running tasks; SAMR is proposed as the successor of LATE to improve its performance; and heterogeneous environment scheduler addresses the data locality. A comparative and detailing chart as shown in the Figure 5 explicitly shows the scope, methodology, heterogeneous environment support, and other metrics addressed in all the systems.

Based on these scheduling algorithms and discussion, Hadoop MapReduce scheduling mechanism for job scheduling and task scheduling needs to be addressed. Most of the techniques in the literature address a single metric of performance such as locality, replication, better resource utilization, and network communication. However, no such system exists that address all these areas. Hadoop MapReduce performance is dependent on these factors; therefore, more research work is needed to improve MapReduce performance by inculcating different metrics that address most of the problems of heterogeneity, data locality, data replication, context-awareness task scheduling, and network communication in Hadoop.

From this literature study, it is clear that we need to come up with a unified scheduling model to address the limitations of existing systems. The most prominent and important features needed to be addressed include heterogeneity of the operating environment of MapReduce. Many novel techniques have already been discussed; however, the research community need to address this limitation in broader context along with data locality, tasks situation, and sharing of cluster among various organizations. Hadoop MapReduce is currently referred as data-intensive batch-processing framework; however, currently, real-time big data need frameworks to acquire the data in real time.

DOI: 10.1002/cpe

Hence, new approaches are needed to address limitations of real-time big data acquisition, processing, and management. These limitations can be addressed using MapReduce with unprecedented techniques and methods of scheduling and processing.

5. CONCLUSION

Hadoop has become a de facto platform for large-scale distributed data-intensive applications and data analytics. Development of a Hadoop scheduling mechanism that provides better performance in a dynamic and variant cluster environment is a key research problem as the current implementations are based on fixed configuration. This paper discusses pros and cons, addressed scheduling problems, and scheduling policies of several Hadoop schedulers developed by research communities so far. We classify, categorize, and reformulate these schedulers and present a taxonomy based on their addressed problems like cluster heterogeneity, data locality, data replication, job scheduling, and service provisioning. Each scheduler discussed and addressed one or more problems in Hadoop default scheduling. Most of the schedulers discussed in this paper cogitates to capture the run-time status and context to schedule dynamically and optimize Hadoop performance. Some of the techniques consider the heterogeneity and homogeneity of clusters environment. Every scheduling technique discussed here considers the resources like I/O, processing, storage, and service provisioning. In Hadoop scheduling, forward step is how to combine and produce mechanism to resolve the resource availability, heterogeneity, interoperability, and service provisioning to the end users.

ACKNOWLEDGEMENTS

This class file was developed by Sunrise Setting Ltd, Torquay, Devon, UK (website: `www.sunrise-setting.co.uk`).

REFERENCES

1. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. *Communications of the ACM* 2010; **53.4**:50–58.
2. Garfinkel S. The cloud imperative. *Technology Review* 2011; **114**:74–75.
3. Mell P, Tim G. The NIST definition of cloud computing, 2011.
4. Howe D, Costanzo M, Fey P, Gojobori T, Hannick L, Hide W, Hill DP, Kania R, Schaeffer M, St Pierre S, Twigger S, White O, Rhee SY. Big data: the future of biocuration. *Nature* 2008; **455.7209**:47–50.
5. Manyika J, Chui M, Brown B, Bughin J, Dobbs R, Roxburgh C, Byers AH. Big data: the next frontier for innovation, competition, and productivity, 2011.
6. Apache. Hadoop, 2013. (Available from: http://hadoop.apache.org/) [Accessed on 30 April 2015].
7. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. *Communications of the ACM* 2008; **51.1**:107–113.
8. Shahrivari S. BeyondBatch processing: towards real-time and streaming big data. *Computers* 2014; **3.4**:117–129.
9. Hadoop Community. Fair Scheduler:. (Available from: https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.pdf) [Accessed on 30 April 2015].
10. Apache. Capacity Scheduler, 2013. (Available from: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html) [Accessed on 30 April 2015].
11. Azeez A, Perera S, Gamage D, Linton R, Siriwardana P, Leelaratne D, Weerawarana S, Fremantle P. Multi-tenant SOA middleware for cloud computing. In *2010 IEEE 3rd International Conference on Cloud Computing (Cloud)*. IEEE: Mountain View, CA, USA, 2010.
12. Wu S. Context-Aware Computing, 2001.
13. Schilit B, Adams N, Roy W. Context-aware computing applications. *First Workshop on IEEE Mobile Computing Systems and Applications. 1994, WMCSA 1994*, Santa Cruz, CA, USA, 1994.
14. Yoo D, Sim KM. A comparative review of job scheduling for MapReduce. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*. IEEE: Gwangju, South Korea.
15. Rao BT, Reddy LSS. Survey on improved scheduling in Hadoop MapReduce in cloud environments, 2012. arXiv preprint arXiv 1207.0780.
16. Tiwari N, Sarkar S, Bellur U, Indrawan M. Classification framework of MapReduce scheduling algorithms. *ACM Computing Surveys (CSUR)* 2015; **47.3**:49.
17. Pakize SR. A comprehensive view of Hadoop MapReduce scheduling algorithms. *International Journal of Computer Networks & Communications Security* 2014; **2**:9.

DOI: 10.1002/cpe

18. Zaharia M, Konwinski A, Joseph AD, Katz RH, Stoica I. Improving MapReduce performance in heterogeneous environments. *OSDI* 2008; **8**(4):7.
19. Chen Q, Zhang D, Guo M, Deng Q, Guo S. SAMR: a self-adaptive MapReduce scheduling algorithm in heterogeneous environment. In *2010 IEEE 10th International Conference on Computer and Information Technology (CIT)*. IEEE: Shanghai Jiao Tong Univ., Shanghai, China, 2010; 2736–2743.
20. Wang F, Qiu J, Yang J, Dong B, Li X, Li Y. Hadoop high availability through metadata replication. In *Proceedings of the First International Workshop on Cloud Data Management*. ACM: New York, NY, USA, 2009; 37–44.
21. Ghemawat S, Gobioff H, Leung ST. The Google file system. *ACM SIGOPS Operating Systems Review* 2003; **37**(5):29–43. ACM.
22. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. *Communications of the ACM* 2008; **51**(1):107–113.
23. Borthakur D. Facebook has the worlds largest Hadoop cluster. Retrieved April 20 (2010): 2012.
24. Kaushik RT, Bhandarkar M. GreenHDFS: towards an energy-conserving, storage-efficient, hybrid Hadoop compute cluster. *Proceedings of the USENIX Annual Technical Conference*, Boston, MA, 2010.
25. Dijcks JP. Oracle: big data for the enterprise. *Oracle White Paper*, 2012.
26. Gunarathne T, Wu T-L, Qiu J, Fox G. MapReduce in the clouds for science. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE: Bloomington, IN, USA, 2010.
27. Jalote P, Jalote P. *Fault Tolerance in Distributed Systems*. PTR Prentice Hall: Englewood Cliffs, 1994.
28. Bolchini C, Curino CA, Quintarelli E, Schreiber FA, Tanca L. A data-oriented survey of context models. *ACM Sigmod Record* 2007; **36**(4):19–26.
29. Phan LTX, Zhang Z, Loo BT, Lee I. *Real-time MapReduce scheduling*, 2010.
30. Zaharia M, Borthakur D, Sen Sarma J, Elmeleegy K, Shenker S, Stoica I. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European Conference on Computer Systems*. ACM: New York, NY, USA, 2010.
31. Zaharia M, Borthakur D, Sarma JS, Elmeleegy K, Shenker S, Stoica I. Job scheduling for multi-user MapReduce clusters. *Technical Report UCB/EECS-2009-55*, EECS Department, University of California: Berkeley, 2009.
32. Kc K, Anyanwu K. Scheduling Hadoop jobs to meet deadlines. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE: North Carolina State Univ., Raleigh, NC, USA, 2010; 388–392.
33. Zhang X, Feng Y, Feng S, Fan J, Ming Z. An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments. In *2011 International Conference on Cloud and Service Computing (CSC)*. IEEE: Henan Polytech. Univ., Jiaozuo, China, 2011; 235–242.
34. Arslan E, Mrigank S, Kosar T. Locality and network-aware reduce task scheduling for data-intensive applications. In *Proceedings of the 5th International Workshop on Data-Intensive Computing in the Clouds*. IEEE Press: Piscataway, NJ, USA, 2014; 17–24.
35. Palanisamy B, Singh A, Liu L, Jain B. Purlieus: locality-aware resource allocation for MapReduce in a cloud. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM: New York, NY, USA, 2011.
36. Zaharia M, Konwinski A, Joseph AD, Katz RH, Stoica I. Improving MapReduce performance in heterogeneous environments. *OSDI* 2008; **8**(4):58.
37. Moseley B, Dasgupta A, Kumar R, Sarlós T. On scheduling in MapReduce and flow-shops. In *Proceedings of the Twenty-Third Annual ACM Symposium on Parallelism in Algorithms and Architectures*. ACM: New York, NY, USA, 2011; 289–298.
38. Vernica R, Balmin A, Beyer KS, Ercegovac V. Adaptive MapReduce using situation-aware mappers. In *Proceedings of the 15th International Conference on Extending Database Technology*. ACM: New York, NY, USA, 2012; 420–431.
39. Vavilapalli VK, Murthy AC, Douglas C, Agarwal S, Konar M, Evans R, Graves T, Lowe J, Shah H, Seth S, Saha B, Curino C, O'Malley O, Radia S, Reed B, Baldeschwieler E. Apache Hadoop yarn: yet another resource negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing*. ACM: New York, NY, USA, 2013; 5.
40. Chen Q, Zhang D, Guo M, Deng Q, Guo S. SAMR: a self-adaptive MapReduce scheduling algorithm in heterogeneous environment. In *2010 IEEE 10th International Conference on Computer and Information Technology (CIT)*. IEEE: Shanghai Jiao Tong Univ., Shanghai, China, 2010.
41. Kang H, Chen Y, Wong JL, Sion R, Wu J. Enhancement of Xen's scheduler for MapReduce workloads. In *Proceedings of the 20th International Symposium on High Performance Distributed Computing*. ACM: San Jose, California, 2011.
42. Ibrahim S. *et al.* Adaptive disk I/O scheduling for MapReduce in virtualized environment. In *2011 International Conference on Parallel Processing (ICPP)*. IEEE: Huazhong Univ. of Sci. & Technol., Wuhan, China, 2011; 251–262.
43. Yoo AB, Jette MA, Slurm MG. Simple Linux Utility for Resource Management. In *Job Scheduling Strategies for Parallel Processing*. Springer: Berlin, Heidelberg, 2003; 44–60.
44. HPC Admin Magazine. The YARN Invitation. (Available from: http://www.admin-magazine.com/HPC/Articles/The-New-Hadoop) [Accessed on 28 July 2015].
45. Ari I, Kocak U. Hybrid job scheduling for improved cluster utilization. In *Euro-Par 2013: Parallel Processing Workshops*. Springer: Berlin, Heidelberg; 395–405.

DOI: 10.1002/cpe

46. Castain RH, Tan W. MR+: a technical overview. (Available from: https://www.open-mpi.org/video/mrplus/Greenplum_RalphCastain-1up.pdf) [Accessed on 27 July 2015].
47. The Apache Software Foundation. Apache Mesos. web-address: (Available from: http://mesos.apache.org/documentation/latest/mesos-architecture/) [Accessed on 05 May 2015].
48. Rao BT, Sridevi NV, Reddy VK, Reddy LSS. Performance issues of heterogeneous Hadoop clusters in cloud computing, 2012. arXiv preprint arXiv 1207.0894.
49. Amazon. Amazon Elastic Cloud (EC2), 2014. (Available from: http://aws.amazon.com/ec2) [Accessed on 30 April 2015].
50. Jin S, Yang S, Jia Y. Optimization of task assignment strategy for MapReduce. In *2012 2nd International Conference on Computer Science and Network Technology (ICCSNT)*. IEEE: Nat. Univ. of Defense Technol., Changsha, China, 2012; 57–61.
51. Cosulschi M, Gabroveanu M, Sbrcea A. Implementing MapReduce in virtualization environment: Hadoop case study. Vol. 1. TR 2012, (vol. 1), 2012.
52. Prabhu S, Rodrigues AP. Hadoop MapReduce job scheduler implementation and analysis in heterogeneous environment. *IJRCCT* 2015; **4.3**:229–233.
53. Jung H, Nakazato H. Dynamic scheduling for speculative execution to improve MapReduce performance in heterogeneous environment. In *2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE: Waseda Univ., Tokyo, Japan, 2014.
54. Anjos JCS, Carrera I, Kolberg W, Tibola AL, Arantes LB, Geyer CR. MRA++: scheduling and data placement on MapReduce for heterogeneous environments. *Future Generation Computer Systems* 2015; **42**:22–35.
55. Chen Qi, Liu C, Xiao Z. Improving MapReduce performance using smart speculative execution strategy. (2013): 1-1.
56. Zaharia M, Borthakur D, Sen Sarma J, Elmeleegy K, Shenker S, Stoica I. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European Conference on Computer Systems*. ACM: Paris, France, 2010; 265–278.
57. Medina E, Shemla D, Solt Y. Head of line blocking, 2004. U.S. Patent No. 6,829,245. 7.
58. Shafer J, Rixner S, Cox AL. The Hadoop distributed file system: balancing portability and performance. In *2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*. IEEE: Rice Univ., Houston, TX, USA, 2010; 122–133.
59. Shih KY, Srinivasan U. Method and system for data replication, 2003. U.S. Patent No. 6,615,223. 2.
60. Kc K, Kemafor A. Scheduling Hadoop jobs to meet deadlines. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE: North Carolina State Univ., Raleigh, NC, USA; 388–392.
61. Sandholm T, Kevin L. Dynamic proportional share scheduling in Hadoop. In *Job Scheduling Strategies for Parallel Processing*. Springer: Berlin Heidelberg, 2010; 110–131.
62. Nelson C. Preemption. *Virginia Law Review* 2000; **99**:225–305.
63. Sandholm T, Lai K. MapReduce optimization using regulated dynamic prioritization. *ACM SIGMETRICS Performance Evaluation Review* 2009; **37**(1):299–310.
64. Appuswamy R, Gkantsidis C, Narayanan D, Hodson O, Rowstron A. Scale-up vs scale-out for Hadoop: time to rethink? In *Proceedings of the 4th Annual Symposium on Cloud Computing*. ACM: Santa Clara, CA, 2013; 20.
65. Waldspurger CA, Weihl WE. Lottery scheduling: flexible proportional-share resource management. In *Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation*. USENIX Association: Berkerly, CA, USA, 1994; 1.
66. Zhang X, Feng Y, Feng S, Fan J, Ming Z. An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments. In *2011 International Conference on Cloud and Service Computing (CSC)*. IEEE: Henan Polytech. Univ., Jiaozuo, China, 2011; 235–242.
67. Vernica R, Balmin A, Beyer KS, Ercegovac V. Adaptive MapReduce using situation-aware mappers. In *Proceedings of the 15th International Conference on Extending Database Technology*. ACM: Berlin, Germany, 2012; 420–431.
68. Ibrahim S, Jin H, Lu L, He B, Antoniu G, Wu S. Maestro: Replica-aware map scheduling for MapReduce. In *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE: Huazhong Univ. of Sci. & Technol., Wuhan, China, 2012; 435–442.
69. Nanduri R, Maheshwari N, Reddyraja A, Varma V. Job aware scheduling algorithm for MapReduce framework. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE: HIT, Hyderabad, India, 2011; 724–729.
70. Polo J, Carrera D, Becerra Y, Torres J, Ayguadé E, Steinder M, Whalley I. Performance-driven task co-scheduling for MapReduce environments. In *2010 IEEE Network Operations and Management Symposium (NOMS)*. IEEE: Tech. Univ. of Catalonia (UPC), Barcelona, Spain, 2010; 373–380.
71. Teng F, Magoulès F, Yu L, Li T. A novel real-time scheduling algorithm and performance analysis of a MapReduce-based cloud. *The Journal of Supercomputing* 2014; **69**(2):739–765.
72. Morton K, Balazinska M, Grossman D. ParaTimer: a progress indicator for MapReduce DAGs. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. ACM: Indianapolis, Indiana, USA, 2010; 507–518.
73. Zaharia M, Borthakur D, Sarma JS, Elmeleegy K, Shenker S, Stoica I. Job scheduling for multi-user MapReduce clusters. *Technical Report UCB/EECS-2009-55*, EECS Department, University of California: Berkeley, 2009.
74. Tayal S. Tasks scheduling optimization for the cloud computing systems. *International Journal of Advanced Engineering Sciences And Technologies (IJAEST)* 2011; **5.2**:111–115.

DOI: 10.1002/cpe

75. Nita M-C, Pop F, Voicu C, Dobre C, Xhafa F. MOMTH: multi-objective scheduling algorithm of many tasks in Hadoop. *Cluster Computing* 2015:1–14.
76. Suresh S, Gopalan NP. An optimal task selection scheme for Hadoop scheduling. *IERI Procedia* 2014; **10**:70–75.
77. Wang C. *et al.* Optimal task scheduling in MapReduce. In *2014 9th IEEE International Conference on Networking, Architecture, and Storage (NAS)*. IEEE, 2014.

DOI: 10.1002/cpe