Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Energy-efficient hadoop for big data analytics and computing: A systematic review and research insights

WenTai Wu [a], WeiWei Lin [a,*], Ching-Hsien Hsu [b], LiGang He [c]

[a] School of Computer Science and Engineering, South China University of Technology, Guangdong, China
[b] Department of Computer Science and Information Engineering, Chung Hua University, Taiwan
[c] Department of Computer Science, University of Warwick, Coventry, CV4 7AL, United Kingdom

HIGHLIGHTS

- This paper presents the new viewpoints/insights in improving the energy efficiency of Hadoop.
- Present valuable and feasible solutions towards improving the energy efficiency of Hadoop.
- Propose five categories of optimizing the energy efficiency of Hadoop.
- Present instructive research insights in future research directions of energy-efficient Hadoop.

ARTICLE INFO

Article history:
Received 11 August 2017
Received in revised form 14 October 2017
Accepted 5 November 2017
Available online xxxx

Keywords:
Energy efficiency
Hadoop
MapReduce
Data centers
Big data analytics

ABSTRACT

As the demands for big data analytics keep growing rapidly in scientific applications and online services, MapReduce and its open-source implementation Hadoop gained popularity in both academia and enterprises. Hadoop provides a highly feasible solution for building big data analytics platforms. However, defects of Hadoop are also exposed in many aspects including data management, resource management, scheduling policies, etc. These issues usually cause high energy consumption when running MapReduce jobs in Hadoop clusters. In this paper, we review the studies on improving energy efficiency of Hadoop clusters and summarize them in five categories including the energy-aware cluster node management, energy-aware data management, energy-aware resource allocation, energy-aware task scheduling and other energy-saving schemes. For each category, we briefly illustrate its rationale and comparatively analyze the relevant works regarding their advantages and limitations. Moreover, we present our insights and figure out possible research directions including energy-efficient cluster partitioning, data-oriented resource classification and provisioning, resource provisioning based on optimal utilization, EE and locality aware task scheduling, optimizing job profiling with machine learning, elastic power-saving Hadoop with containerization and efficient big data analytics on Hadoop. On one hand, the summary of studies on energy-efficient Hadoop presented in this paper provides useful guidance for the developers and users to better utilize Hadoop. On the other hand, the insights and research trends discussed in this work may inspire the relevant research on improving the energy efficiency of Hadoop in big data analytics.

1. Introduction

MapReduce [1], a parallel computing paradigm proposed by Google, has gained a wide adoption due to its features including fault tolerance, high scalability and simplicity in programming. With the ever-growing demands for analyzing large-scale datasets, MapReduce has become a mainstream programming model for the applications of big data analytics [2,3]. The most popular open-source implementation of MapReduce is Hadoop [4], which was originally developed by Yahoo! and now has been widely deployed on production clusters for commercial purposes [5]. Hadoop well screens the complexity of underlying hardware systems and provides high-level programming interfaces, which are designed for processing large-scale datasets following MapReduce paradigm [6]. Besides, it transparently provides applications with scalability and reliable data storage on a cluster. Due to these features, Hadoop has gained wide adoption in many fields of research such as bioinformatics, social network, healthcare and business intelligence.

There is no perfect paradigm or framework in this world. Hadoop provides a highly feasible solution for distributed computing, but meanwhile exposes a number of shortcomings in performance and energy efficiency [7–9]. For example, Zaharia et al. [7] figured out that the default scheduler of Hadoop makes unrealistic assumptions that different worker nodes have same performance

* Corresponding author.
E-mail address: linww@scut.edu.cn (W. Lin).

https://doi.org/10.1016/j.future.2017.11.010
0167-739X/

and throughput and that launching speculative tasks in idle slots will not induce extra time consumption. Generally, these assumptions do not hold in heterogeneous data centers, which may cause the scheduler's decisions to be sub-optimal in terms of job completion time and energy consumption. Since the adoption of Hadoop is extending from batch jobs (e.g., offline log analysis) to streaming data processing and ad hoc data query, people are paying more attention to the defects of Hadoop. They are exposed especially when the system has to run a large proportion of short jobs. Chen et al. [8] developed a MapReduce-oriented benchmark suite and carried out experiments with a one-day long Hadoop workload synthesized from Facebook traces. From the benchmarking results they found that FIFO scheduler might incur the failures of many jobs when long jobs were submitted constantly. As a matter of fact, the scheduler is not the only implementation that needs to be improved in Hadoop. For example, the default task scheduling implementation of Hadoop ignores the performance and workload of servers, which probably vary largely in a heterogeneous cluster. A variety of issues have been raised and most of them are associated to the inefficiency of Hadoop in node management, data management, resource management and task scheduling.

Previous studies mainly focus on how to improve the performance of Hadoop clusters by shortening job execution time [10–12], balancing task execution progress [13,14], reducing the impact of task failures [15–17], etc. However, these years the large amount of energy consumed by data centers emerged to be a prominent issue [18,19]. This made energy saving a topic of interest for MapReduce applications [9]. For example, Yang et al. [20] built a high-performance computing/storage platform using Hadoop for big data processing. They proposed to collect real-time power data of servers via wireless power sensors. As a result, the fine-grained monitoring system can help control the cluster's power consumption and can be integrated with warning and prediction modules. Due to the popularity and open-source nature of Hadoop, a large number of works can be found related to reducing energy consumption of Hadoop clusters. The study of [21] briefly divides them into five categories. However, it does not illustrate their methodologies and implementations in detail. Rao and Reddy [22] analyzes different types of Hadoop schedulers including the embedded FIFO scheduler, Fair scheduler and Capacity scheduler. Improved schemes such as Delay scheduler, Dynamic Priority scheduler and Resource Aware scheduler are also reviewed in their work. However, they are not compared directly in terms of advantages and shortcomings. Besides, energy-aware schedulers are not included. The study of [23] categorizes the researches on scheduling into two groups: cluster-based scheduling and resource-based scheduling. The authors also compare several energy-aware schedulers (e.g., [24–27]) in the paper. But their research is limited to task scheduling. Actually there are a variety of methods and techniques available for improving Hadoop's energy efficiency. Hameed et al. [28] present a practical taxonomy of energy-saving techniques in cloud environment and compare them from the perspectives of resource adaption strategy, target function, allocation and migration policy. Their result provides theoretical guidance for reducing energy consumption in generic cloud systems but not in a specific framework such as Hadoop. In this paper, we summarize mainstream energy-saving schemes and strategies specifically focusing on Hadoop/MapReduce. In each category, we review the state-of-the-art studies and make comparison in terms of their applicable situations, advantages and limitations.

In this paper, the studies on optimizing Hadoop energy efficiency are divided into five categories:

- **Energy-efficient worker node management**. This category surveys the studies on saving energy by dynamically scaling the cluster size (number of workers) and the CPU frequency of the servers.
- **Energy-efficient data management**. This category focuses on data distribution on HDFS. For example, the cost of data transfer can be reduced (i.e., achieving better data locality) through well-designed placing strategies for data replicas and migration schemes for data blocks between Datanodes.
- **Energy-efficient resource allocation**. The scheduler determines the resource share of every job and dynamically reorders the job queue in order to achieve energy saving at the system level.
- **Energy-efficient task scheduling**. A plan of task scheduling is made after comprehensively considering the factors such as data locality, server performance and Service Level Agreement (SLA).
- **Other energy-saving schemes**. Apart from the categories mentioned above, we also introduce some other energy-saving schemes such as data sampling, file merging and using renewable energy.

For each category, we first reveal its basic rationale and introduce every relevant work in detail. Most of these studies present valuable and feasible solutions towards improving the energy efficiency of Hadoop. Moreover, we make comparisons between them and list their pros and cons, which would be extremely useful when trying to apply them to the realistic environment. More importantly, in this paper we present instructive research insights in the discussion about future research directions including energy-efficient cluster partitioning, data-oriented resource classification and provisioning, resource provisioning based on optimal utilization, EE and locality aware task scheduling, optimizing job profiling with machine learning, elastic power-saving Hadoop with containerization and efficient big data analytics on Hadoop. More and more scientific and service applications are directly or indirectly deployed on the platform of Hadoop because of its great potential in big data analytics. Meanwhile, optimizing energy consumption has become a major trend and the topic of interest. In this paper, we systematically review the studies on improving Hadoop's energy efficiency, which offers useful guidance to the users and developers for better utilization of Hadoop. We further discuss some possible improvements and research directions in order to provide instructive insights for the relevant research work on developing energy-aware Hadoop systems.

The rest of the paper is organized as follows. Section 2 is about the background knowledge of Hadoop. Section 3 briefly introduces the application of Hadoop for big data analytics in different research fields. In Section 4, we review the relevant studies on optimizing the energy efficiency of Hadoop by organizing them into five categories. In Section 5, we discuss the directions for future research. Finally, we conclude the paper in Section 6.

2. Background

2.1. MapReduce

MapReduce [1] is a parallel programming framework proposed by Google and designed for data processing in distributed environments. A MapReduce job is mainly composed of two phases: Map and Reduce. Initially the input dataset of the job is split into several blocks while each block corresponds to a single Map task. Typically, each Map task processes a data block and produces a set of intermediate key/values pairs. The finish of Map phase is followed by Shuffle, in which intermediate outputs are collected from every map task and sent to corresponding Reduce tasks after being sorted and partitioned. The temporal results are merged at the Reducer-side at the end of Shuffle. In Reduce phase, every Reducer receives an assigned set of keys and combines all the associated values for output. The process of a MapReduce job is demonstrated in Fig. 1.

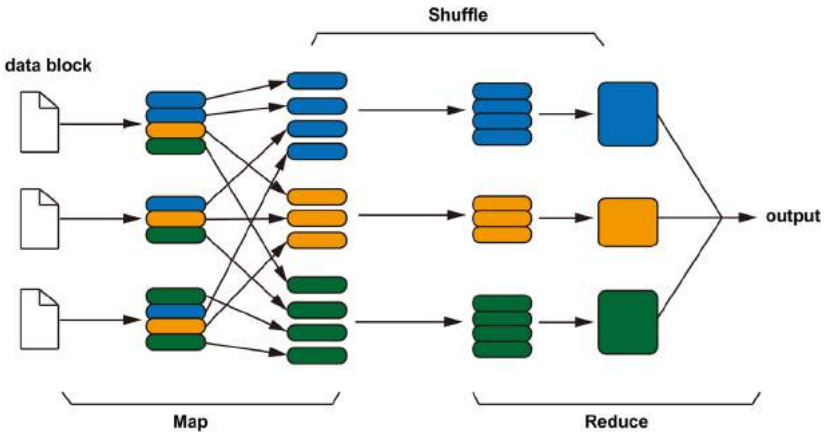

Fig. 1. The process of a MapReduce job.

2.2. Hadoop

As the most famous open-source implementation of MapReduce, Hadoop [4] is now a top-level Apache project. With its constant development, Hadoop has become the de facto framework for large data processing. On one hand, Hadoop releases the users from directly facing hardware complexity of the infrastructure. On the other hand, it simplifies the deployment and resource management in a distributed system.

In Hadoop 1.0 (including Apache Hadoop 0.20.x and 1.x, CDH3, etc.), the JobTracker running on the master node is responsible for monitoring jobs and managing resources. It constantly communicates with the TaskTrackers on worker nodes to track the execution of MapReduce jobs. Each worker node has pre-defined numbers of Map slots and Reduce slots. Hadoop 1.0 only supports static and coarse-grained configurations for Map/Reduce slots (1 CPU core and 2G memory for each slot by default).

In Hadoop 2.0 (including Apache Hadoop 0.23.x and 2.x, CHD4, etc.), YARN was adopted as a dedicated system to deal with resource management. The performance bottleneck of JobTracker is eliminated by separating job management and resource management. YARN adopts the ResourceManager (RM) to manage the resources in the whole cluster. The JobTracker and TaskTracker are replaced by ApplicationMaster (AM) and NodeManager, respectively. Moreover, YARN uses container as the basic unit in resource allocation. Container encapsulates CPU and memory resources and enables more flexible configurations. AM is also running in a container and keeps requiring resources from the RM for its corresponding job. RM then responds with a list of containers as the allocated resource share. At present, the latest version of Hadoop is Hadoop 3.0. But compared with Hadoop 2.0, only a few features such as shuffling using Java Native Method, multiple standby Namenodes and the support for erasure encoding in HDFS, are added.

2.3. HDFS

Hadoop Distributed File System [29] or HDFS is the underlying distributed file system that supports the persistent running of Hadoop. HDFS can be deployed on dedicated servers or commodity machines. It is capable of storing terabytes and petabytes of data. Users can assess HDFS through its abstract interfaces as if they are operating in a local file system. HDFS adopts WORM (Write Once, Read Many times) model to simplify data integration and increase throughput. Hadoop stores redundant copies of data for increasing the system's availability. The replication mechanism of HDFS allows one data block to have multiple replicas distributed on different machines. In order to guarantee fault tolerance, updates of metadata are logged on Namenode.

3. Big data analytics on Hadoop

As big data offers opportunities for scientists to generate, store, access and analyze massive amount of experimental data in a fast and low-cost manner, the adoption of big data techniques is experiencing an unprecedented growth in a diversity of research fields. More and more scientific applications are developed following the MapReduce programming model and taking advantage of the powerful computing and storage capability of Hadoop. For example, bioinformatics systems usually need to provide medical imaging, modeling and simulation services based on terabytes of biological data [30], which requires a fault-tolerant and efficient distributed storage system as well as a powerful and scalable parallel framework for the whole workflow. On this point, Hadoop is particularly applicable. O'Driscoll et al. [31] figured out that Hadoop provides a highly feasible solution to data analytics on large genomics and medical datasets. They also surveyed several MapReduce projects and applications applied in the biotechnology such as CloudAligner [32], BlastReduce[1] and Hydra [33]. MapReduce framework and Hadoop are also widely-used in weather data analytics and geo-data processing. Chang [34] presents a framework based on MapReduce for analyzing weather data and simulating temperature distributions. The author first used MapReduce to forecast temperature of three cities in a period of over two years and demonstrates in the paper its accuracy. Then the paper illustrates an optimized eight-step process of MapReduce for visualizing temperature distribution. Gao et al. [35] gathered gazetteer entries from social media and implemented a scalable platform based on Hadoop for processing big geo-data and facilitating the development of crowd-sourced gazetteers. They integrated Hadoop with third-party geometry APIs to spatially enable the platform for processing geo-tagged datasets. Their experimental result shows that using the Hadoop cluster achieves much higher efficiency than running the geo-processing workflow on a single computer.

As more and more data-intensive applications are migrated from private infrastructures to the cloud, Hadoop also gained wide adoption in cloud computing environments for big data analytics [36]. Large-scale cloud environments provide great processing power and massive storage space, but also increase the difficulties in the runtime analysis and debugging of these applications. To address this issue, Shang et al. [37] proposed a light-weight approach based on execution logs from both the test cloud and the real cloud for assisting the deployment of big data analytics applications in large-scale cloud environments. The on-demand and scalable

[1] BlastReduce. http://www.cbcb.umd.edu/software/blastreduce/.

features of cloud computing make it a platform of choice for big data analytics applications [38]. Running big data applications on a cloud-based Hadoop system can simplify the operations on the client side and more importantly, reduce execution cost. Thus, it has already been a trend to deploy Hadoop-based big data analytics applications (e.g., healthcare big data systems [39]) on clouds.

The advance in data acquisition and discovery makes the demands for processing big data grow rapidly in nearly every field of research. This leads to the ever-growing number of big data applications running on Hadoop providing storage and analytics to support Internet of Things (IoT) [40], business intelligence [41], data mining in social network, etc. [42]. However, the increasingly wide adoption of Hadoop makes its energy inefficiency a prominent concern. As Hadoop was originally designed for generic big data analytics on commodity clusters, little consideration of its efficiency in energy usage is taken. This directly brings about an enormous waste of electricity as hundreds of thousands of data analytical applications are running on Hadoop.

4. Optimization of energy consumption in Hadoop

4.1. Energy-efficient worker node management

Dynamic node management is a common measure to control cluster power consumption in both homogeneous and heterogeneous environments. To dynamically manage worker nodes, a set of policies work at the hardware level, including shutting down servers, turning servers into low-power states and adjusting CPU performance dynamically (e.g., DFS and DVFS). Wirtz and Ge [24] studied through experiments how the number of active workers and dynamic CPU frequency scaling technique impact on the execution time, energy consumption and energy efficiency of MapReduce jobs. They abstract a cluster object into a triple: <number of concurrent workers, number of cores per node, frequency of cores>. The result shows a promotion on energy efficiency in task execution when the cluster was scaled up and DVFS was adopted. As it is illustrated in the paper, clusters need to be dynamically reconfigured according to the change of workload.

According to our survey, energy efficiency can be attained through cluster partitioning and CPU performance scaling. Leverich and Kozyrakis [43] proposed "covering subset" on the basis of Hadoop. More specifically, they refined the default data placement policy of HDFS with a basic principle that at least one data replica is stored in the covering subset. This revised policy guarantees that all the blocks of data in the cluster are accessible even when all machines outside of the subset are down. More importantly, a covering subset is essentially a minimum set of servers for the submitted jobs, which means that we can save energy (at the potential cost of performance) by powering off the servers not belonging to the covering subset. Alternatively, those "unnecessary" workers can be set standby or made sleep. The selection of covering subset plays a vital role in balancing the energy consumption and performance of Hadoop. But the authors did not provide any specific strategy for building a covering subset. Moreover, we suppose that Hadoop's energy efficiency can be further improved with adaptive adjustment of the subset. A covering subset searching algorithm named PACS was proposed in the work of [44]. The algorithm takes into account the heterogeneity of workers and gives priority to those with high power efficiency when selecting nodes into the covering subset. In addition, the authors also present a k-covering subset discovering algorithm to increase data availability. A limitation of PACS is that the computing power of workers is neglected.

Adopting a different approach from retaining a subset, Kaushik and Bhandarkar [45] proposed a partitioning strategy in which all the active Hadoop worker nodes are put into two sets: hot zone and cold zone. Frequently-accessed blocks in HDFS are stored in the hot zone, which mainly consists of high-performance machines with CPUs running at highest frequency. On the contrary, the cold zone is made up of servers attached to large-capacity storage devices and maintains cold (i.e., rarely accessed) data. Workers typically sleep or stand by when they are placed in the cold zone. It is suggested that the hot zone should keep at least 70% of the total number of servers. Data blocks will be exchanged between the zones as the hotness of data changes over time. Actually it is a kind of logical partitioning focusing on consolidating workload and hot data on a few workers. However, the cold zone does not chunk data, which reduces wakeups of servers but may lead to data loss. Moreover, the strategy does not take into account heterogeneity in power efficiency. For instance, energy consumption would increase if servers with low power efficiency are placed in the hot zone. Similarly, Chen et al. [25] improved cluster partitioning by adopting a small-scale interactive zone incorporating high-performance worker nodes. Interactive zone is dedicated to handle jobs with strict response time constraints. On the other hand, batch and preemptive jobs will be allocated to another server set, which is called batch zone. A large number of nodes are kept in the batch zone but workload is consolidated onto a few of them with the rest running in low-power states. This scheme, called BEEMR, helps to reduce the average response time for interactive jobs and save energy by sacrificing partial performance of batch jobs. Experimental result shows that BEEMR outperformed the schemes of [43] and [45] with little influence on write bandwidth or memory usage. However, BEEMR determines the scale of interactive zone statically. It does not consider the change of the data hotness, either.

CPU performance scaling has proved to be effective in energy conservation. Li et al. [46] analyzed the impact of temperature and CPU frequency on system power. They found that the rise of temperature caused the increase in power consumption under the same level of workload. Based on this effect, they proposed a power model as well as a CPU frequency scaling algorithm, which dynamically scales the frequency of CPU according the predicted power value in order to perform power capping. However, it does not take into account the workload of servers when making the scaling decisions. This is the drawback of this work because in highly utilized servers, restriction on CPU frequency may lead to overload and the failure of task execution. Dynamic Voltage and Frequency Scaling (DVFS) has been widely adopted on modern service infrastructures. DVFS allows the CPU frequency of Hadoop workers to adapt to current workload. When the server is under mild workload or idle, CPU performance will be tuned to the lowest level, which achieves effective saving of energy. DVFS has various implementations (e.g., CPUfreq Governors [47] on Redhat Linux) and works according to pre-defined policies. Ibrahim et al. [48] evaluated five distinct governors including *performance*, *powersave*, *ondemand*, *conservative* and *userspace*. They investigated the impacts of different governors on different jobs (*Pi*, *Grep* and *sort*) running on a Hadoop cluster. They observed that the change in execution time and energy caused by the DVFS governors might be inconsistent with what they are designed for. For instance, more energy consumption was incurred by employing *powersave* governor in the execution of CPU-intensive jobs like Pi. Thus, it is impossible to generate the optimal outcome in both performance and energy consumption from any single governor. Instead, we have to face a trade-off when choosing a governor under a specific type of workload. Likewise, the work of [21] also exploited DVFS in Hadoop worker node management through experiments. The authors further analyze the sensitivity of *ondemand* and conservative governors to the parameters including sampling interval, upper bound and lower bound. They found that the setting of parameters has a significant impact on the performance of governors. For example, higher performance and lower energy consumption can be

Table 1
Summary of the studies on energy-efficient worker node management in Hadoop.

Reference	Scheme	Advantages	Limitations
[25]	Interactive/ batch zone	(1) Jobs are classified. (2) The system makes fast respond for interactive jobs. (3) It supports workload consolidation in the Batch zone.	(1) It only supports a static scale of interactive zone. (2) The change of data hotness is not considered.
[43]	Covering subset	(1) The subset can be built by simply redistributing data blocks. (2) It indicates the minimum number of active nodes.	(1) Discovering or adjusting algorithm for covering subset is not provided.
[44]	Power aware covering subset	(1) It considers node heterogeneity. (2) Low-power nodes are picked first. (3) k-covering subset offers high data availability.	(1) The study neglects the performance of workers.
[45]	Hot/cold zone	(1) Cluster is partitioned by data access frequency and data exchange is allowed. (2) Performance is warranted in the hot zone. (3) Energy consumption is reduced as workers in the cold zone typically sleep or stand-by.	(1) The heterogeneity of nodes in power efficiency is not considered. (2) No data chunking in the cold zone may cause data unavailability.
[46]	TAPA	(1) Temperature and CPU frequency are considered. (2) Server power, temperature and CPU frequency are associated. (3) It adopts predictive cluster power capping.	(1) Server workload is not taken into account.
[21,48]	DVFS	(1) It significantly saves server energy by dynamically scaling CPU frequency according to workload. (2) Flexible switching between governors is allowed.	(1) CPU governors are very sensitive to workload and parameters.

achieved by setting a proper lower upper bound of the *ondemand* governor. Table 1 summarizes the studies on optimizing Hadoop worker node management in energy efficiency.

4.2. Energy-efficient data management

HDFS stores input data, immediate results and output data of MapReduce jobs. A typical data-intensive job may produce massive read/write operations on HDFS. According to statistics, data movements (e.g., read, write and shuffle) in MapReduce jobs account for over 10% of the total power consumption of a single node [49]. Energy optimization of Hadoop data management mainly focuses on reducing the total cost of data transfer by selectively placing data blocks on proper data nodes. Meanwhile, factors such as storage device, block size and input data size also affect the throughput of HDFS, which is also vital for achieving high energy efficiency in a Hadoop cluster.

The experiments by Malik et al. [50] revealed how configurations at the system level and machine level influence the performance, power consumption and power efficiency of a Hadoop cluster. The experimental result leads to a conclusion that increasing block size helps to boost both performance and power efficiency. For instance, the optimal setting of block size is 512 MB for *Sort* and *Terasort* [50]. However, block size is usually determined empirically while reconfiguring the block size of HDFS usually requires reformatting.

An important feature of Hadoop is scalability. Though HDFS can adapt to dynamic scaling of the cluster, data distribution may become imbalanced when new workers are added constantly. This causes a large difference between the workers in terms of disk utilization, which consequently increases the energy cost of data access and transfer before job execution. Hadoop incorporates a program named Balancer, which is designed for redistributing data blocks on HDFS. However, the default rebalance mechanism must be refined in order to achieve energy-efficient data management. Maheshwari et al. [51] proposed an elastic Hadoop framework and with a data rebalancing algorithm. When average utilization is too high, cluster scaling operation will be triggered automatically. After new nodes are added and set active, the rebalancing algorithm will check the disk workload on every worker, spot the nodes with high workload and finally shift some data blocks from them onto those new comers. Specifically, intra-rack data movement is conducted with higher priority than inter-rack data movement. In a scale-down occasion, the algorithm can also maintain the workload of every server and average workload of racks at a moderate level by shifting data from the nodes to remove to the rest of the servers. The algorithm has two parameters, namely node workload threshold and cluster average workload threshold. They can be configured flexibly but the constant fluctuation of workload may trigger rebalancing frequently and unnecessarily.

The default data placement policy of HDFS is rack-aware but it presumes a homogeneous cluster. Actually, the default setting of replication and data placement probably lead to imbalance of workload and waste of disk space in a heterogeneous environment. To address this issue, Xiong et al. [52] introduced a novel data placement strategy — SLDP. SLDP takes into account the hotness of data blocks. First, the cluster is divided by performance into Virtual Storage Tiers (VSTs). Meanwhile, the system determines optimal number of replicas for every block according to its hotness (access frequency) on HDFS. After sorting the nodes and data blocks in each VST, SLDP adopts an algorithm to distribute data blocks circuitously with a basic idea of matching hot data blocks with high-performance Datanodes. The performance of a Datanode is measured in IOPS. SLDP considers both server heterogeneity and data hotness and tends to balance the workload in VSTs using a data block placing algorithm. As a result, system energy efficiency is improved as the number of active nodes can be scaled dynamically after SLDP determines a data placement scheme. SLDP takes full advantage of high-performance workers but neglects their power efficiency.

Some researchers tried to exploit high-performance storage devices to optimize the process of MapReduce in energy. Moon et al. [53] used Solid State Drives (SSD) to store data in HDFS and explored its impact on the performance and energy consumption of Hadoop. The result of running Terasort shows that even though adopting SSD throughout all the phases of MapReduce can maximize I/O performance, the optimal scheme in energy efficiency is SSD+HDD, in which Hard Disk Drives (HDD) still serve as the main storage device while SSD store intermediate data. This scheme mainly reduces job execution time and energy consumption in Shuffle phase. But due to the high cost of SSD and limited network bandwidth, it may not be wise to have every machine equipped with an SSD especially when the jobs are basically not IO-intensive. In the framework of MapReduce, intermediate results (the output of Mappers) are exchanged in Shuffle phase, which brings about intensive I/O operations and resource contention as a consequence. Yu et al. [54] proposed a Virtual Shuffle strategy aiming at saving task execution energy by reducing disk accesses in Shuffle phase. Unlike the default mechanism of Hadoop, the strategy first generates and maintains a segment table instead of exchanging data immediately. The table records the locations of all Map output segments and the data transfer only happens when a Reduce task needs to fetch a data segment. In order to avoid exhaustion of memory caused by the merge of virtual segments and the multi-level segment table, the authors make use of balanced subtrees to save memory space. The scheme effectively eliminates redundant data transfer over the network and consequently improves MapReduce's efficiency. However, virtual segment indices will probably occupy huge memory space when terabytes or petabytes of data is involved. Besides, the proposed three-level segment table may lead to increasing data access latency.

MapReduce framework is capable of handling batch jobs that have a large number of input files, but the Namenode is a probable bottleneck as it stores the metadata of every block. Therefore, replacing those small input files with a smaller number of big files can alleviate the load on Namenode and further promote performance [55]. Similarly, Chen et al. [56] proposed a method for merging small files (i.e., file size <HDFS block size) in a HDFS directory. They also added caches on the Datanodes to further speed up data access and reduce energy consumption. A limitation of the study is that the single level global index table of original small files may expand dramatically. Besides, security issue brought about by file merging is not taken into account. The study of [57] makes use of Hadoop Archive (HAR) to merge small files into a big one. Actually, the authors revised the original two-level index structure (masterIndex + index) of HAR and proposed a novel archive file structure — NHAR, which supports single-level index, adding new files (into the archive) and reorganization. Experimental result shows the effectiveness of NHAR file merging in reducing Namenode's workload and memory usage. It is worthwhile to mention that overall data movement cost and energy consumption are reduced only when the proportion of valid data in the NHAR is high because useless data blocks may also be archived. File access contention and data security are not considered in the study. Table 2 summarizes the studies on optimizing Hadoop data management.

4.3. Energy-efficient resource allocation

It is the resource allocation module of Hadoop that enables the MapReduce developers to focus on the job logic without the need of paying too much attention to the complexity of infrastructure such as server heterogeneity. In Hadoop 1.0, JobTracker running on the master node is in charge of both resource provisioning and job scheduling. The granularity of resource allocation is slot (Map/Reduce slot). In Hadoop 2.0, YARN takes the full responsibility of resource management, and the resource provisioning granularity is no longer the slot but the container, which represents a basic resource unit allocated to a job. Resource allocation in Hadoop incorporates multiple aspects such as resource partitioning, resource provisioning and job scheduling. Energy-efficient resource allocation aims at improving resource utilization and energy efficiency without violating SLA (Service Level Agreement) of jobs.

For Hadoop 0.21.0, Tian and Chen [58] proposed a model for predicting job execution time after investigating the relationship among the complexity of a MapReduce process, system resources available and the scale of input data. The model is a function of three variables: total number of Map tasks (the number of input blocks), total number of Reduce tasks and the number of Map slots. After being trained on a small number of nodes, the model can predict job execution time with the amount of resources allocated to the given job. The predictive information helps reduce the resource share (i.e., to reduce the number of Map slots and Reduce slots) and therefore save the energy within the constraint of response time. The predictive results of the model, however, may be inaccurate for new and unknown jobs beyond the training dataset. MapReduce jobs may have different resource demand and execution time. Given this fact, Cardosa et al. [59] proposed an algorithm for provisioning MapReduce applications with resources in virtualized cloud environments. In their study, the goal of reducing job energy consumption is transformed into a problem of minimizing cumulative machine uptime. They proposed a two-step VM allocation algorithm. First, VMs are grouped together according to their estimated runtimes, i.e., each group (called "bin" by the authors) is made up of virtual machines with similar runtimes. In the second step, VMs in the same bin are allocated to servers one by one. The simulation result shows this resource provisioning strategy improves cluster energy efficiency by 20%–35%. The highlight of the strategy is that it resolves a unique spatio-temporal tradeoff over the utilization of physical servers with an efficient heuristic based on "VM fitting recipes" and "VM binning". However, server heterogeneity, which is common in a cloud environment, is not taken into account in their work.

In general, virtualization significantly increases resource utilization, but it also leads to performance degradation [60]. Sharma et al. [61] presented a two-phase resource allocation strategy developed for a more complex circumstance which contains both baremetal machines and VMs. In the first phase, the scheduler decides whether a job should be delivered to the baremetal cluster or the virtual cluster according to its estimated run time and deadline. Then in the second phase, the tasks are allocated by considering resource utilization and contention. The strategy takes advantage of both physical machines and virtual machines to guarantee the response time of interactive jobs while maximizing resource utilization and reducing job energy consumption by consolidating batch jobs. The strategy relies on a job pre-execution mechanism for estimating run time, which will probably increase the latency of response when a number of unknown jobs arrive. Similarly, the study of [62] paid attention to virtual clusters and introduced Cura, a novel MapReduce service framework. Cura adopts a job profiler to analyze the resource demand of jobs without requiring the information from users. It also employs an online, VM-aware scheduler to identify cost-optimal plans for resource provisioning. The virtual cluster is first divided into different resource pools. In each pool, virtual machines are homogeneous. By checking a job's resource demand, the scheduler decides on priority, target resource pool and optimal number of VMs for the job. Experimental result shows that Cura effectively reduces the total consumption of resource under the SLA constraint, which mainly benefits from grouping compute resource (i.e., VMs) and rearranging the job queue. But actually it is of high time complexity to compare the energy impact (on the cluster) of swapping every two jobs in the queue because

Table 2
Summary of studies on energy-efficient data management in Hadoop.

Reference	Scheme	Advantages	Limitations
[50]	Configuring block size	(1) Block size can be conveniently specified in the configuration file. (2) Increasing block size properly can achieve remarkable promotion of performance.	(1) Block size is a static parameter. (2) Reconfiguring block size without reformatting may cause instability of DFS.
[51]	Rebalancing	(1) Disk workload is balanced on Datanodes. (2) Algorithm parameters are Tunable. (3) It supports Power-aware cluster rescaling.	(1) Workload fluctuation may cause unnecessary rebalancing operations.
[52]	SLDP	(1) It considers both server heterogeneity and data hotness. (2) Data hotness-aware replication. (3) Workload in VST is balanced after cluster scaling	(1) Node power efficiency is not considered.
[53]	SSD+HDD	(1) It Adopts SSD to store intermediate results. (2) Energy consumption is reduced in shuffle phase.	(1) It is of too expansive to install an SSD on every worker. (2) Local I/O is not always the bottleneck.
[54]	Virtual Shuffle	(1) Multi-level segment table is adopted. (2) Segments are merged in memory in advance. (3) Data access in Shuffle phase is reduced.	(1) Memory may be crammed with virtual segment indices. (2) Multi-level segment table may be inefficiency for data access.
[56]	File merging +Datanode cache	(1) The number of input files for Map phase is reduced. (2) Distributed cache accelerates data access and reduces energy consumption.	(1) The single-level global index table may expand too fast. (2) Data security is not considered.
[57]	NHAR	(1) Single-level index is adopted. (2) It supports adding new files to NHAR and reorganization. (3) Workload and memory usage are reduced on Namenode.	(1) They do not consider file access contention and data security.

the estimate of energy consumption is further associated with task assignment after the job is scheduled. Thus the scheduler probably becomes a bottleneck in case job arrive rate is high.

Focusing on job scheduling, Cai et al. [63] revised the framework of YARN and proposed an energy-aware resource allocation scheme. The minimum number of concurrent Map or Reduce tasks is first estimated by job profiling before the scheduler determines the resource share of a job according to its SLA constraint. More specifically, Jobs are scheduled based on the EDF (Earliest Deadline First) principle while the NodeManager is allowed to enable DVFS depending on the amount of available resource (i.e., containers) in the cluster. Job characteristics are obtained by a specific job profiler, which increases the response time when the job has to be pre-executed. Moreover, an obvious limitation of EDF scheduling is that job execution time and priority are neglected. Niu et al. [64] also implemented an energy-saving scheduling framework — GreenMR on top of Hadoop YARN. The authors first introduce the execution time models and power consumption models for Map phase and Reduce phase, separately. The models are used to profile the total run time and energy consumption of a job in a given resource configuration. Based on the job profiling results, the scheduler first generates a basic energy-efficient plan according to the cluster capacity, after which a series of optimizations on the plan are conducted to reduce brown (non-renewable) energy consumption. For example, a job may be delayed by a few time slots until sufficient amount of renewable energy supply is available. GreenMR attempts to maximize resource utilization and match the cluster workload with renewable energy supply. But frequent switching may occur between batteries and the grid when green energy is in short supply. Besides, the single FIFO job queue adopted in the framework can be refined to reduce the extra cost of job reallocation. Cluster heterogeneity in power efficiency needs to be taken into account for the implementation of an energy-efficient job scheduler. On this point, Krish et al. [65] proposed a scheduler for MapReduce jobs in heterogeneous clusters, where the workers are partitioned into different sub-clusters. When a job is dequeued, the scheduler estimates the energy efficiency of its execution on every sub-cluster and assigns the job to the optimal one. Then the workload of the sub-cluster is updated. The scheduler makes energy efficiency aware resource provision, but neglects the fact that the cluster's energy efficiency will not remain unchanged as workload increases or decreases. The effectiveness may be further improved if they adopt an adaptive job queue, which can maximize the utilization of those sub-clusters with high energy efficiency by delaying some jobs. Table 3 lists and summarizes the strong points and shortcomings of the relevant studies.

4.4. Energy-efficient task scheduling

A MapReduce job will be decomposed into Map tasks and Reduce tasks according to the amount of input data and system parameters such as HDFS block size. All the Map tasks and Reduce tasks are assigned to worker nodes (physical machines and virtual machines) by the task scheduler. In Hadoop 1.0, the function of task scheduling is incorporated in JobTracker, and the corresponding abstract class is *TaskScheduler*. In Hadoop 2.0 with the YARN framework, ResourceManager is responsible for scheduling the tasks after checking the resource application from *ApplicationMaster* and the available containers in the cluster. Due to the variety of task resource need and server heterogeneity, the allocation strategy has a significant impact on the energy efficiency of task execution.

Table 3
Summary of studies on energy-efficient resource management in Hadoop.

Reference	Scheme	Advantages	Limitations
[58]	Job prediction	(1) It associates the concurrency of Map/Reduce tasks with resource provided and input data scale.	(1) It requires training data. (2) Server heterogeneity and workload are not considered.
[59]	VM placement and grouping	(1) The scheme improves resource utilization both spatially and temporally. (2) It consolidates MapReduce workloads through VM grouping and placement.	(1) Server heterogeneity is not taken into account. (2) Server performance degradation caused by VM contention is neglected.
[61]	Job classification	(1) It works in clusters that consist of both baremetal and virtual machines. (2) Interactive jobs have quick response. (3) Consolidating batch jobs improves energy efficiency.	(1) Pre-execution is needed for job profiling. (2) Job queue optimization is not considered.
[62]	Online VM-aware scheduling	(1) The virtual cluster is divided into pools. (2) It supports flexible job queue.	(1) Determining the best sequence of job execution is of time complexity. (2) The scheduler may become a bottleneck.
[63]	EDF scheduling + deadline-aware DVFS	(1) Minimum concurrency of Map/Reduce tasks is estimated. (2) EDF helps to prevent SLA violations. (3) NodeManager is allowed to enable DVFS to save energy	(1) The scheme does not take into account job run time and priority. (2) Flexible consolidation of tasks is limited by EDF
[64]	GreenMR	(1) Execution time and power consumption models are provided. (2) It allows delay scheduling to maximize the usage of renewable energy supply.	(1) Frequent switching between sources may occur when there is a shortage of green energy. (2) The single FIFO queue cannot efficiently support delay scheduling.
[65]	Energy efficiency aware job assignment	(1) The cluster is partitioned into sub-clusters by hardware configuration. (2) The scheduler matches Jobs and sub-clusters by energy efficiency.	(1) Workload is not taken into account. (2) The job queue is not adaptive. (3) Jobs are scheduled to every sub-cluster to obtain power usage.

Shi and Srivastava [66] proposed a heuristic task scheduling strategy with the focus on the efficiency of cooling system. In the paper, task scheduling is defined as a constrained optimization of the Coefficient of Performance (COP) of the cooling system. They discussed two different cases in which tasks have same or different deadlines, formulated the optimization as an Integer Linear Program and proposed a heuristic allocation algorithm to solve it. Based on minimum cost flow, the algorithm dynamically reschedules tasks to optimize the plans. A limitation of the study is that it ignores server heterogeneity and assumes all tasks are the same in running time. In a virtualized environment, the process of job starts with assigning virtual machines to the Map tasks to be launched. Hwang and Kim [67] proposed and evaluated two cost-aware VM provisioning algorithms through experiments. One is based on List and First-Fit (LFF) principle and the other one is based on Deadline-aware Tasks Packing (DTP). In the paper, cost of a VM is defined as the product of its running cost (predefined) and running time. The LFF-based algorithm uses a list (VPList) to store the running cost of every VM in ascending order and attempts to allocate the Map or Reduce task to the VM at minimum cost. The allocation plan has to be evaluated before execution to guarantee that the constraint of overall time is not violated. The DTP-based algorithm first estimates the execution time of Map phase and adopts a similar way in selecting VM. But it allows assigning multiple tasks to a single VM. The simulation result on CloudSim [57,58] demonstrates that the DTP-based algorithm performed the best in reducing average job execution cost.

Power capping is applied to Hadoop clusters for mitigating overconsumption of energy. Zhu et al. [68] examined the power consumption of a Hadoop cluster in a scale of 200 nodes. They found that the power consumption peaks from time to time, which causes unnecessary scale-ups of the cluster and results in power inefficiency. As they explain in their paper, power peak problem is a consequence of adopting the First Come First Served (FCFS) scheduler. In this case, jobs are scheduled one by one in the order of their arrival time, which means that the tasks of a job can be assigned and executed only when all the former jobs have been scheduled. To mitigate power peaks, the authors proposed a predictive server power model as well as a power-aware task scheduling strategy. The predictive model calculates system power at moment t as a weighted summation of task concurrency and the measured power at moment $(t-1)$. As the relationship between power and concurrency changes over time, the authors adopt a power tracking module and recomputed the model parameters at each control period. The strategy realizes power capping by dynamically restricting the number of concurrent tasks. Similarly, Nghiem and Figueira [69] also noticed that the default mechanism for calculating the number of tasks to launch in Hadoop is likely to cause a waste of resource. To this end, they proposed an algorithm to determine the optimal number of concurrent tasks for *Teragen* and *Terasort*. By running the jobs on a homogeneous cluster with different numbers of tasks and different sizes of input data (10 GB, 100 GB and 1 TB), they collected the data and fitted a job execution time model. Based on the model and the scale of input data, the algorithm is able to calculate the optimal number of tasks to launch. The experiment result shows that the algorithm provides the most accurate estimate and thus the energy consumption of Hadoop is effectively reduced. But the accuracy of the model may

be limited to only a few types of jobs and requires a lot of historical data for training. Another limitation is that the study does not consider server heterogeneity.

Optimizing task scheduling is non-trivial because it can be regarded as a "bin-packing" problem with more constraints. Mashayekhy et al. [70] considered both task dependency and time constraints in their study. They used Integer Programming to model the problem of energy-aware scheduling of a MapReduce job. Since the time complexity to find the optimal solution is pretty high, the authors further proposed EMRSA-X, a heuristic task scheduling algorithm, to solve the problem. First, the algorithm sorts all Map slots and Reduce slots by energy consumption rate, which is defined as the ratio of energy consumption to processing time. Then the longest Map task is picked first and allocated to the node with lowest energy consumption rate if its execution time will not exceed the estimated limit. The deadline of the following Reduce task is updated accordingly. In case there is time remaining before the deadline, the shortest Map or Reduce task will be selected and assigned to this node. EMRSA-I and EMRSA-II are two variants of EMRSA-X while they mainly differ from each other on the definition of energy consumption rate. As a strong point, scheduling long tasks first helps to predict the execution time of Map phase and Reduce phase in advance to effectively avoid missing the job's deadline. The algorithm takes advantage of server heterogeneity in power efficiency. However, it is time-consuming to calculate the energy consumption rate of a slot since we have to examine every task that can be assigned to it. The complexity brings extra cost to task profiling and scheduling. EMRSA-X gives different weights of priority to Map tasks and Reduce tasks of a job according to the ratio of their estimated execution time. However, the study neglects a fact that the ratio changes during the execution of the job.

Physical machines differ a lot in power behaviors. Yigitbasi et al. [26] ran MapReduce jobs on two different clusters consisting of low-power machines and high-performance machines, respectively. Both of the clusters are homogeneous. After testing different types of workload including CPU-intensive jobs and I/O-intensive jobs, they found a great difference in energy efficiency between the two clusters. To this point, they proposed an energy-efficiency greedy scheduler — EESched. When a Map task is to be scheduled, the scheduler prioritizes the worker nodes by their power efficiency under CPU-intensive workload. And for a Reduce task, the scheduler tends to allocate it to a node with high power efficiency (measured in IOPS/Watt) under I/O-intensive workload. EESched also takes data locality into account, but the paper does not mention how to deal with the trade-off between energy efficiency and data locality. As a matter of fact, the embedded FIFO scheduler in Hadoop cannot well guarantee data locality. To this end, Althebyan et al. [71] proposed a multi-threading improvement named MTL scheduler. They partitioned the cluster into several areas while each area incorporates a number of nodes. For a Map task, the MTL scheduler launches multiple threads to search different areas in a parallel manner for the nodes that store input data blocks. When all the required blocks are found, other threads will be notified to serve the next task. As illustrated in [72], MTL scheduler is more effective in reducing job execution time and energy consumption than FIFO scheduler, MatchMaking scheduler and Delay scheduler. On one hand, MTL scheduler puts data locality in the first place. On the other hand, it enables fast scheduling using multiple threads. But the scheme does not consider server heterogeneity and job priority. Wen [73] built energy consumption models for data transfer and task execution, separately. Based on the relationship between job arrival rate, number of active hosts and cluster energy consumption observed in the experiment, the author proposed a two-phase scheme named MinMaxDyn. According to this scheme, the scheduler first activates or deactivates a number of nodes depending on resource demand of the target job and cluster workload at the moment. Next, a heuristic algorithm is adopted to accomplish task scheduling, which means that the scheduler will try to match the most energy-consuming tasks with the servers under the lightest workload. MinMaxDyn considers the energy consumption produced by task execution as well as that by traffic load. The author also adopted a threshold-based approach to save energy by scaling the cluster dynamically. But data locality is not considered in this scheme. Table 4 summarizes the strong points and limitations of the above-mentioned studies on task scheduling in Hadoop.

4.5. Other energy-saving schemes

In previous subsections, we survey the studies on the optimization of Hadoop in energy efficiency with focuses on energy-aware worker node management, data management, resource allocation and task scheduling. Apart from them, many other approaches are also applicable for improving the energy efficiency of Hadoop.

It has been shown through experiments that the amount of energy consumed by a job is closely related to the size of input data [60]. More specifically, larger scale of input data results in more data blocks and consequently, increasing numbers of Map tasks and Reduce tasks. Hence, job energy consumption can be reduced by shrinking the size of input data. Based on this idea, Goiri et al. [74] proposed a special Hadoop framework named ApproxHadoop, which mainly adopts two mechanisms: input data sampling and task dropping. Data sampling is a kind of preprocessing on the original input dataset, which will be sampled into a subset according to some specific rules. Actually, data sampling is a multi-phase process. As an example, we need two phases to preprocess a job which counts the number of 'a' in a dataset consisting of N web pages divided into M data blocks. In the first phase, m ($m<M$) blocks are picked out and allocated to Map tasks with the rest of them discarded. Next, data is sampled again in each data block by the corresponding Map task, which means that a portion of web pages in every block are abandoned. Task dropping is more complicated. The authors adopted different strategies to drop tasks in different types of jobs (i.e., computing aggregation and extreme values). Error bound estimate is also derived for each case. Essentially, task dropping aims at reducing computation by discarding some intermediate results (output of Map tasks) once the desired error bound has been achieved after the completion of former Mappers. ApproxHadoop provides us with a novel method to effectively improve system performance and reduce energy consumption by sacrificing the precision of results. The experimental result of the study shows that job run time and energy consumption are significantly reduced. The error caused by data sampling and task dropping is acceptable when the input data is large in size and uniformly distributed, but issues like missing keys may appear.

Some modern processors support independent frequency scaling of CPU cores. In other words, the cores on a multi-core processor can have different frequency, performance and power efficiency. Yan et al. [75] exploited the heterogeneity of multi-core processors and proposed to divide the whole cluster into virtual resource pools. The vFast pool contains high-performance cores while the vSlow pool contains "slow" cores which usually outnumber "fast" cores. They adopted a multi-class priority scheduling policy by which interactive jobs and batch jobs are assigned to vFast pool and vSlow pool, respectively. In addition, a shared pool is adopted to retain idle core resource in order to prevent starvation of jobs in either queue. Interactive jobs generally demand fewer Map/Reduce slots while batch jobs often generate a large number of concurrent Map tasks and Reduce tasks. Experimental result shows that average job execution time and processor power consumption are optimized by matching different jobs with different

Table 4
Summary of studies on energy-efficient task scheduling schemes in Hadoop.

Reference	Scheme	Advantages	Limitations
[66]	Minimum cost flow	(1) Task execution limit is taken into account. (2) It allows rescheduling dynamically. (3) It mainly focuses on saving energy in data centers with huge storage.	(1) The power model barely considers disk. (2) It assumes that all tasks proceed in a same speed. (3) It does not consider server heterogeneity
[67]	Cost-aware VM selecting	(1) Virtualization, heterogeneity and job SLA are considered. (2) The DTP-based algorithm allows assigning multiple tasks to a single VM.	(1) Data locality is not taken into consideration. (2) It neglects the impact of workload on execution cost.
[68]	Power-aware task concurrency control	(1) Its exploits a predictive server power model. (2) Parameters of the power model are constantly calibrated from period to period using power tracks. (3) The number of concurrent tasks is determined dynamically to mitigate cluster power peaks.	(1) Server heterogeneity is not considered.
[26]	EESched	(1) Server heterogeneity is considered when calculating the energy efficiency of Map tasks and Reduce tasks. (2) Data locality is also taken into account.	(1) The study does not figure out how to prioritize power efficiency and data locality. (2) The study neglects workload's influence on energy efficiency.
[69]	Predicting optimal task concurrency	(1) It presents a job execution time model associated to the number of concurrent tasks. (2) The model's accuracy is limited to the training data including only a few types of jobs. (3) Energy consumption can be reduced by running just enough number of tasks.	(1) Training or historical job data is required for the model. (2) Server heterogeneity is not taken into account.
[70]	EMRSA-X	(1) Scheduling long tasks first helps to estimate the execution time of Map and Reduce in advance. (2) Server heterogeneity is considered.	(1) The calculation of energy consumption rate for each slot is of high complexity and may bring about extra cost. (2) It determines the weights of Map and Reduce tasks of a job statically.
[71]	MTL scheduler	(1) Data locality is considered. (2) The scheduler is more efficient by using multiple threads.	(1) Energy efficiency, server heterogeneity and job priority are not considered.
[73]	MinMaxDyn	(1) The scheme considers energy consumed by task execution and data traffic. (2) It adopts threshold-based dynamic cluster scaling to reduce energy consumption.	(1) Data locality is not taken into account.

pools of cores. A remarkable portion of MapReduce jobs are data-intensive, which means that disks usually account for a great amount of energy consumption. To reduce disk power consumption, Qiao and Li [76] introduced a novel disk state management algorithm which exploits the characteristics of disk workload in different phases of a MapReduce job. For instance, they figured out that sometimes only a few I/O operations may happen on disk in Shuffle phase. Therefore, in this case we can set the disk to a low-power state to achieve energy efficiency. But task response time will probably increase when there is a burst in disk I/O since speeding up a disk takes some time. Besides local disk input/output, massive data transfer through data center networks also causes enormous consumption of both time and energy. To this end, Asad et al. [77] proposed a spate coding method for data packet and implemented an optimized Hadoop system using Software Defined Network (SDN). First, data is labeled by key. Then the data packet is encoded before being routed by the aggregate switch. Finally the Reducer uses the local side information to decode the data packet it received. The proposed scheme significantly reduces the volume of inter-rack communication in the phase of Shuffle, and consequently improves the energy efficiency of data transfer. But it is likely that aggregate switches take heavy workload of data encoding and become a bottleneck of performance.

MapReduce is a fault-tolerant framework since many reactive or proactive failure-handling mechanisms are incorporated including task re-execution and speculative execution. Lin et al. [78] explored the effects of Task Re-execution (TR) on job reliability, turnaround time and energy consumption. They found that re-execution significantly improves job reliability as input data scale increases, and reduces job energy consumption by tuning TR factors of Map task and Reduce task. But in practice, it is difficult to determine the optimal TR factors in advance. Different from task re-execution, speculative execution is essentially a strategy to exchange extra resource for time efficiency. Speculative tasks are launched in order to replace those stragglers which lag behind other tasks. A problem is that speculative execution is bound to bring about overheads in energy consumption. On this point, Phan et al. [79] conducted a series of experiments to observe the cost. They discovered that speculative execution increases energy consumption in most cases especially for I/O-intensive jobs. This

is because the method adopted in Hadoop for judging stragglers is too simple. This directly leads to the generation of a number of unnecessary speculative tasks and consequently, causes resource contention and the increase of server idle time. But actually speculative execution can be energy-efficiency if job execution time is remarkably reduced. For example, a local speculative task may be beneficial. Thus it can be concluded that the negative impact on cluster energy efficiency is closely related to false or inaccurate detection of straggling tasks. However, their study only compares two cases (i.e., enabling or disabling speculative execution in Hadoop) and lacks further analysis of the related parameters like the threshold in straggler detection. Also focusing on optimizing the fault-tolerance nature of Hadoop, Lin et al. [80] proposed PARes, an improved framework that can well resolve the single-failure problem of the NameNode. They set up two different nodes (HNode and WNode) as backups for the master. Based on this infrastructure, they proposed a proactive meta-data synchronization, mutual monitoring between masters and a warm-up mechanism (for WNode) to increase system stability at a minimum cost of energy consumption. As a hot standby node, HNode constantly communicated with the master to synchronize meta-data, thus it is able to take over the cluster immediately once the master is down. WNode will starts to warm up at the same time. Compared with traditional stand-by schemes, PAReS effectively shortens service downtime with low energy overheads. But the mutual monitoring mechanism does not consider the workload on master and may increase traffic load.

Nowadays, there is an emerging trend of adopting renewable energy sources (e.g., solar and wind) in data centers. However, green energy supply has natural defects such as poor sustainability and limited supply period. In order to maximize the power of renewable sources, GreenSlot [81] was proposed. GreenSlot is a green energy aware batch job scheduler designed for maximizing the utilization of solar energy supply. Given the supply pattern of solar energy and the price pattern of grid, jobs in the current time slot are scheduled in a greedy manner. First, the scheduler selects the job with the shortest slack time. Then it tries to delay the job without exceeding its deadline until the coming of a time slot in which the execution cost is the lowest (the cost of consuming renewable energy is considered zero). At the meantime, GreenSlot adjusts the prediction of the amount of green energy in future time slots according to the error between estimated value and actual value in the current time slot. GreenSlot will reschedule once the error in current time slot is too large. A problem of this scheme is that rescheduling may be frequent when the job arrival rate is high but the predictive model of green energy is inaccurate. Similarly, GreenHadoop [82] implements solar energy aware job scheduling and data management on the basis of Hadoop framework. GreenHadoop first scales the cluster by examining the amount of solar energy available in current time slot. Then a part of arrived jobs are put into the running queue according to their priority and SLA constraints. And the nodes with required data blocks are activated. If renewable sources are in short supply and the grid energy is expensive in the current time slot, GreenHadoop will deactivate or shut down some nodes depending on the location of the job's data. And data blocks will be moved from retired nodes to those active nodes. This framework shifts workload and relocates data blocks to maximize the utilization of renewable energy through job scheduling and dynamic node management. But the proposed data movement policy does not consider server workload and power efficiency. Table 5 summarizes the strong points and limitations of the studies discussed in this section.

4.6. An overview of the surveyed studies

The energy efficiency of a Hadoop system can be attained through optimizing multiple metrics such as the number of (active) servers, data movement, server utilization, etc. Nevertheless, estimating or predicting system power or cost as a decision indicator is also widely adopted. The second means requires a predefined model with several parameters. Considering the complexity of Hadoop system as a cloud or data analytics platform, researchers focus on different criteria including energy consumption, job makespan and total cost. Actually, the optimization can be multi-objective. Besides, Service Level Agreements (SLAs) need to be guaranteed in scheduling. For example, constraints in resource shares, cost budget as well as job deadline should be met.

In order to provide a complementary view of the approaches we surveyed in our study, we summarize the works in a table (Table 6) indicating the adoption of a) power/energy/cost model(s) b) constraint(s) c) multi-objective optimization d) online decision or e) training.

5. Research insights and future directions

As the ever-growing consumption of energy gradually raises people's concern, how to improve the energy efficiency of Hadoop has become a topic of interest. Although much work has been done for the optimization of Hadoop and MapReduce framework, we are still facing great challenges as the heterogeneity of data centers gets prevailing and the diversity of big data workloads keeps increasing. Thus, it is of great necessity to overcome the limitations of previous studies and find more effective approaches to promoting Hadoop's energy efficiency. In this section, we present our insights from the previous studies and available techniques and discuss future directions of the research on energy-efficient Hadoop systems.

5.1. Energy-efficient cluster partitioning

As the power efficiency of servers vary a lot in heterogeneous data centers, the way we scale the cluster up or down makes a huge impact on Hadoop's energy efficiency. The design of energy-efficient node management strategies is challenging because we have to prevent the unbalance of data distribution, server overload and underutilization while rescaling the cluster to save energy. As a solution, cluster partitioning proved to be effective [43,45]. Cluster partitioning allows different strategies specified for different groups of machines when the cluster needs to be scaled up or scaled down. For example, we typically retain the nodes in the covering subset [43] and shut down other nodes to reduce system energy consumption. The covering subset provides a baseline for minimizing the number of servers while guaranteeing basic service availability of the system. However, cluster partitioning and node management may be inefficient in energy if we do not consider more factors such as power efficiency and I/O performance of workers and data hotness. On one hand, the strategies by which we partition and manage the cluster should be energy-aware. Barely considering data locations and server performance may lead to high energy consumption. We should give higher priority to the worker nodes with high power efficiency and/or SSD. On the other hand, we need to find an effective approach to dynamically rescaling different partitions in the cluster. It is critical to make the partitions adaptive to the change of data hotness since static partitioning probably leads to unbalanced workload and energy inefficiency.

Please cite this article in press as: W. Wu, et al., Energy-efficient hadoop for big data analytics and computing: A systematic review and research insights, Future Generation Computer Systems (2017), https://doi.org/10.1016/j.future.2017.11.010.

Table 5
Summary of studies on other energy-saving schemes.

Reference	Scheme	Advantages	Limitations
[74]	ApproxHadoop	(1) Workload is reduced by sampling input data and dropping tasks. (2) The scheme effectively saves energy within acceptable error bounds.	(1) There exist problems like missing keys and biased results.
[75]	Pooling of CPU cores	(1) Interactive jobs and batch jobs are matched with vFast pool and vSlow pool, respectively. (2) A shared pool is adopted to avoid job starvation.	(1) Memory constraint is neglected in terms of resource provisioning.
[76]	Disk state Management	(1) It takes advantage of the discontinuity of disk I/O in MapReduce jobs.	(1) Response may be delayed when a disk is resuming from the idle state to handle intensive I/O operations.
[77]	Data packet coding	(1) The packet encoding schemes significantly reduces inter-rack traffic load and improves the energy efficiency of communication.	(1) The encoder on the aggregate switch may become a bottleneck.
[78]	Task re-execution	(1) TR makes job execution more reliable. (2) Job energy consumption can be reduced by tuning TR factors and relevant parameters.	(1) It is tough to decide on the best TR factors for different jobs before beforehand.
[79]	Speculative execution	(1) Job run time as well as energy consumption can be reduced.	(1) Launching unnecessary speculative tasks causes a waste of resources. (2) The authors do not analyze the impacts of relevant parameters.
[80]	PAReS	(1) The framework adopts a comprehensive backup solution (HNode+WNode). (2) It effectively improves service availability at a low energy consumption cost.	(1) The synchronization and mutual monitoring mechanism do not consider the workload on master. (2) It is likely to cause heavier traffic load on the master.
[81]	GreenSlot	(1) The heuristic scheduling strategy maximizes the utilization of green energy utilization. (2) Job slack time is considered.	(1) The scheduler may become a bottleneck once the energy prediction model is not accurate.
[82]	GreenHadoop	(1) The framework effectively matches cluster workload with renewable energy supply via job scheduling and dynamic node management.	(1) Data block movement does not take server workload and energy efficiency into account.

Table 6
An overview of the studies.

	Power model(s) / cost function	Constraint(s)	Multi-objective optimization	Online decision	Training
Ref.	[43] (CPU power), [44] (server power), [46] (server power), [52] (cost), [58] (cost), [62] (cost), [64] (energy), [66] (disk & cost), [67] (cost) [68] (server & cost), [69] (energy), [70] (energy), [73] (transmission & server power)	[43] (data availability), [44] (data availability), [46] (power budget), [52] (data availability), [58] (time or budget), [61] (time), [62] (time), [63] (time), [64] (time), [65] (time), [66] (time & thermal), [67] (time), [70] (time), [74] (error), [82] (time)	[66] (energy & heat removal), [69] (processing time & energy), [70] (processing time & energy), [81](green energy usage & cost), [82](green energy usage & cost),	[45,46], [51,59], [62,63], [64,67], [26,68], [69,71], [73,75], [76,81,82] [82]	[46,58], [61,68] [81]

5.2. Data-oriented resource classification and provisioning

A challenge in front of us is how to make resource provisioning more effective while the energy consumption of Hadoop is optimized. Resource classification not only simplifies matching resources and jobs, but also reduces the complexity of task scheduling. For example, we can divide the whole cluster into several pools. But it is non-trivial to determine the optimal classification because of the uncertainty of workloads and unbalanced data access. To consolidate workload and thus reduce energy consumption in a Hadoop system, resources can be classified and provisioned according to power efficiency and the hotness of data blocks. The motivation of data-oriented resource classification and provisioning is to consolidate workload by redistributing data blocks in HDFS. Specifically, we store frequently accessed blocks in a pool of servers and exchange the data blocks with hot ones if they get cold. This helps to sufficiently utilize the workers with high power efficiency. Data hotness is updated on a regular basis

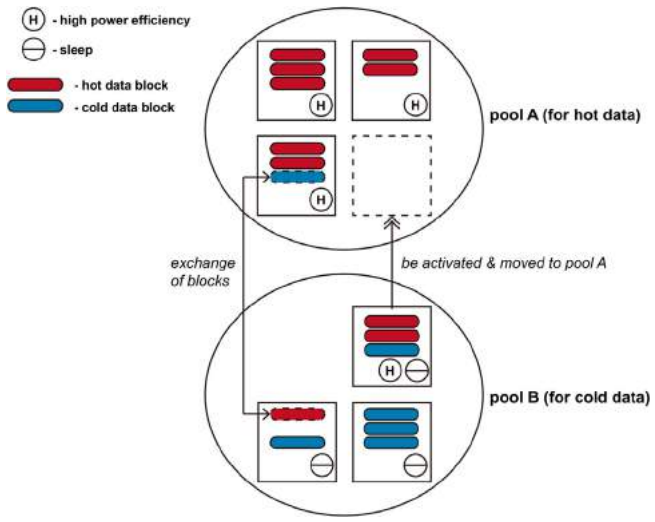

Fig. 2. Data-oriented resource classification and provisioning with multiple pools.

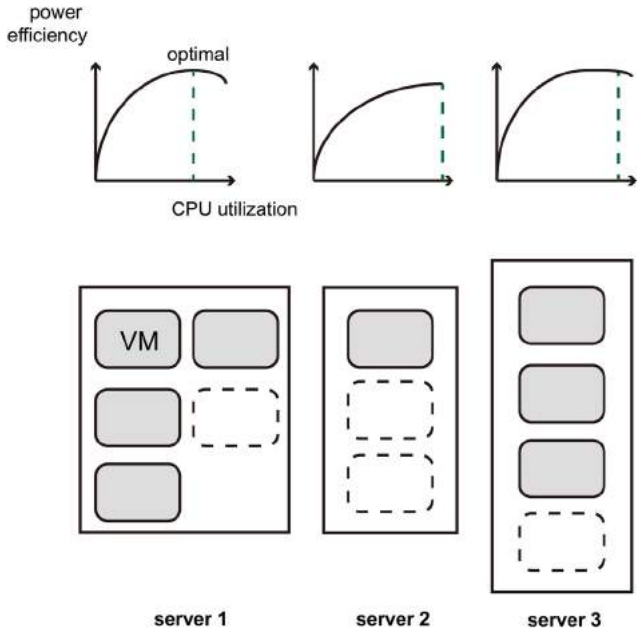

Fig. 3. Servers with different levels of utilization for optimal power efficiency.

and cold data is collected into another pool of servers, which will change to the low-power state to reduce energy consumption. Optimization of data locations is vital for adaptive scaling in both pools. We need to develop effective algorithms for data block redistribution and energy-efficient provisioning strategies for different classes of resources. Fig. 2 shows a simple case in which the resources are classified into two pools. Pool A contains servers storing hot data while pool B is for cold data. Pooling can achieve load balancing by two means. One is data block exchanging, which consolidates frequently-accessed data on a few servers via transmission on network. The other means is repartitioning — sometimes regrouping the servers is cheaper than data movement when the change of data hotness follows a regular pattern. Logical pooling of Hadoop nodes reduces power consumption by powering off servers with only cold data or keeping them in low-power states.

5.3. Resource provisioning based on optimal utilization

Most of the existing resource allocation strategies are job-oriented and do not organize resource in an efficient way. This may lead to extra scheduling overheads and sub-optimal plans. As a proposal, resource management can focus on optimizing the energy efficiency of resources instead of jobs. It has been observed that server power efficiency is maximized when CPU utilization stays at a particular level [83]. Hence, we can determine the optimal CPU utilization for every active node in the cluster by combining its performance curve and power curve. Then we can derive the optimal amount of resource that a server can offer, which can be quantified as the number of slots or containers. This means that the server will change to its most power-efficient state when an optimal number of slots or containers are occupied (Fig. 3). Optimal utilization based resource provisioning is different from but does not violate the common logic of consolidating workload on the fewest number of servers. It takes advantage of the most efficient state of every server whilst avoiding server overload. To provision resources based on their optimal utilization, there are two major challenges: how to organize the resources efficiently and how to maximize energy efficiency in resource allocation. First, the utilization of workers changes frequently while the cluster may scale up/down from time to time. So a structure efficient in search, insert and delete is required to manage the resources. Second, for every job to be scheduled, the system needs to search for an appropriate server or virtual machine that can offer sufficient amount of resources while reaching its optimal utilization. Therefore, we need to develop an effective algorithm for maximizing holistic power efficiency through making more servers stay at their optimal levels of utilization.

5.4. EE and locality aware task scheduling

Energy efficiency (EE) and data locality are critical factors for reducing the energy consumption of MapReduce tasks. But generally we have to face the trade-off in task scheduling between utilizing power-efficient servers and retaining data locality. On one hand, it may not be a good choice if we launch a non-local task even on a server with high power efficiency because copying input data can be of high cost. On the other hand, if we only consider data locality and run the task on a server with low power efficiency, the execution may consume a great amount of energy even though data transfer is avoided. Hence, we first need to build a quantitative model for evaluating the cost of task comprising of both the energy consumed by task execution and data movement. The energy consumption in execution closely relates to task resource demands and server power efficiency whilst the energy consumed by data transfer can be modeled as a function of bandwidth and data size. Besides, time constraint is another important factor to be considered. The response time needs to be estimated before the task is launched on a server in case there is a high risk of SLA violation, which can be regarded as a part of potential cost. Based on the cost model and time constraints, we further need to develop a judicious strategy that enables scheduling a batch of tasks and maximizes the energy efficiency of Hadoop systems.

5.5. Optimizing job profiling with machine learning

A job profile incorporates job-related information that is critical for energy-efficient job scheduling. The Job profiler predicts the execution time, resource demand and workload characteristics of every job arrived. In previous studies, there are mainly two mechanisms we can adopt to realize job profiling: analysis of historical job data and pre-execution. However, they may not be effective

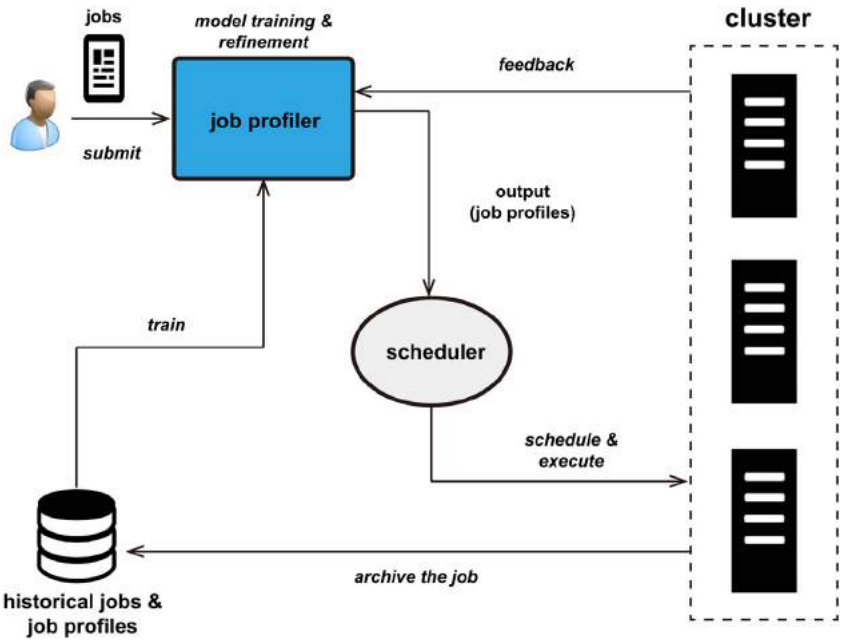

Fig. 4. A framework of job profiling based on machine learning.

or efficient enough to provide accurate job profiles for energy-saving scheduling. Analyzing historical job data can hardly offer the information of unknown jobs without execution records, while job pre-execution inevitably consumes extra resources bringing about remarkable overheads in time and energy consumption especially when job arrival rate is high.

Machine learning can be applied to MapReduce job profiling to avoid or reduce the overheads of time and energy caused by pre-execution. We need to build a job profiling model by which job execution time, resource demand and workload characteristics can be efficiently predicted. The model's input are job features such as the number of virtual cores required, libraries included and the size of input data. With job-related training data from historical jobs and their profiles, the model can be trained using machine learning methods. However, the diversity of jobs submitted to the Hadoop system makes the job profiling model's accuracy a major problem. We need to develop an improved framework of Hadoop that enables the job profiler to "learn" from both historical jobs and new jobs (Fig. 4). Initially, job profiler is trained with a set of historical data (e.g., MapReduce job traces) and may provide inaccurate estimate of profiles when a job is first submitted. During the execution, job-related metrics (e.g., execution time, blocks processed and resource shares) are recorded and will be fed back to the profiler for model refinement using methods like incremental regression. At the same time, measurements of finished jobs are also archived as historical profiles to enrich the training dataset (Fig. 4). As a result, the job profiler is constantly refined and able to provide accurate job information.

5.6. Elastic power-saving Hadoop with containerization

Server virtualization significantly promotes the utilization of physical resource. However, the virtual machine is still too "heavy" for highly elastic clusters. Emerging containerization techniques like Docker provides MapReduce applications with a lightweight solution to enabling the cluster to scale more easily and swiftly. For example, Chen et al. [84] proposed Virtual Hadoop in their study, in which they replaced the basic task execution unit in YARN with Docker container. This allows them to implement an auto-scaling Hadoop cluster. Wang et al. [85] encapsulated and ran the YARN components of ResourceManager, ApplicationMaster and NodeManager in Docker containers. We can take advantage of containerization to reduce Hadoop's energy consumption. First, the density of workers increases. Co-located containers share the kernel of host OS and thus significantly reduce total memory consumption. Consequently, the cost of energy consumption is reduced since we can run an equal number of MapReduce tasks in containers on fewer physical machines. Second, containers can be configured, launched and destroyed more flexibly and swiftly. A container image can be built automatically and it only takes a few seconds to start up or destroy a container. These features of container boost the elasticity of a Hadoop cluster by diminishing the time we need to scale up/down the cluster adaptively to the workload that change constantly. To implement a power-saving Hadoop with containers, the key is to coordinate the scheduling-related components of Hadoop and the container orchestrator. We need to develop a comprehensive framework that enables resource provisioning and task scheduling at container level, and find approaches to leveraging energy-aware workload consolidation on containerized infrastructures.

5.7. Efficient big data analytics on Hadoop

The need of processing big data has been common in recent years. The ecosystem of Hadoop provides data scientists with a systematic approach to designing and implementing an entire workflow of big data analytics. However, most big data analytics applications[38] are extremely time-consuming, which subsequently causes high energy consumption. Therefore, it is critical to optimize big data analytics applications on Hadoop.

Big data applications are typically CPU-intensive and/or IO-intensive, which makes them sensitive to resource contention. More specifically, sharing disk throughput and bandwidth with other MapReduce jobs may lead to unexpected performance degradation. As a solution, running big data analytics jobs with high priority on dedicated servers can avoid resource contention. Huge volume and variety are prominent characteristics of big data. Huge volume of data makes efficient access to records a non-trivia task. Siddiqa et al. [86] experimentally demonstrated the

inefficiency of query execution using Hadoop full scan. It is also necessary to optimize the structure of the data stored on HDFS, HBase and Hive in order to reduce the size of data transferred via network. This can be combined with MapReduce workflow optimization like data compressing and balanced partitioning.

Update of data records may trigger a complete analytical workflow on the whole dataset. This process is inefficient in cost especially when the update is minor. Hadoop can deal with long batch jobs but provides little support for incremental analysis. To this end, we need to optimize Hadoop for some specific big data applications with techniques like reusing intermediate results. Enabling incremental analysis of big data on Hadoop is challenging but represents a promising approach towards reducing the cost of big data analytics applications.

6. Conclusions

As the most popular open source implementation of MapReduce parallel framework, Hadoop provides a generic platform for processing large-scale datasets in a distributed environment. With the ever-growing demand for big data analytics, Hadoop has gained wide adoption in many fields of research and practice including bioinformatics, social network and business intelligence. However, little consideration of energy efficiency is taken in its original design, which usually causes overconsumption of energy especially when the system is running analytical jobs on massive datasets. To this end, there are a great number of studies on reducing the energy consumption of MapReduce applications from different aspects. In this paper, we summarize and compare them in five categories including energy-efficient optimization of cluster node management, data management, resource allocation, task scheduling and other energy-saving schemes. The Optimization of node management is mainly realized via cluster partitioning and server performance scaling. The performance and energy efficiency of Hadoop also closely relate to the data placement policy of HDFS, which can be improved through resizing data blocks, optimizing Shuffle and merging small input files. Recently, energy-efficient resource allocation and task scheduling have become the topics of interest in the research of MapReduce and Hadoop. We can adopt the methods such as resource pooling, job classification and job queue optimization to boost the energy efficiency of Hadoop in resource provisioning and. For task scheduling strategies, they typically require predictive information (e.g., estimated execution time) and assign tasks to worker nodes considering multiple factors like power efficiency (of the nodes), data locality and SLA constraints. In addition, we also survey other state-of-the-art energy-saving schemes such as ApproxHadoop, GreenSlot and data packet coding. For the studies in each category, we elaborate their rationales and analyze their strong points and limitations as well, aiming at providing the users and developers of Hadoop with useful guidance.

Motivated by the previous work, we further illustrate our insights and possible research trends towards improving Hadoop's energy efficiency. Specifically, they are energy-efficient cluster partitioning, data-oriented resource classification and provisioning, resource provisioning based on optimal utilization, EE and locality aware task scheduling, optimizing job profiling with machine learning and elastic power-saving Hadoop with containerization. The ideas we present in this paper may inspire the relevant research on energy-aware Hadoop.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (Grant Nos. 61402183 and 61772205), National Science and Technology Ministry (Grant No. 2015BAK 36B06), Guangdong Provincial Scientific and Technological Projects (Grant Nos. 2017B010126002, 2017A010101008, 2017A01010 1014, 2017B090901061, 2016A010101007, 2016B090918021 and 2014B010117001), Guangzhou Science and Technology Projects (Grant Nos. 201607010048 and 201604010040).

References

[1] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[2] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, H. Jacobsen, Bigbench: Towards an industry standard benchmark for big data analytics, in: ACM SIGMOD International Conference on Management of Data, ACM, 2013, pp. 1197–1208.
[3] V. Chang, An overview, examples, and impacts offered by emerging services and analytics in cloud computing virtual reality, Neural Comput. Appl. (2015) 1–14.
[4] The Apache Hadoop Project. http://www.hadoop.org.
[5] Powered by Hadoop. http://wiki.apache.org/hadoop/PoweredBy.
[6] H. Jin, S. Ibrahim, L. Qi, H. Cao, S. Wu, X. Shi, The mapreduce programming model and implementations, Cloud Comput.: Princ. Paradigms (2011) 373–390.
[7] M. Zaharia, A. Konwinski, A.D. Joseph, R. Katz, I. Stoica, Improving mapreduce performance in heterogeneous environments, in: Usenix Conference on Operating Systems Design and Implementation, USENIX Association, 2008, pp. 29–42.
[8] Y. Chen, A. Ganapathi, R. Griffith, R. Katz, The case for evaluating mapreduce performance using workload suites, in: IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, IEEE, 2011, pp. 390–399.
[9] N. Zhu, L. Rao, X. Liu, J. Liu, H. Guan, Taming power peaks in mapreduce clusters, in: ACM SIGCOMM 2011 Conference, vol. 41, ACM, 2011, pp. 416–417.
[10] J.X. Wu, C.S. Zhang, B. Zhang, P. Wang, A new data-grouping-aware dynamic data placement method that take into account jobs execute frequency for Hadoop, Microprocess. Microsyst. 47 (Part A) (2016) 161–169.
[11] Y. Wang, J. Tan, W. Yu, L. Zhang, X. Meng, X. Li, Preemptive reduce task scheduling for fair and fast job completion, in: 10th International Conference on Autonomic Computing (ICAC '13), 2013, pp. 279–289.
[12] C.H. Hsu, K.D. Slagter, Y.C. Chung, Locality and loading aware virtual machine mapping techniques for optimizing communications in MapReduce applications, Future Gener. Comput. Syst. 53 (2015) 43–54.
[13] G. Ananthanarayanan, C.C. Hung, X. Ren, I. Stoica, A. Wierman, M. Yu, GRASS: Trimming stragglers in approximation analytics, in: Usenix Conference on Networked Systems Design and Implementation, USENIX Association, 2014, pp. 289–302.
[14] Y. Li, Q. Yang, S. Lai, B. Li, A new speculative execution algorithm based on c4.5 decision tree for hadoop, in: International Conference of Young Computer Scientists, Engineers and Educators, vol. 503, Springer, Berlin Heidelberg, 2015, pp. 284–291.
[15] J.A. Quiané-Ruiz, C. Pinkel, J. Schad, J. Dittrich, RAFTing mapreduce: Fast recovery on the raft, in: IEEE International Conference on Data Engineering, vol. 6493, IEEE, 2011, pp. 589–600.
[16] F. Dinu, T.S.E. Ng, Rcmp: Enabling efficient recomputation based failure resilience for big data analytics, in: 2014 IEEE International Parallel & Distributed Processing Symposium, IEEE, 2014, pp. 962–971.
[17] O. Yildiz, S. Ibrahim, G. Antoniu, Enabling fast failure recovery in shared hadoop clusters: Towards failure-aware scheduling, Future Gener. Comput. Syst. 74 (2017) 208–219.
[18] A.M. Sampaio, J.G. Barbosa, Towards high-available and energy-efficient virtual computing environments in the cloud, Future Gener. Comput. Syst. 40 (2014) 30–43.
[19] Y.C. Lee, A.Y. Zomaya, Energy efficient utilization of resources in cloud computing systems, J. Supercomput. 60 (2) (2012) 268–280.
[20] C.T. Yang, Y.Z. Yan, S.T. Chen, R.H. Liu, J.H. Ou, K.L. Chen, iGEMS: A Cloud Green Energy Management System in Data Center, in: Green, Pervasive, and Cloud Computing, Springer International Publishing, 2016, pp. 82–98.
[21] S. Ibrahim, T.D. Phan, A. Carpen-Amarie, H.E. Chihoub, D. Moise, G. Antoniu, Governing energy consumption in hadoop through CPU frequency scaling: An analysis, Future Gener. Comput. Syst. 54 (2016) 219–232.
[22] B.T. Rao, L.S.S. Reddy, Survey on improved scheduling in hadoop mapreduce in cloud environments, Int. J. Comput. Appl. 34 (9) (2012) 28–32.
[23] S. D'Souza, K. Chandrasekaran, Analysis of MapReduce scheduling and its improvements in cloud environment, in: IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems, 2015.
[24] T. Wirtz, R. Ge, Improving mapreduce energy efficiency for computation intensive workloads, in: International Green Computing Conference and Workshops, IEEE Computer Society, 2011, pp. 1–8.

Please cite this article in press as: W. Wu, et al., Energy-efficient hadoop for big data analytics and computing: A systematic review and research insights, Future Generation Computer Systems (2017), https://doi.org/10.1016/j.future.2017.11.010.

[25] Y. Chen, S. Alspaugh, D. Borthakur, R. Katz, Energy efficiency for large-scale mapreduce workloads with significant interactive analysis, in: ACM European Conference on Computer Systems, ACM, 2012, pp. 43–56.
[26] N. Yigitbasi, K. Datta, N. Jain, T. Willke, Energy efficient scheduling of mapreduce workloads on heterogeneous clusters, in: Green Computing Middleware on Proceedings of the 2nd International Workshop, 2011.
[27] X. Wang, Y. Wang, Y. Cui, A new multi-objective bi-level programming model for energy and locality aware multi-job scheduling in cloud computing, Future Gener. Comput. Syst. 36 (7) (2014) 91–101.
[28] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P.P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q.M. Malluhi, N. Tziritas, A. Vishnu, S.U. Khan, A. Zomaya, A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems, Computing 98 (7) (2016) 751–774.
[29] D. Borthakur, The hadoop distributed file system: Architecture and design, Hadoop Proj. Website 11 (2007) 21.
[30] V. Chang, R.J. Walters, G. Wills, Cloud storage and bioinformatics in a private cloud deployment: Lessons for data intensive research, in: International Conference on Cloud Computing and Services Science, vol. 367, Springer International Publishing, 2012, pp. 245–264.
[31] A. O'Driscoll, J. Daugelaite, R.D. Sleator, 'Big Data', hadoop and cloud computing in genomics, J. Biomed. Inform. 46 (5) (2013) 774–781.
[32] T. Nguyen, W. Shi, D. Ruden, Cloudaligner: A fast and full-featured mapreduce based tool for sequence mapping, BMC Res. Notes 4 (2011) 171–177. http://dx.doi.org/10.1186/1756-0500-4-171.
[33] S. Lewis, A. Csordas, S. Killcoyne, H. Hermjako, M.R. Hoopmann, R.L. Moritz, E.W. Deutsch, J. Boyle, Hydra: A scalable proteomic search engine which utilizes the Hadoop distributed computing framework, BMC Bioinformatics. 13 (2012) 324–329. http://dx.doi.org/10.1186/1471-2105-13-324.
[34] V. Chang, Towards data analysis for weather cloud computing, Knowl.-Based Syst. 127 (2017) 29–45.
[35] S. Gao, L. Li, W. Li, K. Janowicz, Y. Zhang, Constructing gazetteers from volunteered big geo-data based on Hadoop, Comput. Environ. Urban Syst. 61 (2017) 172–186.
[36] J. Li, J.W. Li, X.F. Chen, C. Jia, W.J. Lou, Identity-based encryption with outsourced revocation in cloud computing, IEEE Trans. Comput. 64 (2) (2015) 425–437. IEEE.
[37] W. Shang, Z.M. Jiang, H. Hemmati, B. Adams, A.E. Hassan, P. Martin, Assisting developers of big data analytics applications when deploying on hadoop clouds, in: International Conference on Software Engineering, vol. 8114, IEEE, 2013, pp. 402–411.
[38] W.W. Lin, W.T. Wu Z, L.X. Lin, et al., An ensemble random forest algorithm for insurance big data analysis, IEEE Access 5 (2017) 16568–16575. IEEE.
[39] S. Rallapalli, R.R. Gondkar, U.P.K. Ketavarapu, Impact of processing and analyzing healthcare big data on cloud computing environment by implementing hadoop cluster, Procedia Comput. Sci. 85 (2016) 16–22.
[40] I.A.T. Hashem, V. Chang, N.B. Anuar, K. Adewole, I. Yaqoob, A. Gani, E. Ahmed, H. Chiroma, The role of big data in smart city, Int. J. Inf. Manag. 36 (5) (2016) 748–758.
[41] D. Larson, V. Chang, A review and future direction of agile, business intelligence, analytics and data science, Int. J. Inf. Manag. 36 (5) (2016) 700–710.
[42] Z. Farzanyar, N. Cercone, Efficient mining of frequent itemsets in social network data based on mapreduce framework, in: IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, IEEE, 2013, pp. 1183–1188.
[43] J. Leverich, C. Kozyrakis, On the energy (in)efficiency of hadoop clusters, ACM Sigops Oper. Syst. Rev. 44 (1) (2010) 61–65.
[44] J. Kim, J. Chou, D. Rotem, iPACS: Power-aware covering sets for energy proportionality and performance in data parallel computing clusters, J. Parallel Distrib. Comput. 74 (1) (2014) 1762–1774.
[45] R.T. Kaushik, M. Bhandarkar, GreenHDFS: Towards an energy-conserving, storage-efficient, hybrid Hadoop compute cluster, in: International Conference on Power Aware Computing and Systems, 2010.
[46] S. Li, T. Abdelzaher, M. Yuan, Tapa: Temperature aware power allocation in data center with map-reduce, in: 2011 International Green Computing Conference and Workshops (IGCC), IEEE, 2011, pp. 1–8.
[47] Redhat: Using CPUfreq Governors, 2014, URL: https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Power_Management_Guide/cpufreq_governors.html.
[48] S. Ibrahim, D. Moise, H.E. Chihoub, A. Carpen-Amarie, L. Bougé, G. Antoniu, Towards efficient power management in mapreduce: Investigation of cpu-frequencies scaling on power efficiency in Hadoop, Lecture Notes in Comput. Sci. 8907 (2014) 147–164.
[49] T. Wirtz, R. Ge, Z. Zong, Z. Chen, Power and energy characteristics of MapReduce data movements, in: 2013 International Green Computing Conference (IGCC), 2013.
[50] M. Malik, A. Sasan, R. Joshi, S. Rafatirah, Characterizing hadoop applications on microservers for performance and energy efficiency optimizations, in: IEEE International Symposium on PERFORMANCE Analysis of Systems and Software, IEEE, 2016, pp. 153–154.
[51] N. Maheshwari, R. Nanduri, V. Varma, Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework, Future Gener. Comput. Syst. 28 (1) (2012) 119–127.
[52] R. Xiong, J. Luo, F. Dong, Optimizing data placement in heterogeneous hadoop clusters, Clust. Comput. 18 (4) (2015) 1465–1480.
[53] S. Moon, J. Lee, X. Sun, Y.S. Kee, Optimizing the hadoop mapreduce framework with high-performance storage devices, J. Supercomput. 71 (9) (2015) 3525–3548.
[54] W. Yu, Y. Wang, X. Que, C. Xu, Virtual shuffling for efficient data movement in mapreduce, IEEE Trans. Comput. 64 (2) (2013) 556–568.
[55] J. Yu, Z. Hu, Y. Han, The research of measuring approach and energy efficiency for hadoop periodic jobs, Open Fuels Energy Sci. J. 8 (1) (2015) 206–210.
[56] J. Chen, D. Wang, L. Fu, W. Zhao, An improved small file processing method for hdfs, Int. J. Digit. Content Technol. Appl. 6 (20) (2012) 296–304.
[57] C. Vorapongkitipun, N. Nupairoj, Improving performance of small-file accessing in hadoop, in: 11th International Joint Conference on Computer Science and Software Engineering (JCSSE), IEEE, 2014, pp. 200–205.
[58] F. Tian, K. Chen, Towards optimal resource provisioning for running mapreduce programs in public clouds, in: IEEE International Conference on Cloud Computing, vol. 25, IEEE, 2011, pp. 155–162.
[59] M. Cardosa, A. Singh, H. Pucha, A. Chandra, Exploiting spatio-temporal tradeoffs for energy-aware mapreduce in the cloud, IEEE Trans. Comput. 61 (12) (2012) 1737–1751.
[60] E. Feller, L. Ramakrishnan, C. Morin, Performance and energy efficiency of big data applications in cloud environments: A Hadoop case study, J. Parallel Distrib. Comput. 79 (2015) 80–89.
[61] B. Sharma, T. Wood, C.R. Das, HybridMR: A hierarchical mapreduce scheduler for hybrid data centers, in: IEEE International Conference on Distributed Computing Systems, vol. 7973, IEEE, 2013, pp. 102–111.
[62] B. Palanisamy, A. Singh, L. Liu, Cost-effective resource provisioning for mapreduce in a cloud, IEEE Trans. Parallel Distrib. Syst. 26 (5) (2015) 1265–1279.
[63] P. Li, L. Ju, Z. Jia, Z. Sun, SLA-aware energy-efficient scheduling scheme for hadoop yarn, in: 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), and 2015 IEEE 12th International Conference on Embedded Software and Systems (ICESS), IEEE, 2015, pp. 623–628.
[64] Z. Niu, B. He, F. Liu, Not all joules are equal: Towards energy-efficient and green-aware data processing frameworks, in: IEEE International Conference on Cloud Engineering, IEEE, 2016, pp. 2–11.
[65] K.R. Krish, M.S. Iqbal, M.M. Rafique, A.R. Butt, Towards energy awareness in hadoop, in: Fourth International Workshop on Network-Aware Data Management, IEEE, 2014, pp. 16–22.
[66] B. Shi, A. Srivastava, Thermal and power-aware task scheduling for hadoop based storage centric datacenters, in: 2010 International Green Computing Conference, IEEE, 2010, pp. 73–83.
[67] E. Hwang, K.H. Kim, Minimizing cost of virtual machines for deadline-constrained mapreduce applications in the cloud, in: ACM/IEEE International Conference on Grid Computing, vol. 45, IEEE, 2012, pp. 130–138.
[68] N. Zhu, X. Liu, J. Liu, H. Yu, Towards a cost-efficient mapreduce: Mitigating power peaks for hadoop clusters, Tsinghua Sci. Technol. 19 (1) (2014) 24–32.
[69] P.P. Nghiem, S.M. Figueira, Towards efficient resource provisioning in mapreduce, J. Parallel Distributed Comput. 95 (C) (2016) 29–41.
[70] L. Mashayekhy, M.M. Nejad, D. Grosu, Q. Zhang, Energy-aware scheduling of mapreduce jobs for big data applications, IEEE Trans. Parallel Distrib. Syst. 26 (10) (2015) 2720–2733.
[71] Q. Althebyan, O. AlQudah, Y. Jararweh, Q. Yaseen, A scalable map reduce tasks scheduling: A threading-based approach, Int. J. Comput. Sci. Eng. 14 (1) (2017) 44–54.
[72] Q. Althebyan, Y. Jararweh, Q. Yaseen, O. Alqudah, M. Al-Ayyoub, Evaluating map reduce tasks scheduling algorithms over cloud computing infrastructure, Concurr. Comput. Pract. Exp. 27 (18) (2016) 5686–5699.
[73] Y.F. Wen, Energy-aware dynamical hosts and tasks assignment for cloud computing, J. Syst. Softw. 115 (2016) 144–156.
[74] I. Goiri, R. Bianchini, S. Nagarakatte, T.D. Nguyen, ApproxHadoop: Bringing approximations to mapreduce frameworks, in: 20th International Conference on Architectural Support for Programming Languages and Operating Systems, vol. 50, ACM, 2015, pp. 383–397.
[75] F. Yan, L. Cherkasova, Z. Zhang, E. Smirni, Optimizing power and performance trade-offs of mapreduce job processing with heterogeneous multi-core processors, in: IEEE International Conference on Cloud Computing, IEEE, 2014, pp. 240–247.
[76] Q. Zhu, L. Miao, The realization of green storage in hadoop, in: International Conference on Cloud Computing and Internet of Things, IEEE, 2015, pp. 91–95.
[77] Z. Asad, M.A.R. Chaudhry, D. Malone, Greener data exchange in the cloud: A coding-based optimization for big data processing, IEEE J. Sel. Areas Commun. 34 (5) (2016) 1360–1377.

[78] J.C. Lin, F.Y. Leu, Y.P. Chen, Impacts of task re-execution policy on MapReduce jobs, Comput. J. 59 (5) (2016) 701–714.
[79] T.D. Phan, S. Ibrahim, G. Antoniu, L. Bouge, On understanding the energy impact of speculative execution in hadoop, in: IEEE International Conference on Data Science and Data Intensive Systems, IEEE, 2015, pp. 396–403.
[80] J.C. Lin, F.Y. Leu, Y.P. Chen, PAReS: A proactive and adaptive redundant system for MapReduce, J. Inf. Sci. Eng. 31 (5) (2015) 1775–1793.
[81] Í. Goiri, R. Beauchea, K. Le, T.D. Nguyen, M.E. Haque, J. Guitart, J. Torres, R. Bianchini, Greenslot: Scheduling energy consumption in green datacenters, in: 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, 2011.
[82] Í. Goiri, K. Le, T.D. Nguyen, J. Guitart, J. Torres, R. Bianchini, GreenHadoop: leveraging green energy in data-processing frameworks, in: ACM European Conference on Computer Systems, ACM, 2012, pp. 57–70.
[83] C.H. Hsu, S.W. Poole, Power signature analysis of the specpower_ssj2008 benchmark, in: IEEE International Symposium on PERFORMANCE Analysis of Systems and Software, IEEE, 2011, pp. 227–236.
[84] Y.W. Chen, S.H. Hung, C.H. Tu, C.W. Yeh, Virtual hadoop: Mapreduce over docker containers with an auto-scaling mechanism for heterogeneous environments, in: International Conference on Research in Adaptive and Convergent Systems, ACM, 2016, pp. 201–206.
[85] X.Y. Wang, B. Lee, Y.S. Qiao, Experimental evaluation of memory configurations of hadoop in docker environments, in: Irish Signals and Systems Conference, IEEE, 2016.
[86] A. Siddiqa, A. Karim, V. Chang, SmallClient for big data: An indexing framework towards fast data retrieval, Clust. Comput. 20 (2) (2017) 1193–1208.

WenTai Wu is a master student in Computer Science at South China University of Technology. He received his bachelor degree in Computer Science from South China University of Technology in 2015. His research interests include distributed computing and cloud computing.

WeiWei Lin is currently an associate professor in the School of Computer Science and Engineering, South China University of Technology. His research interests include distributed system, cloud computing and big data.

Ching-Hsien Hsu is a professor in the Department of Computer Science and Information Engineering at Chung Hua University, Taiwan. His research includes high performance computing, cloud computing, big data intelligence, parallel and distributed systems, ubiquitous/pervasive computing and intelligence. Dr. Hsu is an IEEE senior member.

LiGang He is an associate professor in the Department of Computer Science, University of Warwick. His research mainly includes High Performance Computing, Cloud Computing and Parallel Processing.